

Research Article

A Hybrid Algorithm Based on ACO and PSO for Capacitated Vehicle Routing Problems

Yucheng Kao, Ming-Hsien Chen, and Yi-Ting Huang

Department of Information Management, Tatung University, 40 Chungshan N. Road, Sec. 3, Taipei 104, Taiwan

Correspondence should be addressed to Yucheng Kao, ykao@ttu.edu.tw

Received 24 February 2012; Revised 5 May 2012; Accepted 8 May 2012

Academic Editor: Jung-Fa Tsai

Copyright © 2012 Yucheng Kao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem. It has been studied for several decades because finding effective vehicle routes is an important issue of logistic management. This paper proposes a new hybrid algorithm based on two main swarm intelligence (SI) approaches, ant colony optimization (ACO) and particle swarm optimization (PSO), for solving capacitated vehicle routing problems (CVRPs). In the proposed algorithm, each artificial ant, like a particle in PSO, is allowed to memorize the best solution ever found. After solution construction, only elite ants can update pheromone according to their own best-so-far solutions. Moreover, a pheromone disturbance method is embedded into the ACO framework to overcome the problem of pheromone stagnation. Two sets of benchmark problems were selected to test the performance of the proposed algorithm. The computational results show that the proposed algorithm performs well in comparison with existing swarm intelligence approaches.

1. Introduction

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem in which the computational complexity is NP-hard. The capacitated vehicle routing problem is one of the variants of VRPs. The objective of CVRPs is to minimize the total traveling distance of vehicles which serve a set of customers. The following constraints are considered in a typical CVRP: each route is a tour which starts from a depot, visits a subset of the customers, and ends at the same depot; each customer must be assigned to exactly one of the vehicles; each customer has its own demand and the total demand of customers assigned to a vehicle must not exceed the vehicle capacity. In the past decades, researchers proposed different strategies to solve the CVRP. One of them is to cluster customers into different routes and then to arrange the visiting sequence for each route. The objective function value will

be apparently influenced by the results of customer clustering and sequencing. For more detailed descriptions of vehicle routing problems, the reader may refer to the articles by Laporte [1], Osman [2], and Cordeau et al. [3].

Because the CVRP is an NP-hard problem [4], the optimal solution of a large-size instance cannot be found within a reasonable time. To overcome this difficulty, many classical heuristic methods and metaheuristic methods are proposed in the past five decades. Some metaheuristic algorithms can provide competitive solutions to the CVRP, such as Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithm (GA). Metaheuristic algorithms have some advantages, for example, the abilities to escape from local optima through stochastic search, to speed convergence using solution replacement, to guide the search direction with the elitist strategy, and so on. The following paragraph gives a brief review of some articles which use these metaheuristic algorithms to solve CVRPs.

Barbarosoglu and Ozgur [5] designed a TS-based algorithm using a new neighborhood generation procedure for the single-depot vehicle routing problems. The neighbors are defined by using two procedures: the first one ignores the scattering patterns of customer locations, and the second one considers the underlying clustering of customer locations. Baker and Ayechev [6] put forward a hybrid of GA algorithms with neighborhood search methods. The pure GA has three specific processes: initialization, reproduction, and replacement. The neighborhood search methods are used to accelerate the convergence of GA. Computational results showed that this approach is competitive with published results obtained using TS and SA. Lin et al. [7] proposed a hybrid algorithm which takes the advantages of SA and TS. In their paper, SA is used to adjust the probability of accepting worse solutions according to the extent of solution improvement and the annealing temperature, while TS is embedded in the framework of SA to avoid cycling to some extent while searching for neighborhood.

In the recent ten years, swarm intelligence, a new category of metaheuristics, has emerged and attracted researchers' attention. Swarm intelligence mimics the social behavior of natural insects or animals to solve complex problems. Some commonly used swarm intelligence algorithms for the solution of the CVRP include ant colony optimization (ACO), particle swarm optimization (PSO) and artificial bee colony [8].

ACO is a population-based swarm intelligence algorithm and was proposed by Dorigo and Gambardella [9]. This algorithm has been inspired by the foraging behavior of real ant colonies and originally designed for the traveling salesman problem (TSP). The artificial ants use pheromone laid on trails as an indirect communication medium to guide them to construct complete solution routes step by step. More pheromone deposits on better routes attract more ants for later search. This effect is called dynamic positive feedback and helps speed convergence of ACO. Recently, some researchers have studied vehicle routing problems using ACO algorithms. Applying ACO to the CVRP is quite natural, since we can view ant nests as depots, artificial ants as vehicles, foods as customers, and trails as routes. Some of the relevant papers are briefly reviewed as follows.

Bell and McMullen [10] made modifications of the ACO algorithm in order to solve the vehicle routing problem. They used multiple ant colonies to search vehicle routes. Each vehicle route is marked with unique pheromone deposits by an ant colony, but the communication among ant colonies is limited. Later, Liu and Cai [11] proposed a new multiple ant colonies technique, which allows ant colonies to communicate with each other in order to escape from local optima. Chen and Ting [12] developed an improved ant colony system algorithm, in which pheromone trails will be reset to initial values for restarting the search if the solution is not improved after a given number of iterations. Zhang and

Tang [13] hybridized the solution construction mechanism of ACO with scatter search (SS). The algorithm stores better solutions in a reference set. Some new solutions are generated by combining solutions selected from the reference set, and some are produced by using the conventional ACO method. Yu et al. [14] also developed an improved ant colony optimization for vehicle routing problems. Their algorithm uses the ant-weight strategy to update pheromone in terms of solution quality and the contribution of each edge to the solution. Lee et al. [15] proposed an enhanced ACO algorithm for the CVRP. Their algorithm adopts the concept of information gain to measure the variation of pheromone concentrations and hence to dynamically adjust the value of heuristic parameter (β) which determines the importance of heuristic value (η) at different iterations.

PSO is also a population-based swarm intelligence algorithm and was originally proposed by Kennedy and Eberhart [16]. PSO is inspired by social behavior of bird flocking. It has been shown that PSO can solve continuous optimization problems very well. In the PSO, solution particles try to move to better locations in the solution space. The movements of particles are guided by the individuals' and the swarm's best positions. PSO can converge very fast due to its two unique mechanisms: memorizing personal best experiences (P_{best}) and information sharing of global best experiences (G_{best}). Note that the G_{best} solution of the particle swarm is equal to the P_{best} solution of the best particle. In 1998, Shi and Eberhart [17] enhanced PSO by adding the concept of inertia weight, which becomes the standard version of PSO.

PSO being originally developed for continuous optimization problems, a special solution representation or solution conversion should be designed first in order to solve CVRPs. Chen et al. [18] first proposed a PSO-based algorithm to solve the CVRP. In their approach, each iteration has two main steps: customers are first clustered by using a discrete PSO algorithm (DSPO) and then sequenced by applying a SA algorithm. Due to its long solution strings, their approach takes much computational time in solving large scale problems. To improve Chen et al.'s work, Kao and Chen [19] addressed a new solution representation and solved the CVRP with a combinatorial PSO algorithm. Ai and Kachitvichyanukul [20] presented two solution representations for solving CVRPs. For example, in their second solution representation (SR-2), each vehicle is represented in three dimensions, with two for the reference point and one for the vehicle coverage radius. SR-2 employs these points and radius to construct vehicle routes. The particle solutions are adjusted by using a continuous PSO. Marinakis et al. [21] proposed a hybrid PSO algorithm to tackle large-scale vehicle routing problems. Their proposed algorithm combines a PSO algorithm with three heuristic methods, with the first for particle initialization, the second for solution replacement, and the third for local search.

This study proposes a new hybrid algorithm for the capacitated vehicle routing problem, which is based on the framework of ACO and is hybridized with the merits of PSO. The reasons why ACO and PSO, rather than SA, TS, and GA, are adopted in the proposed algorithm are given as follows. First, SA and TS perform the so-called single-starting-point search and thus their performance relies highly on a good initial solution. However, GA, ACO, and PSO are all population-based algorithms and can start the search from multiple points. Their initial solutions have little influence on their performance. Thus, we consider adopting the population-based algorithms to solve CVRPs. Second, ACO and PSO have memory that enables the algorithms to retain knowledge of good solutions, while the genetic operators of GA may destroy previously learned knowledge when producing the offspring. In view of these two considerations, we select ACO and PSO as the solution approach for this paper.

In the past, most relevant papers adopted either ACO [10–15] or PSO [18–21] alone without trying to use both in combination for solving CVRPs. In this paper, we try to integrate ACO with PSO to develop a new hybrid approach which can take advantage of both algorithms. That is, the proposed algorithm uses the solution construction approach of ACO to cluster customers and build routes at the same time and use the short-term memory inspired by PSO to speed convergence through laying pheromone on the routes of G_{best} and P_{best} solutions.

Like most ACO-based algorithms, the proposed hybrid algorithm also faces the limitation of pheromone stagnation, which results in premature convergence. To solve this problem, Shuang et al. [22] employed the mechanism of PSO to modify the pheromone updating rules of ACO. Their proposed algorithm is called PS-ACO and is used to solve traveling salesman problems (TSPs). PS-ACO can improve the performance of ACO to some extent, but it may still be trapped in local optima due to the overaccumulation of pheromone on some edges when solving more complicated problems like CVRPs. To attain a high degree of search accuracy, this paper proposes a pheromone disturbance approach to overcome the problem of pheromone stagnation. The remainder of this paper is organized as follows. Section 2 defines the mathematical formulation of the CVRP. The proposed methodology is described in Section 3. Section 4 presents computational results. Finally, conclusions are drawn in the last section.

2. Mathematical Model of CVRP

This section gives a typical mathematical formulation of the CVRP, including notations, objective function, and constraint equations.

Notations. 0: index of depots;

N : total number of customers;

K : total number of vehicles;

C_{ij} : cost incurred when traveling from customer i to customer j ;

S_i : service time needed for customer i , $S_0 = 0$;

Q : maximum loading capacity of a vehicle;

T : maximum traveling distance of a vehicle;

d_i : demand of customer i , $d_0 = 0$;

X_{ij}^k : 0-1 variable, where $X_{ij}^k = 1$ if the edge from customer i to customer j is traveled by vehicle k ; otherwise, $X_{ij}^k = 0$. Note that $i \neq j$;

p : penalty coefficient;

R : set of customers served by a vehicle, and $|R|$ is the cardinality of R .

Objective function

$$\text{Minimize } \sum_{i=0}^N \sum_{j=0}^N \sum_{k=1}^K C_{ij} X_{ij}^k, \quad (2.1)$$

$$\text{subject to } \sum_{k=1}^K \sum_{i=0}^N X_{ij}^k = 1, \quad j = 1, 2, \dots, N, \quad (2.2)$$

$$\sum_{k=1}^K \sum_{j=0}^N X_{ij}^k = 1, \quad i = 1, 2, \dots, N, \quad (2.3)$$

$$\sum_{i=0}^N X_{iu}^k - \sum_{j=0}^N X_{uj}^k = 0, \quad k = 1, 2, \dots, K; \quad u = 1, 2, \dots, N, \quad (2.4)$$

$$\sum_{i=0}^N \sum_{j=0}^N X_{ij}^k d_i \leq Q, \quad k = 1, 2, \dots, K, \quad (2.5)$$

$$\sum_{i=0}^N \sum_{j=0}^N X_{ij}^k (C_{ij} + S_i) \leq T, \quad k = 1, 2, \dots, K, \quad (2.6)$$

$$\sum_{j=1}^N X_{ij}^k = \sum_{j=1}^N X_{ji}^k \leq 1, \quad i = 0; \quad k = 1, 2, \dots, K, \quad (2.7)$$

$$\sum_{i,j \in R} X_{ij}^k \leq |R| - 1, \quad R \subseteq \{1, \dots, N\}, \quad 2 \leq |R| \leq N - 1; \quad k = 1, 2, \dots, K, \quad (2.8)$$

$$X_{ij}^k \in \{0, 1\}, \quad i, j = 0, 1, \dots, N; \quad k = 1, 2, \dots, K. \quad (2.9)$$

Equation (2.1) is the objective function of the CVRP. Equations (2.2) and (2.3) ensure that each customer can be served by only one vehicle. Equation (2.4) maintains the continuity at each node for every vehicle. Equation (2.5) ensures that the total customer demand of a vehicle cannot exceed its maximum capacity. Similarly, (2.6) ensures that the total route distance of a vehicle cannot exceed its route length limit. Equation (2.7) makes sure that every vehicle can be used at most once and must start and end at the depot. The subtour elimination constraints are given in (2.8). Equation (2.9) is the integrality constraint.

3. PACO Algorithm

This section describes the proposed solution algorithm to the capacitated vehicle routing problem. The algorithm, called PACO, hybridizes the solution construction mechanism of

ACO and the short-term memory mechanism of PSO to find optimal or near optimal vehicle routes.

3.1. Basic Idea

The PACO algorithm incorporates the merits of PSO into the ACO algorithm. One of the advantages of applying ACO to the CVRP is that ACO can cluster customers and build routes at the same time. However, laying pheromone (long-term memory) on trails as ant communication medium is time consuming. The merit of PSO is that it can speed convergence through memorizing personal and global best solutions to guide the search direction. Inspired by the merit of PSO, the PACO algorithm allows artificial ants to memorize their own best solution so far and to share the information of swarm best solution. Hence PACO can speed convergence through intensifying pheromone on routes of G_{best} and P_{best} solutions.

To avoid falling into local optima, our approach employs elitist strategy, pheromone disturbance, and short-term memory resetting to resolve pheromone stagnation. After ants complete solution construction, the first r iteration-best ants are allowed to perform local search to improve their current solutions. After that, all ants update their P_{best} solutions and G_{best} solution. The ants with better P_{best} solutions are called elite ants. Pheromone updating is conducted by these elite ants only. Elite ants lay pheromone on their P_{best} solution routes in a distributed way. The elitist strategy updates the pheromone in terms of solution quality and attracts ants searching for solutions around distributed P_{best} solution paths.

When the G_{best} solution is not improved within a given number of iterations, PACO will carry out pheromone disturbance to change pheromone trails randomly in order to find new solutions. Pheromone disturbance can prevent the paths of the P_{best} solution of elite ants from becoming too dominant. After pheromone disturbance, the paths of the P_{best} solution of elite ants may become dominant again because they can still evoke memories of current P_{best} solutions which determine the way of laying pheromone. To avoid that, the algorithm should allow ants to reset their P_{best} solutions. It means that the ants will discard their current G_{best} and P_{best} solutions and find new ones in the following iteration according to the disturbed pheromone trails. The algorithm allows any ant, not limited to elite ants, to change their P_{best} solutions if and only if its P_{best} solution is very similar to the G_{best} solution.

3.2. Solution Representation

For solving the CVRP, each artificial ant represents a candidate solution. A solution is represented with the route representation of all vehicle tours. Let 0 denote the depot and positive integers from 1 to N represent the customers. Suppose that the total number of vehicles is K . The solution code is a permutation of 1 to N and $(K - 1)$ zeros. Zeros divide a solution code into K segments, with each of them representing a vehicle route. For example, a solution code (1, 2, 3, 0, 4, 5, 6, 0, 7, 8, 9) means that customers 1, 2, 3 are serviced by vehicle 1; customer 4, 5, 6 by vehicle 2; customers 7, 8, 9 by vehicle 3. Thus, the length of a solution string is $(N + K - 1)$. It is clear that this solution representation scheme can fully meet the constraints defined in (2.2)–(2.4).

The goodness of a solution is evaluated by using the objective function (2.1). Here, we modify the objective function in order to handle the infeasible solutions violating capacity

and length constraints (see (2.5) and (2.6)). Two penalty functions are added to the original objective function, as defined in (3.1), one for excess vehicle capacity and the other for excess route length. These penalty functions increase the objective function value of infeasible solutions so as to prefer feasible solution to be selected as elite ants which are allowed to perform local search and to update pheromone on their routes (see Sections 3.5 and 3.6).

$$\text{Minimize } f = \sum_{i=0}^N \sum_{j=0}^N \sum_{k=1}^K C_{ij} X_{ij}^k + p \sum_{k=1}^K \left[\max \left\{ 0, \sum_{i=0}^N \sum_{j=0}^N X_{ij}^k d_i - Q \right\} + \max \left\{ 0, \sum_{i=0}^N \sum_{j=0}^N X_{ij}^k (C_{ij} + S_i) - T \right\} \right]. \quad (3.1)$$

3.3. Main Steps

The main steps of PACO are solution construction, local search, G_{best} and P_{best} updating, pheromone updating, and pheromone disturbance. Suppose there are m ants starting from the depot. Each ant selects customers to construct a solution route by applying the state transition rules of ACO. When all ants finish solution construction, the top r best ants perform local search to improve their solutions. Then, ants update their short-term memory: G_{best} solution and individual P_{best} solutions. After that, PACO updates the pheromone trails according to the G_{best} solution and the P_{best} solutions of elite ants. When the G_{best} is not improved over w consecutive iterations, PACO performs pheromone disturbance to modify current pheromone trails. PACO also resets the P_{best} solutions of some ants in order to prevent them from laying pheromone on the same routes again. The flowchart of the PACO algorithm is shown in Figure 1 and described briefly as follows.

Step 1 (Initialization). Initialize all parameters.

Step 2 (Solution construction). Let m ants construct solution routes.

Step 3 (Local search). The top r best ants perform local search.

Step 4. Update the G_{best} and P_{best} solutions of ants and select r elite ants.

Step 5. If G_{best} is not improved within w successive iterations, go to Step 6; otherwise, go to Step 7.

Step 6 (Pheromone disturbance). Randomly disturb the pheromone matrix, reset the P_{best} solutions for some ants, and go to Step 8.

Step 7 (Pheromone updating). Update the pheromone matrix based on the P_{best} solutions of elite ants.

Step 8. If iteration number reaches the maximum number of iterations (MaxIte), go to Step 9; otherwise, go to Step 2 for the next iteration.

Step 9. Output G_{best} , the best solution ever found.

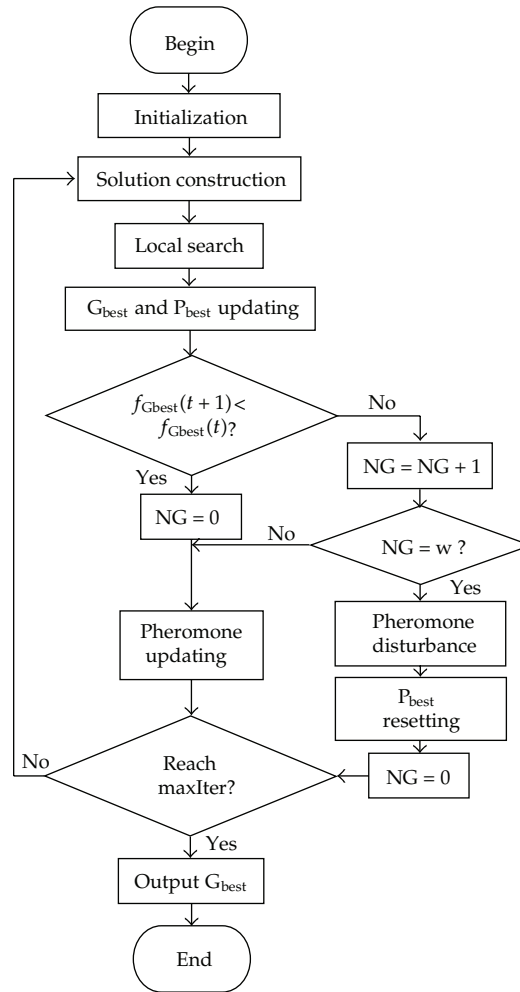


Figure 1: Flowchart of PACO.

3.4. Solution Construction

PACO can cluster customers into K vehicles and arrange vehicle visiting sequences at the same time. At each of the iterations, m ants construct individual solutions independently. Each solution contains K vehicle routes. Each ant starts from the depot, selects customers for the first vehicle, moves back to the depot before the capacity or distance limit is violated, then restarts from the depot and selects customers for the second vehicle. The procedure is repeated until all customers are selected and their sequences are arranged in used vehicles. It is possible that the last vehicle will serve all of the remaining customers even if it violates the capacity or distance limit. The procedure of ant solution construction ensures that the constraints defined in (2.7) and (2.8) can be satisfied.

The customer selection follows the state transition rule defined in (3.2). Suppose ant s is moving from customer i to the next customer, v :

$$v = \begin{cases} \arg \max_{j \in U_s} [(\tau_{ij})^\alpha (\eta_{ij})^\beta] & q \leq q_0, \\ V & q > q_0, \end{cases} \quad (3.2)$$

$$V: P_{ij} = \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{j \in U_s} (\tau_{ij})^\alpha (\eta_{ij})^\beta}, \quad (3.3)$$

where U_s is the set of customers that remain to be selected by ant s positioned on customer i , τ_{ij} is the pheromone trail on edge (i, j) , η_{ij} is the inverse of the distance of edge (i, j) , α and β are parameters which determine the relative importance of pheromone versus distance, q is a random number uniformly distributed in the interval of 0 and 1, q_0 is a parameter ranged between 0 and 1, and P_{ij} is the probability that ant s moves from customer i to customer j . If $q \leq q_0$, then ant s uses the greedy method (3.2) to select the next customer; otherwise, it uses the probabilistic rule (3.3) to determine the next customer. If vehicle k is full or reaches its distance limit, ant s has to go back to the depot and restarts from the depot to load the next vehicle. However, if vehicle k is the last vehicle of ant s , then it has to serve all of the remaining customers, even if the capacity or distance limit is violated.

3.5. Local Search

After all ants complete route construction, only the first r iteration-best ants can perform local search to improve their current solutions. PACO performs three types of neighborhood search: sequence inversion, insertion, and swap. These methods are often used in CVRP papers to improve iteration solutions, as is the case with [7, 8, 15]. Sequence inversion selects a vehicle at random from the current solution string, chooses two customers randomly, and then inverts the substring between these two customers. The swap operation selects two customers at random and then swaps these two customers in their positions. The insertion operation selects a customer at random and then inserts the customer in a random position. The swap operation may select two customers in different vehicles, and the insertion operation may assign the selected customer to a different vehicle. For such a case, the capacity and distance limits of the vehicles have to be checked. If the constraints are violated, the new solution becomes invalid and another one should be generated.

The whole procedure of local search for an ant solution is controlled by using a simulated annealing approach. SA performs neighborhood search R times at each temperature $T(t)$. Each time one of the three local search methods is randomly selected with equal probability to implement neighborhood search. A worse solution may have a chance to be accepted according to the following equations:

$$\begin{aligned} \Delta &= f(S') - f(S), \\ P(S') &= \exp\left(-\frac{\Delta}{T(t)}\right), \end{aligned} \quad (3.4)$$

where S is the current solution, S' is the new solution, $f(S)$ is the objective function value of S , $f(S')$ is the new objective function value, t is the current temperature, and $P(S')$ is the probability that SA accepts new solution S' . After carrying out neighborhood search R times, SA reduces the temperature. New temperature ($T(t+1)$) is equal to $\lambda \times T(t)$, where $0 < \lambda < 1$. PACO adopts a best improvement strategy in its local search step. That is, the best-so-far solution will be recorded during the run of the SA algorithm. When the termination condition is met, the ant solution will be replaced with the best-so-far solution of SA if the latter is really better.

After solution construction and local search, all ants compare their P_{best} solutions with their iteration solutions and perform P_{best} replacement if the iteration solutions are better. Then, ants are ranked in terms of the goodness of their P_{best} solutions, and the first r ants are the elite ants. Of course, the G_{best} solution of all ants is equal to the P_{best} solution of the best elite ant.

3.6. Pheromone Updating

Pheromone trails play the role of long-term memory in the PACO algorithm. Global updating is used to enhance the search in the neighborhood of better solutions. The paths of better solutions have higher levels of pheromone so as to attract more ants for later search. The pheromone on other paths evaporates over time and becomes less attractive to ants. Therefore, convergence of PACO can be accelerated by intensifying pheromone on the paths of elite solutions. The pheromone updating rule is defined as follows:

$$\begin{aligned}\tau_{ij} &= (1 - \rho)\tau_{ij} + \sum_{s=2}^r \Delta\tau_{ij}^{P_{\text{best}_s}} + \Delta\tau_{ij}^{G_{\text{best}}}, \\ \Delta\tau_{ij}^{P_{\text{best}_s}} &= \frac{1}{f_{P_{\text{best}}}^s}, \\ \Delta\tau_{ij}^{G_{\text{best}}} &= \frac{1}{f_{G_{\text{best}}}},\end{aligned}\tag{3.5}$$

where ρ is the pheromone evaporation rate and ranges between 0 and 1, r is the total number of elite ants, $\Delta\tau_{ij}^{P_{\text{best}_s}}$ and $\Delta\tau_{ij}^{G_{\text{best}}}$, are the pheromone added by elite ant s and the best ant, respectively, and $f_{P_{\text{best}}}^s$ and $f_{G_{\text{best}}}$ are the objective function values of elite ant s and the best ant, respectively.

3.7. Pheromone Disturbance

Since only elite ants lay pheromone on the routes of their P_{best} solutions, the pheromone on the paths of elite solutions will accumulate very fast. It leads to the state of pheromone stagnation, and the search may be trapped by local optima. To overcome this problem, PACO adopts pheromone disturbance to escape from local optima and to explore different areas of the search space. Pheromone disturbance is performed when the G_{best} solution of ants is not updated (improved) up to w successive iterations. w is called the disturbance period. Large w makes PACO easy to raise the chances of settling for a false optimum, while small w retards

Table 1: Original pheromone matrix.

i	j				
	1	2	3	4	5
1	0	0.1	0.8	0.2	0.5
2		0	0.2	0.1	0.2
3			0	0.3	0.5
4				0	0.8
5					0

the search convergence and takes more computational time. It is suggested that w be equal to the number of customers.

The basic idea of pheromone disturbance is similar to arithmetic crossover in GA, but it is used to produce new pheromone trails, rather than to generate new solutions. Pheromone disturbance has three steps. The first step is to select edges for disturbance. The disturbance rate (μ) determines the probability that an edge is selected for disturbance. If μ is set too high, it will be difficult to retain previous search experiences; on the other hand, if it is set too low, the effect of disturbance will not be evident. The second step is to cluster selected edges into groups, each of which has two edges with the same customer node. The groups with single edges will be abandoned. The third step is to use a random number $q \in [0, 1]$ to determine a disturbance type for each of the paired edges. There are three types of pheromone disturbance: unchanging, replacement, and weighted average. As defined in (3.6), paired edges (i, j) and (i, u) have three possible results. That is, the pheromone on edge (i, j) may remain the same, be replaced by the pheromone level of edge (i, u) , or be replaced with the weighted average of the pheromone levels of these two edges.

$$\tau_{ij}^{t+1} = \begin{cases} \tau_{ij}^t & q < 0.2, \\ \tau_{iu}^t & 0.2 \leq q < 0.4, \\ \delta\tau_{ij}^t + (1 - \delta)\tau_{iu}^t, & j \neq u, q \geq 0.4, \end{cases} \quad (3.6)$$

where δ is a uniform random number in the range $[0, 1]$ and determines the ratios of pheromone on the two edges.

After pheromone disturbance, ant s positioned on customer node i has a chance to explore different edges rather than to keep selecting the same next customer node to move to. Hence, pheromone disturbance increases the probability of finding optimal solutions. Note that we deal with symmetric CVRPs in this paper, where the distances between customer nodes are independent of the direction of traversing the edges. The same situation applies to pheromone trails. Accordingly, $d_{ij} = d_{ji}$ and $\tau_{ij} = \tau_{ji}$ for each pair of nodes.

We use an example to illustrate the idea of pheromone disturbance. Suppose we have five customers, then there are 20 edges with pheromone deposits. The original pheromone matrix is shown in Table 1. Since $\tau_{ij} = \tau_{ji}$ for each pair of nodes, the algorithm considers only half the edges to be disturbed.

The first step is to select edges for disturbance. Each edge is selected with a probability of $\mu = 0.3$. For each edge, we generate a random number from uniform distribution in the range between 0 and 1. The generated random numbers are shown in Table 2. A edge is selected if its random number is less than μ . Table 2 tells us that four edges are selected and marked in bold type. The second step is to pair two edges that have the same customer node.

Table 2: Edges selected for disturbance.

i	j				
	1	2	3	4	5
1	0	0.5	0.2	0.6	0.1
2		0	0.1	0.2	0.7
3			0	0.8	0.4
4				0	0.5
5					0

Table 3: Disturbance results.

Edge	$\tau_{ij}(t)$	q	Option	$\tau_{ij}(t+1)$
(1-3)	0.8	0.1	1	0.8
(1-5)	0.5	0.8	3	0.65
(2-3)	0.2	0.3	2	0.1
(2-4)	0.1	0.5	3	0.11

Table 4: New pheromone matrix.

i	j				
	1	2	3	4	5
1	0	0.1	0.8	0.2	0.65
2		0	0.1	0.11	0.2
3			0	0.3	0.5
4				0	0.8
5					0

Scanning the matrix in Table 2 row by row, we obtain the pairing results: {edge (1,3), edge (1,5)} and {edge (2,3), edge (2,4)}. The third step uses (3.6) to determine new pheromone trails for paired edges. Random number q is first generated to determine a disturbance type for each selected edge. The disturbance results are shown in Table 3. For example, for the first paired edges, option 1 (unchanging) applies to edge (1, 3) and option 3 (weighted average) applies to edge (1, 5). Note that here δ is a random number in the range of $[0, 1]$. In our example of edge (1, 5), δ is equal to 0.5. The new pheromone matrix is presented in Table 4.

3.8. P_{best} Solution Resetting

Since pheromone updating is based on the P_{best} solutions of elite ants, it may diminish the effect of pheromone disturbance very fast in the following iterations. To avoid that situation, PACO resets the P_{best} solutions of some better ants. That is, the P_{best} solutions of the ants are removed from their memory and find new ones in the next iteration. Note that not all of the ants need to reset their P_{best} solutions. The resetting is determined by the difference in objective function value between P_{best} and G_{best} solutions. For ant s , if the difference is less than or equal to a threshold, Δf , (i.e., $\Delta f \leq f_{P_{best}}^s - f_{G_{best}}$), then ant s has to reset its P_{best} solution.

4. Computational Results

The PACO algorithm described in Section 3 was coded in Java, and all experiments were performed on a personal computer with Intel Core 2 CPU T7500 running at 2.20 GHz. Two sets of benchmark problems were selected to evaluate the effectiveness of our proposed algorithm for the capacitated vehicle routing problem.

The first set has 16 test problems and can be downloaded from the website <http://www.branchandcut.org/VRP/data/>. The problems in this benchmark set are subject to capacity constraints only. The total number of customers varies from 29 to 134, and the total number of vehicles ranges from 3 to 10. The locations of customers appear in clusters in the problems with their names initiated with B and M, while in the remaining problems customers are randomly scattered or semiclustered. The first benchmark set was used by Chen et al. [18] and Ai and Kachitvichyanukul [20] to test their PSO-based algorithms.

The second benchmark set can be downloaded from the website <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/vrpinfo.html>. It has been widely used in previous studies and contains 14 classical test problems selected from Christofides et al. [23]. Problems 1, 2, 3, 4, 5, 11, and 12 consider the constraint of capacity only, while the remaining problems are subject to the capacity and distance limits. The total number of customers varies from 50 to 199, and the total number of vehicles ranges from 5 to 18. Besides, customers are randomly distributed in the first ten problems whereas customers are clustered in the last four problems.

The PACO parameters set as follows were found to be robust for most of the test problems according to our pilot tests. PACO parameters are maximum iteration number $\text{MaxIte} = 1000$, population size $\text{pop} = N/2$, number of elite ants $r = 3$, penalty coefficient $p = 100$, $q_0 = 0.8$, $\rho = 0.5$, $\alpha = 2$, $\beta = 1$, and $\tau_0 = 1/(N \times L_{mn})$, where L_{mn} is the tour length found by the nearest neighbor heuristic. Local search parameters are initial temperature $t_0 = 2$, final temperature $t_f = 0.01$, $R = \max\{N \times K/2, 250\}$, $\lambda = 0.9$. Pheromone disturbance parameters are: $\mu = 0.3$, $w = N$, $\Delta f = 5$.

Table 5 lists the computational results of PACO on two sets of test problems. The best solution, average solution, worst solution, and standard deviation (Std.) computed over 20 independent runs on each problem are summarized, along with their average computational time (in seconds) required to reach the final best solutions. The best solutions equal to the best-known solutions of benchmark problems are asterisked and typed in bold. Table 5 reveals that PACO is able to generate reasonable good solutions for most of CVRPs in terms of solution quality. Twelve out of sixteen test problems can be solved successfully by the proposed algorithm in the first benchmark set. For the second set, seven out of fourteen test problems can be solved successfully by PACO.

To evaluate the pheromone disturbance strategy and P_{best} -resetting operation, the proposed algorithm is compared with standard ACO [9] and PS-ACO [22] in terms of their convergence trends. Standard ACO and PS-ACO were originally proposed for solving traveling salesman problems (TSP), not CVRPs. We implemented these two algorithms in Java and tried to apply them to solve CVRPs with the same local search method and parameter settings used in PACO.

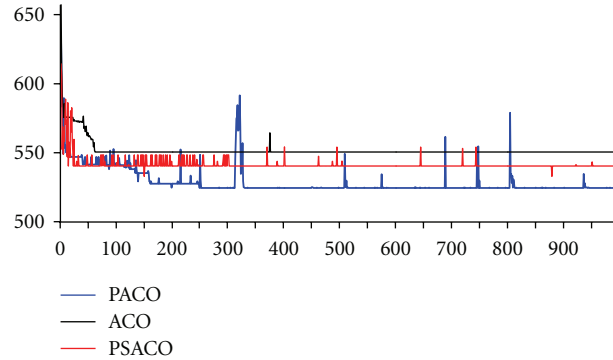
Benchmark problem C1 was selected to test three ACO-based algorithms. Three data sets of iteration-best solutions are plotted in Figure 2, The curves reveal that both PACO and PS-ACO can converge faster than standard ACO but only PACO can find the optimal solution. It demonstrates that, with pheromone disturbance and P_{best} solution resetting, PACO can effectively escape from local optima and find better solutions. In Figure 2, a

Table 5: Computational results of PACO over 20 runs on two benchmark sets.

	First benchmark set					Second benchmark set					
	Best	Avg.	Worst	Std.	AT (s)		Best	Avg.	Worst	Std.	AT (s)
A-n33-k5	661*	661	661	0	0.87	C1	524.61*	527.54	543.16	4.98	32.3
A-n46-k7	914*	914	914	0	6.02	C2	835.26*	842.71	852.14	4.31	107.91
A-n60-k9	1354*	1356.4	1369	4.49	52.88	C3	829.92	838.29	844.65	3.76	141.89
B-n35-k5	955*	955	955	0	2.65	C4	1040.23	1053.22	1077.68	9.31	377.83
B-n45-k5	751*	751	751	0	5.85	C5	1348.73	1375.05	1392.93	12.73	1048.45
B-n68-k9	1275	1286.25	1288	2.86	62.97	C6	555.43*	556.72	560.24	1.09	27.09
B-n78-k10	1221*	1228.1	1252	8.6	98.78	C7	909.68*	917.93	932.06	7	98.70
E-n30-k3	534*	534	534	0	4.38	C8	868.61	880.47	895.7	8.37	117.74
E-n51-k5	521*	522.65	528	2.87	19.46	C9	1171.94	1194.96	1233.9	15.24	505.89
E-n76-k7	685	691.15	694	2.35	46.85	C10	1454.81	1498.23	1577.52	28.8	939.08
F-n72-k4	237*	237	237	0	30.64	C11	1042.11*	1045.01	1049.45	2.87	196.49
F-n135-k7	1170	1193.45	1229	15.99	248.77	C12	819.56*	821.55	825.95	1.91	148.67
M-n101-k10	820*	822.9	824	1.41	113.28	C13	1562.64	1575.55	1596.63	8.16	320.92
M-n121-k7	1034*	1039.6	1127	20.12	80.62	C14	866.37*	866.81	867.77	0.51	173.15
P-n76-K4	593*	597.95	616	5.72	53.48						
P-n101-k4	683	693.35	706	7.21	64.92						

*The solution equals the best-known solution.

AT denotes the average CPU time required to reach the final best solution over 20 runs.

**Figure 2:** Convergence trends of three ACO-based algorithms tested on problem C1.

couple of peaks on the PACO curve indicate the effects of pheromone disturbance and P_{best} -resetting. The disadvantage of PS-ACO is that all of the local best solutions are considered in the pheromone updating procedure. When most ants have similar P_{best} solutions, the total amount of pheromone increment on edges will become very large. It results in the state of pheromone stagnation and the search is trapped by a local optimum.

We conducted a comparative study to compare PACO with a couple of swarm intelligence methods available for the CVRP. The comparative study contains two parts. In the first part, the PACO algorithm is compared with three different PSO-based algorithms, which were all tested on the 16 problems from the first benchmark set. The smaller the objective function value, the better the solution. Table 6 displays the computational results of PACO and the best results found in the papers of Chen et al. [18], Ai and Kachitvichyanukul [20], and Kao and Chen [19], denoted as DPSO-SA, SR-2, CPSO-SA, respectively. Experimental

Table 6: Comparison of PACO with three PSO-based algorithms.

No.	Problem	N	K	BKS	DPSO-SA [18]	SR-2 [20]	CPSO-SA [19]	PACO
1	A-n33-k5	32	5	661	661* /32.2	661* /13	661* /0.7	661* /0.12
2	A-n46-k7	45	7	914	914* /128.9	914* /23	917/2.4	914* /0.16
3	A-n60-k9	59	9	1354	1354* /308.8	1355/40	1354* /6.5	1354* /14.15
4	B-n35-k5	34	5	955	955* /37.6	955* /14	955* /1.2	955* /0.06
5	B-n45-k5	44	5	751	751* /134.2	751* /20	751* /4.8	751* /1.12
6	B-n68-k9	67	9	1272	1272* /344.2	1274/50	1274/27.2	1275/87.19
7	B-n78-k10	77	10	1221	1239/429.4	1223/64	1237/24	1221* /54.38
8	E-n30-k3	29	3	534	534* /28.4	534* /16	534* /0.3	534* /0.05
9	E-n51-k5	50	5	521	528/300.5	521* /22	521* /4.6	521* /0.51
10	E-n76-k7	75	7	682	688/526.5	682* /60	692/9.5	685/18.95
11	F-n72-k4	71	4	237	244/398.3	237* /53	237* /5.3	237* /6.28
12	F-n135-k7	134	7	1162	1215/1526.3	1162* /258	1200/202.8	1170/246.85
13	M-n101-k10	100	10	820	824/874.2	820* /114	825/6.1	820* /66.02
14	M-n121-k7	120	7	1034	1038/1733.5	1036/89	1039/51.5	1034* /8.18
15	P-n76-K4	75	4	593	602/496.3	594/48	596/27.6	593* /8.61
16	P-n101-k4	100	4	681	694/977.5	683/86	691/29.4	683/25.7

Notes: $x/y = f(\text{best solution})/\text{shortest CPU Time (s)}$, BKS is the best-known solution provided by published papers, DPSO-SA used Intel Pentium IV CPU 1.8GHz with 256 M RAM, SR-2 used Intel Pentium IV CPU 3.4GHz with 1 GB RAM, and CPSO-SA used Intel Core 2 CPU E8400 3 GHz with 3.5 G RAM.

Table 7: Comparison of PACO with three ACO-based algorithms.

Prob.	N	K	BKS	IACS [12]	SS.ACO [13]	EACO [15]	PACO
C1	50	5	524.61	524.61* /3	524.61* /55.23	524.61* /—	524.61* /0.79
C2	75	10	835.26	836.18/26	835.26* /70.43	835.26* /—	835.26* /163.44
C3	100	8	826.14	835.6/101	830.14/120.25	826.14* /—	829.92/183.08
C4	150	12	1028.42	1038.22/617	1038.20/250.76	1041.83/—	1040.23/547.88
C5	199	17	1291.45	1327.07/3080	1307.18/707.80	1338.48/—	1348.73/1455.64
C6	50	6	555.43	555.43* /5	559.12/65.17	555.43* /—	555.43* /7.72
C7	75	11	909.68	909.68* /41	912.68/90.42	909.68* /—	909.68* /66.68
C8	100	9	865.94	865.94* /115	869.34/210.46	865.94* /—	868.61/146.5
C9	150	14	1162.55	1173.76/853	1179.4/520.52	1168.81/—	1171.94/614.53
C10	199	18	1395.85	1413.83/4223	1410.26/1012.23	1413.69/—	1454.81/1114.21
C11	120	7	1042.11	1042.11* /204	1044.12/232.46	1045.5/—	1042.11* /95.33
C12	100	10	819.56	832.67/88	824.31/156.47	819.56* /—	819.56* /122.87
C13	120	11	1541.14	1547.07/428	1556.52/467.24	1554.93/—	1562.64/156.32
C14	100	11	866.37	866.37* /125	870.26/368.72	866.37* /—	866.37* /32.75

Notes: $x/y = f(\text{best solution})/\text{shortest CPU Time (s)}$, BKS is the best-known solution provided by published papers, — denotes that the data is not available, IACS used Intel Pentium III CPU 1000 MHz with 128 MB RAM, SS.ACO used IBM computer CPU 1600 MHz with 512 MB RAM, and EACO used Pentium IV CPU 3.0 GHz.

results show that PACO is able to obtain the same or better results compared with three PSO-based algorithms.

In the second part of the comparative study, the computational results of PACO are compared with three ACO-based algorithms, which were all tested on the 14 problems from the second benchmark set. Table 7 reports the best results obtained by these various ACO algorithms. The best results of IACS, SS.ACO, and EACO are found in [12, 13, 15],

respectively. Table 7 indicates that PACO generates very good solutions for seven problems, which are equal to the best solutions published so far. For three compared algorithms, only EACO is better than PACO because it can find best-known solutions for eight problems. It can be noticed that both PACO and EACO can obtain optimum solutions for six problems in common. For problem C11, PACO can reach the best-known solution but EACO cannot, while, for problems C3 and C8, EACO can outperform PACO. However, the best solutions produced by PACO and EACO are nearly equal for these three problems.

Tables 6 and 7 also list the shortest computational times required to reach the final best solution over 20 independent runs. From the data, it can be observed that the computation time taken by PACO is reasonable in practice in comparison with existing PSO- and ACO-based algorithms. It shows that pheromone disturbance can improve solutions but does not increase much computational time. Note that the computation time has not been reported in [15]. Instead, the authors of [15] used the maximum execution time as the termination condition. They stopped the EACO algorithm after one hour of running for 14 benchmark problems. The overall result of comparative study shows that the proposed algorithm is competitive with recent swarm intelligence approaches in terms of solution quality and CPU time.

5. Conclusion

This paper proposes a hybrid algorithm, PACO, which takes advantage of ant colony optimization and particle swarm optimization for capacitated vehicle problems. During the searching process, artificial ants construct solution routes, memorize the best solution ever found, and lay pheromone on the routes of swarm and personal best solutions. To prevent being trapped in local optima and to increase the probability of obtaining better solutions, PACO performs pheromone disturbance and short-term memory resetting operations to adjust stagnated pheromone trails. Disturbed pheromone trails guide ants to find new P_{best} and G_{best} solutions. The merits of PSO adopted in PACO can speed convergence during a run, even after pheromone disturbance operations. Computational results show that the performance of PACO is competitive in terms of solution quality when compared with existing ACO- and PSO-based approaches. For future research, PACO can be modified to extend its application to vehicle routing problems with time windows or multiple depots, among others.

Acknowledgments

The authors would like to thank the referees for their valuable comments and suggestions that have greatly improved the quality of this paper. The authors are also grateful to National Science Council, Taiwan (Grant no. NSC 99-2410-H-036-003-MY2) for the financial support.

References

- [1] G. Laporte, "The vehicle routing problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.
- [2] 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.

- [3] J. F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo, "Vehicle routing," in *Handbook in Operations Research and Management Science*, C. Barnhart and G. Laporte, Eds., vol. 14, pp. 367–428, 2007.
- [4] J. K. Lenstra and A. H. G. Rinnooy Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–228, 1981.
- [5] G. Barbarosoglu and D. Ozgur, "A tabu search algorithm for the vehicle routing problem," *Computers and Operations Research*, vol. 26, no. 3, pp. 255–270, 1999.
- [6] B. M. Baker and M. A. Ayechev, "A genetic algorithm for the vehicle routing problem," *Computers and Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.
- [7] S. W. Lin, Z. J. Lee, K. C. Ying, and C. Y. Lee, "Applying hybrid meta-heuristics for capacitated vehicle routing problem," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1505–1512, 2009.
- [8] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126–135, 2011.
- [9] 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.
- [10] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.
- [11] Z. Liu and Y. Cai, "Sweep based multiple ant colonies algorithm for capacitated vehicle routing problem," in *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE '05)*, pp. 387–394, October 2005.
- [12] C. H. Chen and C. J. 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.
- [13] X. Zhang and L. Tang, "A new hybrid ant colony optimization algorithm for the vehicle routing problem," *Pattern Recognition Letters*, vol. 30, no. 9, pp. 848–855, 2009.
- [14] B. Yu, Z. 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.
- [15] C. Y. Lee, Z. J. Lee, S. W. Lin, and K. C. Ying, "An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem," *Applied Intelligence*, vol. 32, no. 1, pp. 88–95, 2010.
- [16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.
- [17] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'98)*, pp. 69–73, May 1998.
- [18] A. L. Chen, G. K. Yang, and Z. M. Wu, "Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem," *Journal of Zhejiang University: Science*, vol. 7, no. 4, pp. 607–614, 2006.
- [19] Y. Kao and M. Chen, "A hybrid PSO algorithm for the CVRP problem," in *Proceedings of the International Conference on Evolutionary Computation Theory and Applications (ECTA '11)*, pp. 539–543, Paris, France, October 2011.
- [20] T. J. Ai and V. Kachitvichyanukul, "Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem," *Computers and Industrial Engineering*, vol. 56, no. 1, pp. 380–387, 2009.
- [21] Y. Marinakis, M. Marinaki, and G. Dounias, "A hybrid particle swarm optimization algorithm for the vehicle routing problem," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 463–472, 2010.
- [22] B. Shuang, J. Chen, and Z. Li, "Study on hybrid PS-ACO algorithm," *Applied Intelligence*, vol. 34, no. 1, pp. 64–73, 2011.
- [23] 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.

