A Hybrid ACO Algorithm for Capacitated Vehicle Routing Problems

Xiaoqi Sun
School of Electronic Informantion and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
Email: sun_xiaooo@sjtu.edu.cn

Yuzhuo Fu
School of Electronic Informantion and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
Email: yzfu@sjtu.edu.cn

Ting Liu
School of Electronic Informantion and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
Email: louisa_liu@sjtu.edu.cn

***Abstract*—The Capacitated Vehicle Routing Problem (CVRP) is a well-known combinatorial optimization problem. The traditional ant colony optimization (ACO) method usually can only find a local optimum of the CVRP, and the simulated annealing (SA) method can help ACO get trapped out of the local optima to explore new solutions. Thus, combining the idea of ant colony optimization (ACO) and simulated annealing (SA), we propose a novel hybrid algorithm named SACO to solve the CVRP. In addition, inspired by decomposed ACO algorithms, we extend our algorithm into the decomposed version named DSACO to improve the solution quality and decrease the computational time. We test the standard benchmark problems on both of the proposed two algorithms SACO and DSACO, and make a comparison with other existing ACO-based algorithms. The computational results show the advantages of the proposed algorithms.**

***Keywords*—CVRP, ant colony optimization, simulated annealing, decomposed algorithm.**

I. Introduction

The vehicle routing problem (VRP) is a classical combinatorial optimization problem which has been studied extensively for last five decades. The capacitated vehicle routing problem (CVRP) is one of the variants of VRPs, introduced by Dantzig et al. (1959). The aim of the CVRP is to find a set of vehicle routes, starting and ending at a same single depot, and fulfilling all customers' demands at minimum cost.

Since the CVRP belongs to the class of NP hard problems, no efficient exact algorithms are available for solving large size instances with reasonable computational time. Therefore, various of heuristic and meta-heuristic algorithms are developed to achieve approximate solution of the CVRP, such as ant colony optimization (ACO) method, simulated annealing (SA) method, genetic algorithms (GA) and tabu search (TS).

ACO is a population-based algorithm, inspired by the behaviour of ant colonies for food-seeking. The artificial ants construct solutions guided by the global pheromone they laid in previous iterations. After each ant has finished its current construction, the pheromone is then updated with a bias towards better solutions found. ACO was first introduced by Dorigo et al. [1] to solve the traveling salesman problem (TSP), which can be seen as a special case of the CVRP. By making some modifications, Bullnheimer et al. [2] first applied ant system algorithm with 2-opt-heuristic for the VRP and got impressive results. Later, Reimann et al. [3] proposed a decomposed ACO approach called D-Ants, which solves the large-scale CVRP by decomposing it into several small problems, and solves each small problem using ACO. However, the original ACO is prone to achieve the stagnation, due to the pheromone overaccumulation in some local optimal solutions. In this situation, some modified ACO algorithms are proposed to overcome this limitation. Chen and Ting [4] developed an improved ant colony system algorithm, in which the pheromone level of each edge is reset to the initial state if the solution is not improved after a given number of generations. Yu et al. [5] proposed an ACO method with a new pheromone updating strategy called ant-weight, and a mutation operation, so that the ant colonies can escape from local optima. Kao et al. [6] proposed a hybrid algorithm based on ACO and particle swarm optimization (PSO).

This paper proposes a new hybrid algorithm called SACO for solving the CVRP, which is based on the framework of ACO and is hybridized with the merits of SA. As is widely known, SA performs single-starting-point neighbourhood search, and its convergence speed heavily depends on the initial state. Note that ACO can rapidly achieve a local optima, which can be used as the initial solution of SA, and meanwhile SA can explore better solution using neighbourhood search. Then, an intuitional idea is to combine both of the two algorithms to search for a global optima. We additionally add a new pheromone perturbation strategy to further avoid pheromone stagnation. Inspired by the decomposed ACO approach [3], we further extend the SACO algorithm into the decomposed version called DSACO, which shows better performance comparing to SACO.

The remainder of this paper is organized as follows. In section II, we give the formulation of the SACO algorithm. In section III, we propose a decomposed SACO algorithm by extending the SACO algorithm into decomposed version. In section IV, we present simulated results of a comprehensive computational study. We draw the conclusion in the last section.

978-1-4673-8979-2/17/$31.00 ©2017 IEEE

II. SACO Algorithm

A. The Framework of SACO Algorithm

The SACO algorithm combines the merits of SA and ACO. In ACO algorithm, the artificial ants search good solutions in parallel based on the pheromone information from preceding search, and often construct a fair solution in short time. However, the ACO algorithm usually falls into local optima. The merit of SA is that it performs neighbourhood search in random directions near a given solution and can enlarge search space. Therefore, in the SACO algorithm, the SA algorithm will be triggered and perform neighbourhood search if the best-so-far solution S_{best} is not improved after a given number of ACO iterations K_b. When SA is enabled, the ACO algorithm is probably trapped in a local optimum with pheromone overaccumulation, then our approach will update pheromone tails after a better solution is achieved by the SA algorithm.

To further avoid pheromone overaccumulation in ACO, our approach employs a new pheromone perturbation strategy to resolve pheromone stagnation. When the best solution keeps unchanged after a given number of iterations K_t, our approach will add a pheromone perturbation, which is realized by decreasing the variance of pheromone. After pheromone perturbation, the excessive pheromone on some local optimal path is diluted, and the paths with less pheromone are assigned with more pheromone, so these paths will have more probabilities to be constructed in further iterations.

The SACO algorithm mainly consists of two processes: ACO (including solution construction, local search, pheromone updating, and pheromone perturbation) and SA neighbourhood search. A high level description of the algorithm is summarized in Algorithm 1. In the following, we describe each step in more detail.

Algorithm 1: SACO Algorithm

```
Initialize the system (parameters and pheromone matrix);
repeat
    Construct a new solution;  (Step 1)
    Perform local search (2-opt and swap operations);
     (Step 2)
    Update pheromone;  (Step 3)
    if t_cnt = K_t then
        Apply pheromone perturbation strategy;  (Step 4)
        t_cnt ← t_cnt − 2;
    end
    if b_cnt = K_b then
        Run SA (Algorithm 2) with the initial solution
         S_best;  (Step 5)
        b_cnt ← 0;
    end
until termination condition met;
```

The parameters in SACO are as follows:

S_{best}: Best-so-far solution;

t_{cnt}: Number of iterations that the best solution keeps unchanged;

b_{cnt}: Number of iterations that the best-so-far solution is not improved;

K_t: the upper bound of t_{cnt};

K_b: the upper bound of b_{cnt}.

B. Solution Construction

In ACO, an individual ant represents a vehicle, starting from the depot 0, and continuously constructs tour by selecting customers until all customers are visited. At each construction step, ant k applies a probabilistic action choice rule to decide which customer to visit next. The probabilistic formula with which ant k from present customer i, chooses the next customer j is

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \times \eta_{ij}^\beta}{\sum_{l \in \mathcal{N}_i^k} \tau_{il}^\alpha \times \eta_{lj}^\beta} & \text{if } j \in \mathcal{N}_i^k, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathcal{N}_i^k$ represents the feasible neighborhood of ant k at current node i, τ_{ij} represents the pheromone trail on edge (i,j), η_{ij} represents the heuristic information between edge (i,j), α and β are the parameters determining the relative influence of pheromone trail and heuristic information.

C. Local Search

After the ants have constructed solutions, local search is applied to improve the quality of the solution of each ant. We first apply a local search based on swap moves, which is proposed by Osman [7]. According to [8] we then apply the 2-opt exchange. The 2-opt exchange operation randomly selects two nodes in a same vehicle route, and then iteratively inverts all nodes between these two nodes until no further improvements can be made. The two operations (swap and 2-opt) are often used in ACOs ([3], [4], [5], [8], [9]). In the following, we will explain both of the two operations in details.

D. Pheromone Updating

Pheromone updating is the key feature of ACO algorithm which is used to improve the future solutions. There are several strategies for pheromone updating, such as elitist ant system (EAS), rank-based ant system (AS_{rank}) and MAX-MIN ant system (MMAS). SACO applies AS_{rank} for pheromone updating, which is proposed by Bullnheimer et al. [10], and is defined as follows:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{r=1}^{n_e-1}(n_e - r)\Delta\tau_{ij}^r + n_e\Delta\tau_{ij}^*, \tag{2}$$

where ρ is the pheromone evaporation rate, n_e is the number of elite ants in each iteration, $\Delta\tau_{ij}^* = 1/C^*$ and $\Delta\tau_{ij}^r = 1/C^r$. Here, C^* and C^r are the objective function values of solutions. In each iteration only the (n_e-1) best-ranked ants and the ant with best-so-far solution are allowed to deposit pheromone. $\Delta\tau_{ij}^r$ with weight $(n_e - r)$ and $\Delta\tau_{ij}^*$ with weight n_e are the pheromone deposited by them respectively.

E. Pheromone Perturbation

Since only elite ants and best-so-far ant are allowed to deposit pheromone, the pheromone on paths of these ants will increase rapidly, and may lead to pheromone stagnation. SACO employs a new pheromone perturbation strategy to avoid pheromone stagnation. The rule of this proposed strategy is as follows:

$$\tau_{ij} = \delta\overline{\tau} + (1-\delta)\tau_{ij} \tag{3}$$

where $\overline{\tau}$ is the average of current pheromone, and $0 \leq \delta \leq 1$ is the perturbation ratio which determines the intensity of averaging. If δ is too big, every τ_{ij} will be very close to their average, which seems like resetting the ACO algorithm. On the other hand, if it is too small, the effect of perturbation will not be evident.

After pheromone perturbation, ant k at node i has more chances to explore different edges rather keeping selecting the same next node to move to. If ACO is still trapped in the same local optima, then the pheromone perturbation strategy will be applied repeatedly. Obviously, such an iteration will finally make every τ_{ij} achieve their average. Hence, pheromone perturbation can ensure the escape from local optima and increase the probability of searching better solution.

F. Searching using SA algorithm

Though pheromone perturbation helps ACO escape from local optima, it can not guarantee a better solution. SACO employs SA to further explore better solutions.

SA is a local search meta-heuristic that simulates the annealing process of solid. SA optimizes a solution by starting at a high initial temperature T_0, cooling it with neighbourhood search until a final temperature T_f is reached. In the process of simulated annealing, better solutions are always accepted, whereas worse solutions are accepted with a probability, which is defined as follows:

$$\begin{aligned} P(S^{'}) &= e^{-\frac{\Delta}{T(t)}} \\ \Delta &= C(S^{'}) - C(S) \end{aligned} \tag{4}$$

where S is the current solution, $S^{'}$ is the new solution, $C(S)$ is the objective function value of S, $C(S^{'})$ is the new objective function value of $S^{'}$, $T(t)$ is the current temperature, and $P(S^{'})$ is the probability that SA accepts the new solution $S^{'}$. At each temperature $T(t)$, SA performs neighbourhood search Z times. Then the temperature is reduced to $T(t+1) = \lambda T(t)$, where $0 < \lambda < 1$.

In SACO, SA performs three types of neighbourhood search operations: swap, inversion (or 2-opt) and insertion. The swap operation randomly selects two customers and then swaps their positions. Note that these two customers can be either from a single vehicle route or two vehicle routes. The inversion operation selects a vehicle route at random, choose two customers in this route randomly, and then inverts all customers between these two customers. The insertion operation selects a customer randomly, and then inserts this customer in a random position.

In SACO, SA starts with the best-so-far solution S_{best} as the initial solution, and randomly selects one of the three operations to perform each neighbourhood search. When a better solution is found, the pheromone in the path of this solution will be updated. Further, to avoid repeated search on a neighbourhood solution that has been previously visited within a certain short-term period, a tabu list with size SZ_{tabu} is applied to SA. The process of SA is summarized in Algorithm 2.

Algorithm 2: The Process of SA in SACO

```
S_0 ← S_best; S ← S_best;
tabuList ← []; T ← T_0; cnt ← 0;
while T > T_f do
    Generate solution S' in the neighborhood of S;
    if tabuList contains S' then
        Skip this loop;
    else
        Add S' to tabuList;
    end
    Δ ← C(S') − C(S);
    if Δ < 0 then
        Accept the new solution S ← S';
    else
        Generate rnd ← U(0,1) randomly;
        if rnd < e^(−Δ/T(t)) then
            Accept the new solution S ← S';
        end
    end
    if C(S) < C(S_best) then
        Update best-so-far solution S_best ← S;
        Update pheromone in the path of new solution S;
    end
    if cnt = Z then
        Reduce the temperature T ← λT;
        cnt ← 0;
    end
end
```

III. DSACO Algorithm

As stated above, the DSACO algorithm decomposes problem based on the divide and conquer strategy of D-Ants, which is proposed by Reimann [3]. Let us briefly explain this strategy. First, an ACO solves the master problem for a given number of iterations it_{master}, and gets the best-so-far solution S_{best}. D-Ants then clusters the vehicle routes of S_{best} according to their centers of gravity. Each resulting cluster (subproblem) is then solved independently by an ACO for a given number of iterations it_{sub}. After all subproblems have been solved, the pheromone of master problem is updated if a better solution is found. The steps described above are repeated until the termination condition is met. For more details about the strategy of D-Ants see [3].

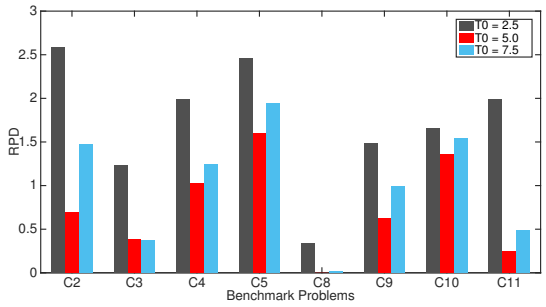

Fig. 1. Average RPD histogram with different T_0.

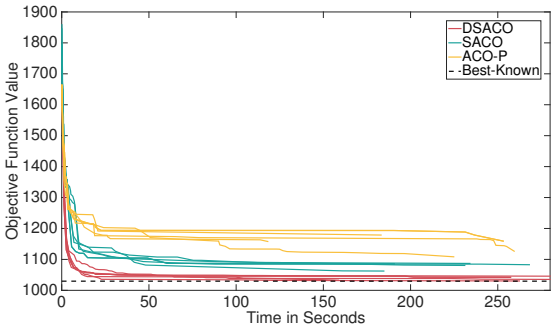

Fig. 2. Convergence trends of three algorithms tested on problem C4.

In DSACO, SACO is the the specific ACO for solving the master problem and subproblems iteratively. Since data communication between subproblems and the master process is only happened when all subproblems are solved, each SACO for each subproblem performs concurrently. The paper [3] did not involve the details on how the master pheromone is reinforced by subproblems, and in this paper, we give our detailed strategy. The strategy is as follows:

$$\begin{aligned} ratio &= C(S_{sub})/C(S_{master}) \\ \tau_{ij}^{master} &= \tau_{ij}^{master} + R \times \tau_{ij}^{sub} \times ratio \end{aligned} \tag{5}$$

where $C(S_{sub})$ and $C(S_{master})$ are the objective function value of a subproblem solution and the master problem best-so-far solution respectively, τ_{ij}^{master} and τ_{ij}^{sub} are the master pheromone and subprobem pheromone respectively and, R is the reinforcement coefficient.

IV. Numerical Analysis

The SACO and DSACO algorithms described in the previous sections are coded in GNU C++ and are executed on an Intel Xeon Processor (16 Cores, 2.30GHz, 40MB Cache). The project is available at github: https://github.com/i-sunny/cvrp_aco. A set of benchmark problems with 14 classical test problems from Christofides et al. [12] are selected to evaluate the effectiveness of both SACO and DSACO. The benchmark set can be downloaded from the website: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/. For each instance in the benchmark set, we have run our algorithms repeatedly 20 times within the time limit of $600s$.

The parameters of SACO are number of elite ants $n_e = 6$, $\alpha = 1$, $\beta = 2$, $\rho = 0.1$, $K_t = N$, $K_b = 5$, $\delta = 0.7$ and number of ants $nAnts = N$. The parameters of SA process are $Z = max(4 \times N, 250)$, $T_f = T_0/50$, $\lambda = 0.97$, and $SZ_{tabu} = 3$. DSACO uses the parameter values that were found to be favorable in [3] for D-Ants, and these are $it_{master} = 1$, $it_{sub} = 75$, number of subproblems $n_{sub} = N/50$ and reinforcement coefficient $R = 0.1$. In the preliminary experiment, we see that the effectiveness of the proposed algorithms is highly affected by the initial temperature T_0. A small T_0 narrows neighbourhood search space of SA, and the diversity can not be guaranteed. On the other hand, a large one indicates the sufficient diversity, but the precision of optimal solution may have lesser improvement. We test DSACO with different values for T_0 ($T_0 = 2.5$, 5.0, 7.5) on problems (C2, C3, C4, C5, C8, C9, C10 and C11) and summarize the average relative percentage deviation (RPD) from the best known solutions in Fig. 1. The histogram reveals that DSACO with $T_0 = 5.0$ yields the lowest average RPD among all three different values. Hence, in the remainder of this section we set $T_0 = 5.0$.

To evaluate the hybrid and decomposed strategies, we compare the three algorithms, SACO, DSACO and ACO with pheromone perturbation (ACO-P). Table I lists the computational results of these three algorithms on 14 benchmark problems. The best solution, the average solution, the worst solution and the average computational time on each problem are summarized. The asterisked numbers in boldface are the results that equal to the best-known solutions. The table reveals that for almost all test problems DSACO has better performance than the other two in term of solution quality. DSACO obtains reasonable good solution for most of problems, in which ten of fourteen reach the best-known solutions. SACO can also successfully obtain the best-known solutions in eight problems, while ACO-P can only obtain two. Compared with ACO-P, SACO generally provides better solutions for 14 problems, except the problem C10. This improvement is attributed to that the SA process can enlarge search space and explore better solutions near local optimum. The SA process also increases the computational time. Further, the DSACO algorithm, which integrates the SA process and decomposed strategy, significantly enhances the efficiency compared with SACO.

We further compare the convergence speed of the three algorithms on problem C4. Fig. 2 shows the convergence trends by five independent runs for each algorithm on problem C4. The curves reveal that DSACO converges fastest, SACO next, and ACO-P converges the most slowly. Moreover, only DSACO successfully obtains the best-known solution. It demonstrates that, with decomposed strategy, DSACO yields an improvement both in solution quality and computational time.

We compare DSACO with other four ACO-based algorithms, who are SbAS in [13], IACS in [4], IACO in [5] and PACO in [6]. The Table II lists the best solutions obtained by the above ACO-based algorithms. The overall results indicate

TABLE I
COMPUTATIONAL RESULTS OF DSACO, SACO AND ACO-P.

No.	DSACO				SACO				ACO-P			
	Best	Avg.	Worst	AT(s)	Best	Avg.	Worst	AT(s)	Best	Avg.	Worst	AT(s)
C1	**524.61***	524.77	527.71	4.88	**524.61***	525.04	533.00	1.60	**524.61***	526.18	533.00	17.95
C2	**835.26***	841.07	849.70	111.59	**835.26***	840.86	848.08	139.76	856.88	869.04	891.46	110.47
C3	**826.14***	829.27	835.28	114.38	832.01	837.70	848.53	186.12	861.20	873.00	885.63	162.23
C4	1029.64	1038.93	1049.54	368.56	1062.33	1079.32	1089.40	192.91	1090.47	1137.14	1178.71	192.15
C5	1298.74	1312.15	1329.45	449.03	1398.11	1429.18	1481.62	416.21	1483.49	1521.80	1550.30	185.03
C6	**555.43***	**555.43***	**555.43***	35.15	**555.43***	555.49	556.68	40.88	**555.43***	559.80	565.56	25.09
C7	**909.68***	910.95	916.89	82.43	**909.68***	911.33	916.89	75.10	913.94	928.63	940.65	102.36
C8	**865.94***	867.95	875.90	54.37	**865.94***	867.99	876.56	238.77	870.60	880.75	889.87	118.72
C9	1164.12	1169.76	1179.22	227.85	1164.08	1185.79	1208.91	392.24	1164.40	1172.80	1197.29	118.65
C10	1405.27	1414.76	1434.43	411.51	1446.68	1464.15	1496.63	393.62	1419.04	1440.88	1485.07	234.43
C11	**1042.11***	1044.67	1047.64	162.40	**1042.11***	1043.17	1044.80	384.17	1056.61	1066.35	1080.94	132.92
C12	**819.56***	819.56	819.60	13.81	**819.56***	819.57	819.60	30.67	846.98	847.22	848.61	64.82
C13	**1541.14***	1546.12	1550.99	174.81	1547.80	1550.79	1553.97	321.22	1548.46	1551.98	1555.49	108.39
C14	**866.37***	**866.37***	**866.37***	9.81	**866.37***	**866.37***	**866.37***	50.85	866.79	866.80	866.95	68.67

TABLE II
COMPARISON OF DSACO WITH OTHER ACO-BASED ALGORITHMS.

No.	SbAS [13]	IACS [4]	IACO [5]	PACO [6]	DSACO
C1	524.63	**524.61***	**524.61***	**524.61***	**524.61***
C2	838.60	836.18	**835.26***	**835.26***	**835.26***
C3	828.67	835.60	830.00	829.92	**826.14***
C4	1040.09	1038.22	**1028.42***	1040.23	1029.64
C5	1303.53	1327.22	1305.50	1348.73	1298.74
C6	**555.43***	**555.43***	**555.43***	**555.43***	**555.43***
C7	**909.68***	**909.68***	**909.68***	**909.68**	**909.68***
C8	866.87	**865.94***	**865.94***	868.61	**865.94***
C9	1171.34	1173.76	**1162.55***	1171.94	1164.12
C10	1416.05	1413.83	**1395.85***	1454.81	1405.27
C11	**1042.11***	**1042.11***	**1042.11***	**1042.11***	**1042.11***
C12	**819.56***	832.67	**819.56***	**819.56***	**819.56***
C13	1545.12	1547.07	1545.93	1562.64	**1541.14***
C14	**866.37***	**866.37***	**866.37***	**866.37***	**866.37***

that DSACO is competitive with other ACO-based algorithms in terms of solution quality. In particular, the table reveals that our DSACO performs better than SbAS, IACS and PACO for all the problems. It can also be observed that both DSACO and IACO obtain best-known solutions for eight problems in common. Besides that, DSACO performs better than IACO for problems C3, C5 and C13, while IACO performs better for problems C4, C9 and C10.

V. CONCLUSION

This paper proposes a hybrid algorithm, SACO, which combines the merits of SA and ACO. ACO first explores solutions with pheromone perturbation strategy. When ACO fails to improve the best-so-far solution continuously, the SA algorithm is used to perform neighbourhood search and explore better solutions. Furthermore, we present DSACO, the decomposed version of SACO, to enhance both the effectiveness and efficiency of SACO. Computational results show that DSACO is competitive with other ACO-based algorithms in terms of solution quality.

ACKNOWLEDGMENT

This work is financed by the Innovation Program of Shanghai Municipal Education Commission [14ZZ018] and by the National Natural Science Foundation of China [61472244].

REFERENCES

[1] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation,* vol. 1, no. 1, pp. 53-66, 1997.

[2] B. Bullnheimer, R. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem," in *Second Metaheuristics International Conference,* MIC97, Sophia-Antipolis, France.

[3] M. Reimann, K. Doerner, and R. F. Hartl, "D-ants: Savings based ants divide and conquer the vehicle routing problem," *Computers & Operations Research,* vol. 31, no. 4, pp. 563-591, 2004.

[4] C. Chen and C. Ting, "An improved ant colony system algorithm for the vehicle routing problem,"*Journal of the Chinese Institute of Industrial Engineers,* vol. 23, no. 2, pp. 115-126, 2006.

[5] B. Yu, Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *European journal of operational research,* vol. 196, no. 1, pp. 171-176, 2009.

[6] Y. Kao, M. H. Chen, and Y. T. Huang, "A Hybrid Algorithm Based on ACO and PSO for Capacitated Vehicle Routing Problems," *Mathematical Problems in Engineering,* vol. 2012, Article ID 726564, 2012.

[7] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of Operations Research,* vol. 41, no. 4, pp. 421-451, 1993.

[8] B. Bullnheimer, R. F. Hartl, and C. Strauss, "An improved ant system algorithm for the vehicle routing problem," *Annals of Operations Research,* vol. 89, pp. 319-328, 1999.

[9] B. Yu, Z. Z Yang, and B. Z. Yao, "A hybrid algorithm for vehicle routing problem with time windows," *Expert Systems with Applications,* vol. 38, no. 1, pp. 435-441, 2011.

[10] B. Bullnheimer, R. F. Hard, and C. Strauss, "A new rank-based version of the Ant System: A computational study," *Central European Journal for Operations Research and Economics,* vol. 7, no. 1, pp. 25-38, 1999.

[11] Y. Xiao, Q. Zhao, I. Kaku, and N. Mladenovic, "Variable neighbourhood simulated annealing algorithm for capacitated vehicle routing problems," *Engineering Optimization,* vol. 46, no. 4, pp. 562-579, 2014.

[12] N. Christofides, A. Mingozzi, and P. Toth, "The vehicle routing problem," in *Combinatorial Optimization,* N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, Eds., Wiley, Chichester, UK, 1979.

[13] M. Reimann, M. Stummer, and K. Doerner. "A Savings Based Ant System For The Vehicle Routing Problem," *GECCO,* pp. 1317-1326, 2002.