Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research xxx (2008) xxx–xxx
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

Multi-ant colony system (MACS) for a vehicle routing problem with backhauls

Yuvraj Gajpal *, P.L. Abad

Michael G DeGroote School of Business, McMaster University, Hamilton, Ontario, Canada L8S4M4

Received 17 July 2006; accepted 20 February 2008

Abstract

The vehicle routing problem with backhaul (VRPB) is an extension of the capacitated vehicle routing problem (CVRP). In VRPB, there are linehaul as well as backhaul customers. The number of vehicles is considered to be fixed and deliveries for linehaul customers must be made before any pickups from backhaul customers. The objective is to design routes for the vehicles so that the total distance traveled is minimized. We use multi-ant colony system (MACS) to solve VRPB which is a combinatorial optimization problem. Ant colony system (ACS) is an algorithmic approach inspired by foraging behavior of real ants. Artificial ants are used to construct a solution by using pheromone information from previously generated solutions. The proposed MACS algorithm uses a new construction rule as well as two multi-route local search schemes. An extensive numerical experiment is performed on benchmark problems available in the literature.

Keywords: Metaheuristic; Ant colony; Vehicle routing

1. Introduction

In vehicle routing problem with backhaul (VRPB), there are two groups of customers: linehaul or delivery customers and backhaul or pickup customers. A linehaul customer requires a given quantity of product to be delivered while a backhaul customer requires a given quantity of product to be picked up. The problem can be defined using graph theory as follows. Consider a complete undirected graph $G = (N, A)$, where $N = \{0\} \cup L \cup B$ is a set of $l + b + 1$ vertices and subsets $L = \{1, \ldots, l\}$ and $B = \{l + 1, l + 2, \ldots, l + b\}$ correspond to the linehaul and backhaul customers, respectively. Set $A = \{(i, j), i, j \in N\}$ is a set of arcs. Associated with arc (i, j), there is a nonnegative cost $c(i, j)$ and associated with vertex i there is a non-negative quantity d_i. Vertex 0 represents the depot (with fictitious demand 0), where v identical vehicles, each with capacity Q are located. It is assumed that the number of available vehicles $v \geqslant \max(v_L, v_B)$, where v_L and v_B are the minimum number of vehicles needed to serve all linehaul and all backhaul customers, separately. The values of v_L and v_B can be calculated by solving the bin packing problems (BPP) associated with the subsets of linehaul and backhaul customers. Although BPP is an NP-hard problem in strong sense, it can be solved optimally within reasonable CPU time even when the problem involves hundreds of items/customers (see Silvano and Paolo, 1990).

In this paper, we do not deal with mixed service, i.e., we assume that each vehicle is unloaded first by satisfying the demands of linehaul customers and later it is loaded by collecting the loads from backhaul customers. Also a route consisting entirely of backhaul customers is not allowed (Toth and Vigo, 2001). The objective is to design a set of minimum

* Corresponding author. Tel.: +1 905 525 9140; fax: +1 905 525 8995.
E-mail address: gajpaly@mcmaster.ca (Y. Gajpal).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.02.025

cost routes so that the total load of linehaul or the total load of backhaul customers does not exceed the vehicle capacity on each route. VRPB is an NP-hard problem in a strong sense since by setting $B = \phi$, the problem becomes CVRP, which is known to be an NP-hard problem. In this paper, we use the abbreviations CVRP and VRP interchangeably.

Many exact and heuristic methods are proposed in the literature to solve VRPB with the objective of minimizing the total distance traveled by the vehicles. Goetschalckx and Jacobsblecha (1989) propose a two phase approach for VRPB. In the first phase, an initial solution is generated based upon space filling curves for linehaul and backhaul customers. In the second phase, these solutions are merged to get a final set of vehicle routes. This algorithm is not able to handle the case where the number of available vehicles is fixed. Goetschalckx and Jacobsblecha (1993) extend the generalized assignment heuristic of VRP to VRPB. Toth and Vigo (1996) developed a "cluster first and route second" algorithm based upon the generalized assignment approach of VRP. Further, Toth and Vigo (1999) proposed a new cluster method which exploits the information associated with the lower bound based on Lagrangian relaxation. This heuristic is used for both the symmetric as well as asymmetric versions of the problem. Apart from the above, few studies have investigated mixed service and other variations of VRPB (e.g., Thangiah et al., 1996; Anily, 1996; Wade and Salhi, 2002).

Yano et al. (1987) developed an exact procedure for VRPB using the set covering approach based upon Lagrangian relaxation and branch and bound framework. They applied the procedure to the routing problem for quality stores where the number of customers served was small. Toth and Vigo (1997) and Mingozzi et al. (1999) used a branch and bound algorithm to solve the problem exactly for up to 100 customers. Toth and Vigo (1997) used Lagrangian relaxation of the underlying integer programming formulation to obtain a lower bound whereas Mingozzi et al. (1999) used a heuristics to obtain a lower bound in their branch and bound algorithm.

The exact methods such as branch and bound algorithm have limitations in solving a large size problem whereas simple heuristic approaches have limitations in terms of the solution quality. A metaheuristic attempts to improve the simple heuristics by relaxing the computing time restriction. A metaheuristic may take longer CPU time but it would produce a good quality solution. Many attempts have been made to solve CVRP using metaheuristic approaches (e.g., Gendreau et al., 1994; Osman, 1993; Rochat and Taillard, 1995; Rego and Roucairol, 1996). Recently, few attempts have been made to solve VRPB using Tabu search. Osman and Wassan (2002) proposed reactive tabu search (RTS) for VRPB. Further Wassan (2004) combined adaptive memory programming with tabu search and proposed reactive tabu adaptive memory programming search (RTS-AMP). Both tabu search heuristics use λ-interchange of customers as proposed by Osman (1993). RTS-AMP maintains a set of solutions called elitist solutions and uses these solutions to turn the direction of search towards the unexplored region of the space. Recently, Brandao (2006) has proposed a new tabu search algorithm where the initial solution is obtained from a pseudo lower bound based upon the *K-tree* approach. Other attempts of metaheuristics are based on different variants of VRPB (e.g. Potvin et al., 1996; Thangiah et al., 1996; Duhamel et al., 1997; Reimann et al., 2002; Wade and Salhi, 2004).

In recent times, attempts have also been made to solve CVRP using ant colony system (ACS) metaheuristic. For an introduction to the ACS approach, see Dorigo et al. (1996). Bullnheimer et al. (1999) designed for the first time an ant colony system for the vehicle routing problem. This ant colony system is further improved by Doerner et al. (2002) by combining it with the problem based savings heuristic. Recently, Reimann et al. (2004) proposed a decomposition based approach (D-Ants) for CVRP. In the D-Ants system, the problem is decomposed into several small sub-problems and each sub-problem is solved using the saving based ant colony system. Reimann et al. (2002) designed insertion based ant colony system for the vehicle routing problem with backhauls and time windows. Wade and Salhi (2004) designed an ant colony algorithm for the vehicle routing problem with backhauls and mixed load. Although there are several studies on using tabu search in VRPB, to our knowledge, there have been no published studies on using the ant colony system in VRPB in which all linehaul customers are to be served before any backhaul customers.

This paper is organized as follows. Section 2 describes the proposed multi-ant colony system (MACS). We describe two multi-route local search schemes used in MACS. An extensive computational experiment with benchmark problems available in the literature is presented in Section 3. We compare the performance of different algorithms with MACS. An interesting property of MACS is the control over solution quality and CPU time, which is discussed in Section 4. Conclusions follow in Section 5.

2. Multi-ant colony system (MACS) for VRPB

The existing ant colony system for CVRP (Bullnheimer et al., 1999; Doerner et al., 2002) is not suitable when the number of vehicles is fixed. It can generate an infeasible solution in terms of the number of vehicles required. We use multi-ant colony system (MACS) to deal with the infeasibility. The concept of multiple types of ants was first proposed by Gambardella et al. (1999) who designed an ant colony system to solve the vehicle routing problem with time windows. They used two types of ants to minimize the two different objective functions: the first type of ant minimizes the number of vehicles used while the second type of ant minimizes the total tour length. Our multi-ant approach is inspired by "cluster first and route second" procedure of VRP. We also use two types of ants to construct a solution. The first type of ant is used to assign

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

customers to vehicles and the second type of ant is used to construct a route for a vehicle given the assigned customers; i.e., to solve the underlying traveling salesman problem. An outline of the algorithm is given below and a more detailed description of the algorithm is provided in the next sub-section.

Step1: Initialize the trail intensities and parameters by generating an initial solution. Set COUNT = 1.
Step 2: While COUNT $\leqslant$ 5 do the following:

- Generate an ant-solution for each ant using the trail intensities.
- Improve each ant-solution by carrying out the following local search:
  1. Use 2-opt local search once.
  2. Use the following two local search schemes until there is no further improvement:
     (i) Customer insertion/interchange multi-route scheme,
     (ii) Sub-path exchange multi-route scheme.
  3. Update elitist ants.
  4. If there is no change in the elitist ant solutions in the last 10 iterations,
     Then
     Reinitialize the trail intensity and destroy all elitist ant solutions. Increase COUNT by 1, i.e., set COUNT = COUNT + 1.
     Else
     Update trail intensities.

Step 3: Return the best solution found so far.

In MACS, we have added the feature of reinitialization of trail intensities and destroying the elitist ant solutions. This feature is similar to the re-start feature of Genetic Algorithm used by Prins (2004).

2.1. The initial solution and the reinitialization of trail intensities

The trail intensities are initialized using an initial solution, which is generated using a two-phase procedure. In the first phase, customers are randomly assigned one by one to v vehicles ignoring the capacity restriction. In the second phase, a route is constructed for each vehicle by iteratively adding the assigned customers. The initial solution plays an important role in many metaheuristic approaches such as simulated annealing (Johnson et al., 1989). However, it does not play a significant role in ant colony system. Hence, we avoid using a time-consuming procedure for constructing an initial feasible solution.

We use two types of ants: *vehicle-ants* and *route-ants*. The trail intensity of a *vehicle-ant* $\tau 1_{ik}$, represents the intensity of customer i being served by vehicle k. The trail intensity of a *route-ant* $\tau 2_{ij}$, is the intensity of visiting customer j immediately after i. Suppose L is the objective function value (i.e., the total tour length) in the initial solution. Initially, we set $\tau 1_{ik} = 1/L$; $i = 1, \ldots, n$; $k = 1, \ldots, v$. and $\tau 2_{ij} = 1/L$; $i = 1, \ldots, n$; $j = 1, \ldots, n$; $i \neq j$.

In *Step 2* of the algorithm, trail intensities are reinitialized five times. We use the objective function value of the best solution found to date to reinitialize the trail intensities. Thus, during re-initialization all $\tau 1_{ik}$ and $\tau 2_{ij}$ are set to $1/L_{best}$, where L_{best} is the objective function value (i.e., the total tour length) in the best solution found to date.

2.2. Generation of a solution by an ant

The construction process used in existing ACS (Bullnheimer et al., 1999; Doerner et al., 2002) for solving CVRP can produce an infeasible solution in terms of the number of vehicles used. In MACS, we keep the number of vehicles fixed. We use a two phase approach based upon *vehicle-ants* and *route-ants*. The approach is described in detail below.

2.2.1. Assigning customers to vehicles

In this phase, we assign customers to vehicles insuring that the capacity of each vehicle is not exceeded. Let

l	total number of linehaul customers
b	total number of backhaul customers
$n = l + b$	total number of customers which includes linehaul and backhaul customers
Ω	the set of customers not yet served
C_k	the set of customers to be served by vehicle k
v	the number of available vehicles
Q	capacity of each vehicle
AC_k^L	available capacity of vehicle k for assigning a linehaul customer
AC_k^B	available capacity of vehicle k for assigning a backhaul customer

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Note that as stated in Section 1, v is known and its value is such that feasible solutions exist. The procedure given below should return a feasible solution as long as such a solution exists. The steps for assigning customers to vehicles are as follows.

Step 1: Initialize the sets and the available capacities:

$$\Omega = \{1, 2, 3, \ldots, n\},$$
$$C_k = \phi; \quad k = 1, 2, \ldots, v,$$
$$AC_k^L = AC_k^B = Q; \quad k = 1, 2 \ldots., v.$$

Step 2: Randomly choose a customer, say customer i, who is not yet assigned to a vehicle. Remove customer i from set Ω.

Step 3: Let Ψ be the set of feasible vehicles. A vehicle is feasible for assigning customer i if available capacity AC_k^L is more than or equal to the demand for customer i and i is a linehaul customer, or if AC_k^B is more than the load for customer i and i is a backhaul customer. We use the following *random-proportional rule* for choosing the vehicle to serve customer i

$$p_{ik} = \begin{cases} \frac{[\tau 1_{ik}]^{\alpha 1}[\eta_k]^{\beta 1}}{\sum_{h \in \Psi}[\tau 1_{ih}]^{\alpha 1}[\eta_h]^{\beta 1}} & \text{if vehicle } k \in \Psi, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where

$$\eta_k = \begin{cases} \frac{AC_k^L}{Q} & \text{if customer } i \text{ is a linehaul customer,} \\ \frac{AC_k^B}{Q} & \text{if customer } i \text{ is a backhaul customer.} \end{cases} \tag{2}$$

Here, η_k is called dummy visibility since it insures that a high probability is given to a vehicle that has high available capacity. The dummy visibility increases the chance of generating a feasible solution since it enhances the probability of assigning at least one linehaul customer to each vehicle. Parameters $\alpha 1$ and $\beta 1$ reflect the relative influence of trail intensity and dummy visibility. AC_k^L or AC_k^B is updated once a vehicle is chosen for serving customer i. Customer i is then added to set C_k.

Step 4: if $\Omega = \phi$ then stop else go to *Step 2*.

There is a chance of getting an infeasible solution in *Step 3* because of the limited capacity of a vehicle. In particular, while assigning the last customer, we may find that no vehicle can accommodate the customer; i.e., $\Psi = \phi$. In this case, *Steps 1–4* are repeated wherein we set customers in Ω in non-increasing order of demand. We repeat cycles of *Steps 1–4* until a feasible solution is found.

Arranging the customers in non-increasing order of demand increases the likelihood of obtaining a feasible solution in a given cycle. Since customers are assigned to vehicles according to the random proportion rule, the solution would vary from cycle to cycle. Since, we are assuming that feasible solutions exist, repeated cycles of *Steps 1–4* would eventually lead to a feasible solution.

2.2.2. Construction of routes for individual vehicles

Once customers are assigned to vehicles, a route is constructed for each vehicle by solving the underlying traveling salesman problem. First, all linehaul customers assigned to vehicle k are considered for visit. When all linehaul customers are visited, the backhaul customers assigned to vehicle k are considered. The ant starts from the depot and successively builds the solution by choosing the next customer j probabilistically from the set of feasible customers, *Re*. Initially, *Re* is set to linehaul customers assigned to vehicle k and it is updated as linehaul customers are visited. When *Re* becomes empty, it is set to backhaul customers assigned to vehicle k. The attractiveness value of visiting customer j immediately after customer i, ξ_{ij} is defined as

$$\xi_{ij} = \begin{cases} [\tau 2_{ij}]^{\alpha 2}[s(i,j)]^{\beta 2} & \text{if } i \text{ is a customer,} \\ [\tau 2_{ij}]^{\alpha 2}[c(i,j)]^{\beta 2} & \text{if } i \text{ is a depot.} \end{cases} \tag{3}$$

Here, parameter $\alpha 2$ represents the relative influence of the trail intensity and parameter $\beta 2$ represents the relative influence of the saving value. The saving value $s(i,j)$ represents the savings (in terms of distance traveled) when customers i and

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

j are served jointly by one vehicle instead of two separate vehicles. Here, $c(i,j)$ is the distance between customer i and j. We use the following *random proportional rule* to select the next customer j to visit immediately after customer i

$$p_{ij} = \begin{cases} \frac{\xi_{ij}}{\sum_{l \in \Re} \xi_{il}} & \text{for } j \in \Re, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The process is continued until the route for the vehicle is built. Then the procedure is repeated for other vehicles.

2.3. Local search

After the construction of a solution by an ant, the solution is improved by local search. Three types of local search schemes are used including two multi-route schemes. We use 2-*opt* local search scheme to improve the solution generated. Other two local search schemes are the customer insertion/interchange multi-route scheme and the sub-path exchange multi-route scheme. These local search schemes are described in detail in the next sub-sections. Two multi-route schemes are used iteratively until both are not able to improve the solution. The best solution obtained by the insertion/interchange scheme is used as an initial solution for the sub-path exchange scheme and the best solution obtained by the sub-path exchange scheme is used as an initial solution for the insertion/interchange scheme in the subsequent cycle. The process is repeated until both multi-route schemes are not able to improve the solution. During our experiment, we found that generally a local minimum is reached after three or four cycles.

2.3.1. 2-opt local search scheme

The 2-*opt* local search scheme was originally proposed by Lin (1965) for improving the route in a traveling salesman problem (TSP). The 2-*opt* scheme is the one of the best known local search scheme for TSP and it is based on the arc exchange approach. In CVRP, each vehicle route is a traveling salesman problem. The 2-*opt* scheme starts with a given route and breaks it at two places to generate three segments. The route is then reconstructed by reversing the middle segment (the segment which does not contain the depot). A given route is broken at each combination of two places and is updated whenever there is an improvement. The process is repeated until there is no further improvement in the solution (i.e., until there is no improvement in the solution in one complete cycle of the search). In VRPB, we would need to treat the linehaul and backhaul customers separately while using the 2-*opt* scheme.

2.3.2. Customer insertion/interchange multi-route scheme

This approach uses two types of operations for a given customer. These two operations are described below for customer i assigned to vehicle p:

1. Insertion operation: Customer i is removed from his original position and inserted in all vehicle routes (including the same vehicle route). Vehicle capacity is checked before the insertion of customer i to another vehicle. Our variant of VRPB requires that each vehicle should serve at least one linehaul customer and the linehaul customers on a route must be served before backhaul customers. In order to maintain this feasibility, the insertion operation for a linehaul customer is used only when the remaining number of linehaul customers assigned to vehicle p is more than 1. To insure feasibility, a linehaul customer is allowed to be inserted at the following positions in the route.
   - just after the depot,
   - between two linehaul customers,
   - between the last linehaul customer and the first backhaul customer.

   Similarly, a backhaul customer is allowed to be inserted at the following positions in the route.
   - between the last linehaul customer and the first backhaul customer,
   - between two backhaul customers,
   - just before the depot.
2. Interchange Operation: In this operation, customer i (from vehicle p) is shifted to another vehicle q and customer j from vehicle q is shifted to vehicle p. We place customer i on vehicle q at his *best* insertion place and similarly we place customer j on vehicle p at his *best* insertion place.

In order to reduce CPU time, the following neighborhood criteria is used before an insertion/interchange operation is performed. Let the ϵ be some fixed number and let ϵ- neighborhood for customer i, $N_\epsilon(i)$ contain the ϵ customers closest (in term of the distance) to customer i. The interchange operation between customer i and customer j is performed only if at least one of the ϵ neighbors of customer i is served by vehicle q and at least one of the ϵ neighbors of customer j is served by

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

vehicle p. Similarly, the insertion of customer i on vehicle q is performed only if one of the ϵ neighbors of customer i is served by vehicle q.

The steps in the local search are:

Step 1: Initialize $\Omega = \{1, 2, \ldots, n\}$.
Step 2: Randomly choose a customer, say customer i from vehicle p for evaluation, and remove this customer from set Ω.
Step 3: Perform insertion of customer i to determine the best insertion position and evaluate the total tour length of the resultant solution.
Step 4: Consider also the interchange of customer i and each customer j. Both customers i and j must be either linehaul customers or backhaul customers and the interchange should be feasible in terms of the vehicle capacity. Identify the best interchange customer j and the corresponding solution after performing the interchange operation.
Step 5: Choose the best solution obtained from *Steps 3* and *4* and compare it with the current solution. If the best solution is better than the current solution then update the current solution.
Step 6: If $\Omega = \phi$ then stop and report the current solution, else go to *Step 2*.

The customer insertion/interchange multi-route improvement scheme is similar to the intersection/exchange schemes used by other researchers (e.g., Osman, 1993; Van Breedam, 1995; Kindervater and Savelsbergh, 1997). Our insertion operation is similar with previous work but our interchange operation differs in the way a customer is placed in the new route. In previous work, the positions of customer i from vehicle p and customer j from vehicle q are interchanged. In our approach, customer i from vehicle p is placed in the route of vehicle q at his *best* insertion place. Similarly, customer j from vehicle q is placed in the route of vehicle p at his *best* insertion place.

The cost of the solution generated by the interchange as well as the insertion operation can be determined in constant time. In the absence of the customer neighborhood criteria, the theoretical computational complexity of one iteration of the insertion/interchange multi-route scheme can be calculated as follows.

The insertion operation has complexity of $O(l^2 + b^2)$. The interchange operation considers a customer for interchange with each customer from another vehicle. Hence for linehaul customers, there are a total of $l \times (l - l/v)$ pairs for possible interchange assuming that n customers are distributed evenly among the v routes. Each customer from a given pair is placed at his best insertion place. Thus, a customer is evaluated at l/v places to find out the best insertion place for him. Therefore, the overall complexity of an interchange operation for the linehaul customers is

$$O(l) \times O(l - l/v) \times O(l/v) + O(l/v) = O(l^3/v^2).$$

Similarly the interchange operation for backhaul customers has complexity of $O(b^3/v^2)$. Hence, the overall complexity of the insertion/interchange multi-route scheme is

$$O(l^2 + b^2) + O(l^3/v^2) + O(b^3/v^2) = O(l^3/v^2 + b^3/v^2).$$

The introduction of the customer neighborhood criteria should reduce CPU time in practice. However, in the worst case scenario, there may still be $l \times (l - l/v)$ pairs of linehaul customers and $b \times (b - b/v)$ pairs of backhaul customers for interchange even in the presence of the customer neighborhood criteria. Thus, the theoretical computational complexity of the scheme remains the same with or without the customer neighborhood criteria.

2.3.3. Sub-path exchange multi-route scheme

The customer insertion/interchange multi-route scheme considers shifting a given customer to another vehicle route. The sub-path exchange multi-route scheme can shift more than one customer from one route to another. Suppose $S = \{R_1, R_2, \ldots, R_p, \ldots R_q, \ldots, R_{v-1}, R_v\}$ represents a solution to VRPB. Here, R_p denote the route of vehicle p. The sub-path exchange multi-route scheme considers two routes R_p and R_q and combines them in two new routes R'_p and R'_q to produce a new solution $S' = \{R_1, R_2, \ldots, R'_p, \ldots, R'_q, \ldots, R_{v-1}, R_v\}$. Consider sub-path $(e, \ldots, f)$ belonging to route R_p and sub-path $(g, \ldots, h)$ belonging to R_q. To maintain feasibility, all customers in sub-path $(e, \ldots, f)$ and sub-path $(g, \ldots, h)$ are either linehaul or backhaul customers. Furthermore, the feasibility in terms of vehicle capacity is checked before an exchange is considered. The new route R'_p is obtained by replacing sub-path $(e, \ldots, f)$ by either sub-path $(g, \ldots, h)$ or reverse sub-path $(h, \ldots, g)$. Similarly, the new route R'_q is obtained by replacing sub-path $(g, \ldots, h)$ by either sub-path $(e, \ldots, f)$ or reverse sub-path $(f, \ldots, e)$.

We again use the customer neighborhood criteria to reduce the CPU time. The exchange of sub-path $(e, \ldots, f)$ of vehicle p with sub-path $(g, \ldots, h)$ of vehicle q is considered only if customer f is among the 15 nearest neighbors of customer g.

Each possible sub-path from route R_p and from route R_q is considered for possible exchange and the best pair of sub-paths is used to produce new routes R'_p and R'_q. The calculation of the cost of the route produced by an exchange can be performed in constant time. Let γ_i represent the immediate predecessor of customer i and π_i represent the immediate successor of customer i. Then the cost (i.e., total tour length) of new solution S' is calculated as follows:

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

$$\begin{aligned}\text{Cost}(S') = \text{Cost}(S) &- c(\gamma_e, e) - c(f, \pi_f) - c(\gamma_g, g) - c(h, \pi_h) \\ &+ \min\{(c(\gamma_e, g) + c(h, \pi_f)), (c(\gamma_e, h) + c(g, \pi_f))\} \\ &+ \min\{(c(\gamma_g, e) + c(f, \pi_h)), (c(\gamma_g, f) + c(e, \pi_h))\}.\end{aligned} \tag{5}$$

There are a total of $v \times (v-1)$ combinations of vehicles for the sub-path exchange. Assuming that n customers are evenly distributed among v routes, there are $l/v \times (l/v - 1)/2$ sub-paths for the linehaul customers in a vehicle route. Thus, the overall complexity of the sub-path exchange scheme for linehaul customers in the absence of neighborhood criteria is

$$O(v^2) \times O(l^2/v^2) \times O(l^2/v^2) = O(l^4/v^2).$$

Similarly, the overall complexity of the sub-path exchange scheme for backhaul customers is $O(b^4/v^2)$. Therefore, the overall complexity of one iteration of the sub-path exchange local search in absence of neighborhood criteria is $O(l^4/v^2 + b^4/v^2)$.

With the customer neighborhood criteria, a given sub-path containing linehaul customers can be exchanged with the maximum of $\epsilon \times l/v$ sub paths from the other vehicles. Thus, the overall complexity of the sub-path exchange scheme for linehaul customers with the neighborhood criteria is

$$O(v^2) \times O(l^2/v^2) \times O(\epsilon \times l/v) = O(l^3/v).$$

Similarly, the overall complexity of the sub-path exchange scheme for backhaul customers is $O(b^3/v)$. Therefore, the overall complexity of one iteration of the sub-path exchange local search schemes with the neighborhood criteria is $O(l^3/v + b^3/v)$.

The sub-path exchange multi-route scheme is similar to CROSS exchange heuristic of Taillard et al. (1997) and ICROS heuristic of Bräysy and Dullaert (2003). Our sub-path exchange multi-route scheme falls between the above two heuristics. Considering all possible sub-paths for possible exchange is similar to CROSS heuristic whereas connecting the reverse sub-paths is similar to ICROSS heuristic.

2.4. Updating elitist ants

We use elitist ants to update trail intensities. We use σ elitist ants which are distinct from one another. Elitist ants are updated by comparing the present elitist ant solutions with the current ant solution. If the current ant solution is found to be better than the μth elitist ant solution, then the current ant solution becomes the new μth elitist ant solution. The ranks of the previous μth elitist ant and the elitist ants below it are lowered by one. All current solutions are considered for updating the elitist ants. In order to keep σ elitist ants distinct from one another, the current solution is compared only if the total tour length in the current solution is not equal to the total tour length in any of the elitist ant solutions. The hashing function used by Osman and Wassan (2002) can be used instead of the total tour length for identifying the identical solutions, but calculation of the hashing function would require more CPU time.

2.5. Updating trail intensities

Vehicle-trail intensities and *route-trail* intensities are updated using the elitist ant solutions. *Vehicle-trail* intensities are updated as follows:

$$\tau 1_{ik}^{\text{new}} = \rho \tau 1_{ik}^{\text{old}} + \Delta\tau 1_{ik}^{*} + \sum_{\mu=1}^{\sigma} \Delta\tau 1_{ik}^{\mu} \quad i = 1, 2, \ldots, n; \;\; k = 1, 2, \ldots, v, \tag{6}$$

where

$$\Delta\tau 1_{ik}^{\mu} = \begin{cases} 1/L^{\mu} & \text{if customer } i \text{ is served by vehicle } k \text{ in the } \mu\text{th elitist ant solution,} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

$$\Delta\tau 1_{ik}^{*} = \begin{cases} 1/L^{*} & \text{if customer } i \text{ is served by vehicle } k \text{ in the best solution found to date,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Here, ρ is the trail persistence and L^{μ} is the total tour length in the μth elitist ant solution and L^{*} is the total tour length in the best solution found to date.

Similarly, *Route-trail* intensities are updated as follows:

$$\tau 2_{ij}^{\text{new}} = \rho \tau 2_{ij}^{\text{old}} + \Delta\tau 2_{ij}^{*} + \sum_{\mu=1}^{\sigma} \Delta\tau 2_{ij}^{\mu}, \quad i = 1, 2, \ldots, n; \;\; j = 1, 2, \ldots, n; \;\; i \neq j, \tag{9}$$

where

$$\Delta\tau 2_{ij}^{\mu} = \begin{cases} 1/L^{\mu} & \text{if customer } j \text{ is visited after customer } i \text{in the } \mu\text{th elitist ant-solution,} \\ 0 & \text{otherwise,} \end{cases} \quad (10)$$

$$\Delta\tau 2_{ij}^{*} = \begin{cases} 1/L^{*} & \text{if customer } j \text{ is visited after customer } i \text{ in the best solution found so far,} \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

The scheme allows all elitist ants to lay trails with *equal importance* as opposed to the updating scheme used in an ant colony system algorithm for VRP by Bullnheimer et al. (1999). The strategy of giving more importance to the best elitist ant solution and less importance to the solutions of other elitist ants can divert the search towards the global minimum but it can also increase the risk of being trapped at a local minimum.

The best solution found so far will be equivalent to the top elitist ant solution when COUNT = 0 but it may differ from the top elitist ant solution when COUNT > 0.

3. Numerical analysis

This section presents the numerical results for MACS and compares MACS with the existing heuristic and metaheuristic approaches.

3.1. Parameter settings

The quality of the solution depends on the number of ants used which ultimately affects the CPU time of the algorithm. We use 25 artificial ants of each type in MACS, i.e., at each iteration, 25 solutions are generated and each solution is improved by local search. We reinitialize the trail intensities and destroy the elitist ants if there is no change in the elitist ant solutions in the last 10 iterations. Our algorithm stops after the trail intensities are reinitialized 5 times.

We set $\alpha 1 = \beta 1 = 3$, $\alpha 2 = \beta 2 = 1$, $\rho = 0.95$, $\sigma = 6$. We thus use six elitist ants to update the pheromone. Our customer interchange local search scheme uses €-neighborhood of size 10. We found that € of more than 10 increases the CPU time without enhancing the solution quality much. If none of the 10 neighboring customers are served by the given vehicle, then the probability is low that this customer is served by the same vehicle in the optimal solution. Similarly, our sub-path multi-route scheme uses €-neighborhood of size 15. Parameters $\alpha 1$, $\beta 1$, $\alpha 2$, $\beta 2$, ρ and σ are set by performing sensitivity analysis carried out in limited CPU time.

3.2. Benchmark problem

The numerical experiment is performed using two sets of problem instances available in literature. The first set (set-1) was proposed by Goetschalckx and Jacobsblecha (1989). This data set includes 62 instances where the total number of customers ranges from 25 to 150 with backhaul customers being 25–50% of the linehaul customers. In problem instances for set-1, customer coordinates are uniformly distributed in interval $[0, 24{,}000]$ for the x coordinate, and in interval $[0, 32{,}000]$ for the y coordinate. The depot is located centrally at ($x = 12{,}000$, $y = 16{,}000$). Customer demands are generated from Normal distribution with mean of 500 and standard deviation of 200. The second set (set-2) was proposed by Toth and Vigo (1997). This set consists of 33 instances with 21–100 customers. These instances are generated from 11 problems chosen from the VRP literature. Three instances are generated from each problem with backhaul customers being 50%, 66% and 80% of the linehaul customers.

Euclidean distance between two customers is calculated in two ways in the literature. In the first approach, the distance is multiplied by 10 and is rounded to the nearest integer. The total distance is then divided by 10 and rounded to the nearest integer. In the second approach, the distance is used without rounding but the total tour length is rounded to two decimal places. Since many studies use the distance without rounding, we use only approach 2.

3.3. Existing heuristics for VRPB

We compare MACS with the following metaheuristic algorithms:RTS-AMP: reactive tabu adaptive memory programming of Wassan (2004).

LNS: large neighborhood search by Ropke and Pisinger (2006).

BTS: new tabu search of Brandao (2006).

We do not include reactive tabu search (RTS) in our table because RTS-AMP has produced either similar or better quality solutions on all instances at a lower computational cost compared to RTS. Brandao (2006) developed three types of

tabu search; namely *open*, *K-tree* and *K-tree_r* tabu search. Out of these three methods, *K-tree_r* seems to yield better solutions on average compared to the other two methods. Therefore, we use the solutions obtained by *K-tree_r* method alone and call it BTS for comparison purposes. Similarly, Ropke and Pisinger (2006) developed three configuration of large-scale neighborhood search; namely *standard*, *6R-no learning* and *6R-normal learning*. Out of these three methods, we use the solutions obtained by *6R-no learning* configuration and call it LNS in our comparison. Each of the studies reports solutions in a different way. RTS-AMP reports the best solution from 8 runs, LNS reports the best solution from 10 runs and BTS reports a single run solution but uses the best run from 5 runs.

Generally, it is difficult to compare the CPU times of different algorithms as the computational experiments have been performed on different machines. A comparison of CPU time can be done using Mflops (million floating point calculation per second) values of different computers given in the Linpack benchmark report of Dongarra (2006). Specifications of different computers and their Mflops values are presented in Table 7. We use the Mflops values to scale the running time of an algorithm in reference to our computer (i.e., Intel Xeon 2.4 GHz). The scaled CPU time represents the approximate CPU time if Intel Xeon 2.4 GHz were to be used to run the particular algorithm. We report the average CPU time of MACS over 8 runs. In order to keep the CPU time of MACS comparable with other algorithms, we have restricted the number of ants of each type to 25.

3.4. Computational experiment and performance analysis of the algorithms

The proposed MACS was coded in C and implemented on an Intel Xeon with 2.4 GHz computer system. We run MACS 8 times and report the results under different criteria. The detailed results are reported in Tables 5 and 6 for data set-1 and set-2, respectively. We summarize the overall results of MACS in Tables 1 and 2 using the following performance measures.

1. Overall average solution: The overall average cost over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem instance, the average cost is calculated from costs over 8 runs.
2. Average standard deviation: The average of the standard deviation over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem, the standard deviation is computed using the solutions from 8 runs.
3. Average best solution: The average of the best solutions over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem instance, the cost associated with the best solution is obtained from 8 runs.
4. Average best run: We execute the algorithm for 8 runs. The best run is the run that gives the lowest average over 62 problem instances for set-1 and 33 problem instances for set-2. The lowest average is called "average best run".
5. Average scaled CPU time: The average of the scaled CPU time per run over 62 problem instances for set-1 and 33 problem instances for set-2. For each problem instance, the scaled CPU time per run is calculated using the scaled CPU time over 8 runs.

We present summary results of different algorithms in Table 2. A blank space in Table 2 indicates that the corresponding summary value was not reported in the original paper.

Table 1
Average performance of MACS over 62 instances for set-1 and 33 instances for set-2

	Overall average solution	Average standard deviation	Average best solution	Average best run	Average scaled CPU time
Set-1	290920.90	276.04	290655.29	290838.73	67.57
Set-2	702.35	0.7985	701.48	701.76	25.64

Table 2
Comparison of MACS with different heuristics for VRPB

	RTS-AMP	LNS	BTS	MACS
Set-1				
Overall average solution	–	291823.34	291305.7	290920.90
Average best solution	290981.8	291014.7	–	290655.29
Average best run	–	–	291160.4	290838.73
Average scaled CPU time	20.20	26.22	66.84	67.57
Set-1				
Overall average solution	–	704.54	702.50	702.35
Average best solution	706.48	701.18	–	701.48
Average best run	–	–	702.15	701.76
Average scaled CPU time	7.95	15.55	24.40	25.64

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Table 3
Effect of trail intensities and local search on solution quality

Setting	Set-1		Set-2	
	Overall average solution	% deviation from the average best known sol.	Overall average solution	% deviation from the average best known sol.
No change from current MACS	290920.90	0.13	702.35	0.22
Local search applied on 25 randomly generated solution	293435.69	0.99	710.70	1.41
In the absence of the sub-path exchange multi-route scheme	291661.34	0.38	704.92	0.59
In the absence of the insertion/interchange multi-route scheme	294465.94	1.35	710.73	1.42

The results for set-1 reported in Tables 1 and 2 indicate that the proposed multi-ant colony system (MACS) gives better results than the other heuristics. Table 2 shows that MACS has the lowest overall average solution followed by BTS and LNS. Also, Table 2 indicate that MACS is on average better than the other algorithms in terms of the best solution and the best run. In terms of CPU time, MACS and BTS are using equivalent CPU times. LNS takes less than half of the CPU time than MACS and RTS-AMP takes one third of the CPU time than MACS. However, MACS does have a control over solution quality and the CPU time and this feature is described in Section 4.

The computational results for set-2 have similar trend in terms of the average solution. MACS has the lowest overall average solution followed by BTS and LNS. The average best run solution of MACS is slightly better than BTS. The overall average solution of MACS is better than LNS and RTS-AMP but the average best solution of MACS is slightly worse than the average best solution of LNS. The trend of CPU time is similar to the trend for set-1.

Finally, note that we have found four new best known solutions for set-1 and 1 new best known solution for set-2. The new best solutions are described in Appendices A and B for set-1 and set-2, respectively. The new best known solutions are shown in bold in Tables 5 and 6.

3.5. Effect of trail intensities and local search schemes on MACS

In order to see the effect of trail intensities and local search schemes on MACS, we executed MACS and calculate the average solution over 8 runs under each setting. In Table 3, we report the overall average solution its percentage deviation from the average best known solution for each setting. The average best known solution is the average of the best known solutions for 62 problem instances for set-1 and 33 problem instances for set-2 as reported in the literature. For each setting, we keep the parameter values to be same and keep the number of ants to be 25. However, we vary the stopping criteria to keep CPU time comparable. The first row in Table 3 reports the solution without any change in the algorithm. The second row reports the solution when 25 randomly generated vehicle routes are improved by three local search schemes used in MACS. In this case, the average solution deteriorates from 0.13% to 0.99% for set-1 and from 0.22% to 1.41% for set-2. This result indicates that the trail intensities play a crucial role for diverting the solution towards the global optimum and shows that a purely random search cannot explore the promising regions in an effective way. When sub-path exchange multi-route scheme is not used in MACS, the percentage deviation deteriorates to 0.38 for set-1 and 0.59 for set-2. Similarly, when the customer insertion/interchange multi-route scheme is not used, the percentage deviation value deteriorates to 1.35 for set-1 and 1.42 for set-2. These two results indicate that the customer inser-

Table 4
The effect of number of ants on solution quality and CPU time

Number of ants of each type	Set-1			Set-2		
	Overall average solution	Average best solution	Average CPU time (seconds)	Overall average solution	Average best solution	Average CPU time (seconds)
5	291430.90	290827.20	15.63	703.72	701.73	6.52
10	291128.80	290724.20	30.25	703.08	701.76	11.86
15	291049.90	290719.00	44.25	702.70	701.70	15.81
20	291010.80	290658.40	57.03	702.58	701.67	20.81
25	290920.90	290655.29	67.57	702.35	701.48	25.64
30	290894.10	290662.20	80.16	702.17	701.36	29.92
35	290891.00	290622.90	91.34	702.11	701.18	36.30
40	290868.10	290642.80	103.75	702.05	701.33	41.08
45	290861.10	290629.60	116.94	701.88	701.15	44.18
50	290848.40	290622.80	127.12	701.80	701.15	49.76

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Table 5
Total tour length obtained by MACS for data set-1

No.	Class	Average solution	Standard deviation	Best solution	Best run	Overall best solution[a]	Average CPU time
1	A1	229885.65	0	229885.65	229885.65	229885.65	1.00
2	A2	180119.21	0	180119.21	180119.21	180119.21	1.75
3	A3	163405.38	0	163405.38	163405.38	163405.38	3.00
4	A4	155796.41	0	155796.41	155796.41	155796.41	1.88
5	B1	239080.16	0	239080.16	239080.16	239080.15	2.13
6	B2	198047.77	0	198047.77	198047.77	198047.77	2.50
7	B3	169372.29	0	169372.29	169372.29	169372.29	2.00
8	C1	250556.77	0	250556.77	250556.77	250556.77	3.88
9	C2	215020.23	0	215020.23	215020.23	215020.23	4.13
10	C3	199345.96	0	199345.96	199345.96	199345.96	4.88
11	C4	195366.63	0	195366.63	195366.63	195366.63	3.88
12	D1	322530.13	0	322530.13	322530.13	322530.13	6.13
13	D2	316708.86	0	316708.86	316708.86	316708.86	6.25
14	D3	239478.63	0	239478.63	239478.63	239478.63	5.63
15	D4	205831.94	0	205831.94	205831.94	205831.94	6.50
16	E1	238879.58	0	238879.58	238879.58	238879.58	6.75
17	E2	212263.11	0	212263.11	212263.11	212263.11	6.50
18	E3	206659.17	0	206659.17	206659.17	206659.17	10.38
19	F1	263173.96	0	263173.96	263173.96	263173.97	11.13
20	F2	265214.16	0	265214.16	265214.16	265214.16	9.13
21	F3	241484.85	420.14	241120.78	241120.78	241120.77	11.25
22	F4	233861.85	0	233861.85	233861.85	233861.84	15.00
23	G1	307007.11	190.04	306536.78	307074.3	306305.40	18.00
24	G2	245440.99	0	245440.99	245440.99	245440.99	10.38
25	G3	229507.48	0	229507.48	229507.48	229507.48	14.25
26	G4	232521.25	0	232521.25	232521.25	232521.25	21.75
27	G5	221730.35	0	221730.35	221730.35	221730.35	20.38
28	G6	213457.45	0	213457.45	213457.45	213457.45	20.63
29	H1	268996.28	178.8	268933.06	268933.06	268933.06	24.50
30	H2	253365.50	0	253365.50	253365.5	253365.50	21.50
31	H3	247449.04	0	247449.04	247449.04	247449.04	20.25
32	H4	250220.77	0	250220.77	250220.77	250220.77	27.13
33	H5	246121.31	0	246121.31	246121.31	246121.31	26.00
34	H6	249135.32	0	249135.32	249135.32	249135.32	30.25
35	I1	350395.33	219.17	350245.28	350245.28	350245.28	41.63
36	I2	310318.17	1058.78	309943.84	309943.84	309943.84	37.50
37	I3	294839.85	510.65	294507.38	294507.38	294507.38	43.88
38	I4	296129.01	260.39	295988.45	295988.45	295988.44	46.63
39	I5	301826.52	894.41	301236.01	302441.47	301236.00	47.38
40	J1	335124.95	218.99	335006.68	335479.75	335006.69	66.63
41	J2	310417.21	0	310417.21	310417.21	310417.21	59.25
42	J3	279343.35	351.11	279219.21	279219.21	279219.21	77.63
43	J4	296584.68	145.71	296533.16	296533.16	296533.16	62.75
44	K1	396139.64	676.66	395075.67	395075.67	**394071.16**	104.88
45	K2	362563.92	462.14	362130.00	362130	362130.00	96.88
46	K3	366709.20	873.68	365694.08	365694.08	365694.09	114.25
47	K4	350317.26	751.71	349870.36	349870.36	348949.39	106.00
48	L1	420063.78	1307.78	417921.58	418828.93	**417896.72**	148.38
49	L2	401360.36	318.64	401247.70	401247.7	401228.81	142.00
50	L3	404315.09	1896.83	402677.72	403990.54	402677.72	160.63
51	L4	384834.32	274.73	384636.33	385225.91	384636.34	159.50
52	L5	390329.49	1192.23	387564.55	390987.74	387564.56	158.00
53	M1	399120.67	587.75	398729.82	398729.82	398593.19	229.50
54	M2	398156.41	563.83	397324.15	397664.54	396916.97	183.38
55	M3	377812.03	296.44	377328.75	377328.75	**375695.41**	183.75
56	M4	348464.23	69.28	348417.94	348512.42	**348140.16**	164.13
57	N1	408168.11	124.97	408100.62	408100.62	408100.62	213.88
58	N2	408246.94	246.84	408065.44	408065.44	408065.44	214.38
59	N3	394701.38	864.08	394337.86	396827	394337.86	199.75
60	N4	394866.92	108.42	394788.36	394788.36	394788.37	219.00
61	N5	374120.15	561.91	373723.46	373723.46	373476.31	272.00
62	N6	374791.55	1488.52	373758.65	373758.65	373758.656	255.00
Average		290920.92	276.04	290655.29	290838.73	290576.218	67.567

[a] Overall best solution is the cost associated with the best solution found from different runs while performing overall activity of MACS.

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Table 6
Total tour length obtained by MACS for data set-2

No.	Class	Average solution	Standard deviation	Best solution	Best run	Overall best solution[a]	Average CPU time
1	eil22_50	371.00	0.00	371	371	371	1
2	eil22_66	366.00	0.00	366	366	366	0.13
3	eil22_80	375.00	0.00	375	375	375	0.5
4	eil23_50	682.00	0.00	682	682	682	1.5
5	eil23_66	649.00	0.00	649	649	649	0.63
6	eil23_80	623.00	0.00	623	623	623	3
7	eil30_50	501.00	0.00	501	501	501	1.38
8	eil30_66	537.00	0.00	537	537	537	2
9	eil30_80	514.00	0.00	514	514	514	3
10	eil33_50	738.00	0.00	738	738	738	4.25
11	eil33_66	750.00	0.00	750	750	750	2.88
12	eil33_80	736.00	0.00	736	736	736	2
13	eil51_50	559.00	0.00	559	559	559	8.25
14	eil51_66	548.00	0.00	548	548	548	10.75
15	eil51_80	565.00	0.00	565	565	565	13.25
16	eilA76_50	739.25	0.46	739	739	739	19
17	eilA76_66	768.00	0.00	768	768	768	23.63
18	eilA76_80	782.63	3.85	781	781	781	53.5
19	eilB76_50	801.00	0.00	801	801	801	21.25
20	eilB76_66	873.13	0.35	873	873	873	25.38
21	eilB76_80	920.13	2.80	919	919	919	38.25
22	eilC76_50	713.00	0.00	713	713	713	18.75
23	eilC76_66	734.00	0.00	734	734	734	27.88
24	eilC76_80	736.13	1.55	734	734	733	36
25	eilD76_50	690.00	0.00	690	690	690	27.88
26	eilD76_66	715.00	0.00	715	715	715	28.88
27	eilD76_80	698.63	1.19	696	699	694	42.13
28	eilA101_50	834.75	2.60	831	832	831	56.13
29	eilA101_66	846.00	0.00	846	846	846	59.13
30	eilA101_80	866.88	5.14	860	860	857	88.25
31	eilB101_50	927.63	2.62	925	930	**923**	67
32	eilB101_66	1008.88	5.03	1002	1002	988	84.88
33	eilB101_80	1008.50	0.76	1008	1008	1008	73.88
Average		702.34	0.80	701.48	701.76	700.82	25.644

[a] Overall best solution is the cost associated with the best solution found from different runs while performing overall activity of MACS.

tion/interchange scheme is more effective than the sub-path exchange multi-route scheme. It is interesting to note that a combination of the two schemes is necessary to produce better quality solutions in the same CPU time. In addition, a hybrid local search that combines different strategies such as insertion, interchange, exchange, etc., seems to produce better solutions than the individual strategies.

4. Effect of number of ants on MACS

An interesting feature of MACS is that the quality of the solution can be controlled by varying the number of ants. In Table 4, we vary the number of ants and report the overall average solution and the average of best solution. It is evident that the solution does not deteriorate too much when we reduce the number of ants and thereby reduce the CPU time.

Table 7
Summary of the CPU's used in testing various heuristics and rough conversion factors (r) relative to a 2.4 GHz Intel Xeon processor with 884 Mflops

Reference	CPU used	Mflops	r
HTV96 and TV99 Toth and Vigo (1996, 1999)	IBM 386/20	1.3	0.001
RTS-AMP (Wassan, 2004)	Sun Sparc1000 at 50 MHz	10	0.011
BTS (Brandao, 2006)	Pentium III at 500 MHz	72.5	0.082
LNS (Ropke and Pisinger, 2006)	Pentium IV at 1.5 GHz	326	0.369
MACS	Intel Xeon 2.4 GHz	884	1

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Table 4, also allows us to compare MACS with other heuristic solutions keeping the CPU time compatible. The CPU time of MACS is comparable with the CPU time of LNS when the number of ants is set to 10. The solution quality of MACS when the number of ants is 10 is still better than the solution quality of LNS in terms of the average solution for both set-1 and set-2. Similarly, the CPU time of RTS-AMP is comparable with the CPU time of MACS when number of ants is set to 5. The solution quality of MACS is still better than the solution quality of RTS-AMP under the criteria of best solution for both set-1 and set-2.

5. Conclusions

This paper presents a multi-ant colony system (MACS) algorithm for solving VRPB. MACS contains the following features:

1. Two types of ants, vehicle-ants and route-ants are used. Similarly, two types of trail intensities – vehicle trail intensity and route trail intensity – are used to construct a feasible solution.
2. An inherent part of ant colony system is the local search. Two multi-route local search schemes – the customer insertion/interchange scheme and the sub-path exchange scheme – are used in this paper.
3. While updating trail intensities, no distinction is made between the elitist ants; i.e., equal importance is given to each elitist ant. This reduces the risk of being trapped at a local minimum.
4. The trail intensities are reinitialized and elitist ants are destroyed when there is no change in the elitist ant solutions in consecutive 10 iterations. The best solution found to date is used in reinitializing the trail intensities.

Extensive computational experiment on benchmark problems has shown that the overall proposed multi-ant colony system (MACS) gives better results than the other algorithm. In addition, MACS has produced five new best known solutions for the benchmark problem instances available in the literature. Another interesting feature of MACS is that the solution quality and the CPU time of MACS can be controlled by varying the number of ants. Further research can be done to improve the local search schemes in ant colony systems. Hybrid local search schemes can make the ant colony system approach more effective for the other variants of the vehicle routing problem.

Acknowledgements

We are thankful to Professors P. Toth and D. Vigo for providing us with data set-2. We accessed data set-1 from the website provided by Goetschalckx, M. and C. Jacobsblecha. We are thankful to the reviewers for their valuable comments. This research is supported by NSERC discovery Grant 1226-00.

Appendix A. New best solutions obtained by MACS for set-1

Problem 44

Number of customers	Route	Delivery demand	Pickup Demand	Length
8	0 4 3 35 44 67 50 26 105 0	3556	311	23156.90
8	0 15 23 52 51 18 59 28 54 0	3877	0	25982.26
13	0 49 48 16 68 41 57 1 110 94 103 113 91 109 0	4047	3669	46279.98
11	0 37 6 47 69 63 2 73 84 92 90 108 0	3812	1458	42502.46
8	0 70 53 75 45 71 14 88 101 0	3573	487	28839.83
14	0 24 7 17 25 66 46 21 85 112 86 106 82 81 78 0	3969	3575	52502.62
9	0 55 31 38 43 10 56 72 30 77 0	4098	890	35340.09
14	0 39 11 27 36 61 5 22 32 8 76 87 104 80 107 0	3758	2748	39654.36
12	0 58 60 13 74 34 29 19 102 99 97 79 111 0	3764	1940	44937.95
16	0 12 65 40 9 33 42 20 62 64 93 83 100 95 96 98 89 0	3737	3485	54874.72
$n = 113$, $n_l = 75$, $n_b = 38$		$Q = 4100$		394071.17

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Problem 48

Number of customers	Route																					Delivery demand	Pickup Demand	Length
13	0	9	43	19	15	23	20	21	99	131	86	133	104	98	0							3062	3729	39554.42
15	0	64	1	57	56	72	41	31	60	59	45	93	127	113	101	91	0					4376	2820	46503.38
19	0	33	74	34	68	61	5	37	70	69	12	115	150	105	95	84	139	136	120	78	0	4341	4199	56518.64
2	0	54	146	0																		215	521	3924.20
18	0	47	65	2	30	29	13	62	24	49	118	126	145	129	87	108	117	112	80	0		3996	4323	43818.08
19	0	63	67	52	66	35	3	28	14	18	121	89	148	103	122	96	125	130	114	119	0	4345	4126	48727.66
17	0	39	46	55	7	16	48	58	75	123	144	83	137	140	143	77	116	102	0			3931	4219	43651.00
17	0	73	22	6	44	42	8	32	38	85	135	76	92	97	128	149	111	142	0			3585	4082	53799.63
14	0	10	40	71	4	11	27	94	134	107	110	109	79	90	100	0						3042	3604	38454.92
16	0	36	26	50	53	51	17	25	82	138	88	141	106	124	132	81	147	0				4200	4078	42944.78
$n = 150$, $n_l = 75$, $n_b = 75$																						Q = 4400		417896.71

Problem 55

Number of customers	Route																					Delivery demand	Pickup Demand	Length
19	0	51	64	47	62	37	20	42	40	25	17	54	31	7	103	124	110	122	105	117	0	6045	2871	58765.01
18	0	3	35	84	55	88	39	79	46	53	45	24	9	33	98	102	114	101	111	0		6059	1587	59074.48
18	0	15	90	65	66	91	75	36	22	32	29	52	83	71	119	125	118	115	108	0		6135	2790	56400.34
15	0	67	2	99	89	28	43	19	18	96	8	6	106	112	104	107	0					5935	2125	48755.24
13	0	76	63	30	69	68	13	80	48	16	73	4	123	116	0							6156	1003	50132.69
13	0	41	14	94	85	61	5	92	11	27	113	120	109	121	0							5817	2054	30833.82
12	0	59	10	56	72	26	81	50	44	93	100	77	38	0								6198	0	28971.73
6	0	97	49	82	74	34	23	0														2281	0	12135.21
11	0	60	57	1	58	87	78	21	86	12	70	95	0									5578	0	30892.25
$n = 125$, $n_l = 100$, $n_b = 25$																						$Q = 6200$		375695.42

Problem 56

Number of customers	Route																							Delivery demand	Pickup Demand	Length
22	0	65	66	91	75	45	71	83	52	29	32	22	36	9	24	33	98	102	119	125	118	115	108	0	7636	3482	69743.48
22	0	41	95	70	57	1	58	87	78	21	86	12	25	17	54	31	7	103	124	110	122	105	117	0	7372	2871	62826.18
19	0	15	90	53	46	79	39	88	55	28	43	19	18	96	8	6	106	112	104	107	0				7799	2125	62637.38
17	0	38	77	100	93	44	50	73	48	16	68	13	80	4	123	114	101	111	0						6898	1471	54704.53
16	0	76	51	64	14	94	85	61	5	92	11	27	40	113	120	109	121	0							6941	2054	34384.73
15	0	59	10	26	81	63	30	69	42	20	37	62	47	56	72	116	0								7795	427	33675.12
14	0	23	34	74	82	97	49	3	35	84	67	2	99	89	60	0									5763	0	30168.75
$n = 125$, $n_l = 100$, $n_b = 25$																										Q = 8000	348140.16

Appendix B. New best solutions obtained by MACS for set-2

Problem 31

Number of customers	Route	Delivery demand	Pickup Demand	Length
15	0 45 42 23 9 3 43 46 57 71 100 99 96 98 97 53 0	112	95	113.00
14	0 48 49 19 30 50 47 31 58 72 69 93 92 80 59 0	112	110	117.00
15	0 14 39 2 15 40 17 41 89 67 62 90 84 88 75 64 0	112	104	126.00
15	0 27 7 44 29 22 8 21 37 11 86 87 61 51 70 79 0	110	85	116.00
15	0 16 35 1 26 5 18 33 36 83 60 65 85 55 81 94 0	112	104	147.00
11	0 28 13 20 34 12 38 78 52 77 56 63 0	111	79	121.00
15	0 4 24 25 10 6 32 95 66 82 68 73 54 74 91 76 0	101	111	183.00
$n = 100$, $n_l = 50$, $n_b = 50$		$Q = 112$	923	

References

Anily, S., 1996. The vehicle-routing problem with delivery and back-haul options. Naval Research Logistics 43, 415–434.

Brandao, J., 2006. A new tabu search algorithm for the vehicle routing problem with backhauls. European Journal of Operational Research 173, 540–555.

Bräysy, O., Dullaert, W., 2003. A fast evolutionary metaheuristic for the vehicle routing problem with time windows. International Journal on Artificial Intelligence Tools 12, 153–172.

Bullnheimer, B., Hartl, R.F., Strauss, C., 1999. An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research 89, 319–328.

Doerner, K., Gronalt, M., Hartl, R.F. Reimann, M., Strauss, C., Stummer M., 2002. Savings ants for the vehicle routing problem. In: Applications of Evolutionary Computing, Proceedings, pp. 11–20.

Dongarra, J.J., 2006. Performance of various computers using standard linear equation software. University of Tennessee Computer Science, Technical Report, CS-89-85.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernatics – Part B 26, 29–41.

Duhamel, C., Potvin, J.Y., Rousseau, J.M., 1997. A tabu search heuristic for the vehicle routing problem with backhauls and time windows. Transportation Science 31, 49–59.

Gambardella, L.M., Taillard, E., Agazzi, G., 1999. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw-Hill, pp. 63–76.

Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. Management Science 40, 1276–1290.

Goetschalckx, M., Jacobsblecha, C., 1989. The vehicle-routing problem with backhauls. European Journal of Operational Research 42, 39–51.

Goetschalckx, M., Jacobsblecha, C., 1993. The vehicle routing problem with backhauls: Properties and solution algorithms. Georgia Institute of Technology, Technical Report, MHRC-TR-88-13.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., 1989. Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning. Operations Research 37, 865–892.

Kindervater, G.A.P., Savelsbergh, M.W.P., 1997. Vehicle routing: Handling edge exchanges. In: Aarts, E.H., Lenstra, J.K. (Eds.), Local Search in Combinatorial Optimization. John Wiley & Sons, Chichester, UK, pp. 37–360.

Lin, S., 1965. Computer solutions to the travelling salesman problem. Bell System Technology Journal 44, 2245–2269.

Mingozzi, A., Giorgi, S., Baldacci, R., 1999. An exact method for the vehicle routing problem with backhauls. Transportation Science 33, 315–329.

Osman, I.H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research 41, 421–451.

Osman, I.H., Wassan, N.A., 2002. A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. Journal of Scheduling 5, 263–285.

Potvin, J.-Y., Duhamel, C., Guertin, F., 1996. A genetic algorithm for vehicle routing with backhauling. Applied Intelligence 6, 345–355.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 31, 1985.

Rego, C., Roucairol, C., 1996. A parallel tabu search algorithm using ejections chain for the vehicle routing problem. In: Osman, I.H., Kelly, J. (Eds.), Meta-Heuristics: Theory and Applications. Kluwer, pp. 661–675.

Reimann, M., Doerner, K., Hartl, R.F., 2002. Insertion based ants for vehicle routing problems with backhauls and time windows. Lecture Notes in Computer Science, 135–148.

Reimann, M., Doerner, K., Hartl, R.F., Ants, D., 2004. Savings based ants divide and conquer the vehicle routing problem. Computers and Operations Research 31, 563–591.

Rochat, Y., Taillard, É., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167.

Ropke, S., Pisinger, D., 2006. A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research 171, 750.

Silvano, M., Paolo, T., 1990. Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons Inc., pp. 296.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 31, 170–186.

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025

Thangiah, S.R., Potvin, J.Y., Sun, T., 1996. Heuristic approaches to vehicle routing with backhauls and time windows. Computers and Operations Research 23, 1043–1058.

Toth, P., Vigo, D., 1996. A heuristic algorithm for the vehicle routing problem with backhauls. In: Bianco, L., Toth, P. (Eds.), . In: Advanced Method in Transportation Analysis. Springer-Verlag, Berlin, pp. 585–608.

Toth, P., Vigo, D., 1997. An exact algorithm for the vehicle routing problem with backhauls. Transportation Science 31, 372–385.

Toth, P., Vigo, D., 1999. A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. European Journal Of Operational Research 113, 528–543.

Toth, P., Vigo, D., 2001. An overview of vehicle routing problems. In: Toth, P., Vigo, D. (Eds.), The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1–26.

Van Breedam, A., 1995. Improvement heuristics for the vehicle routing problem based on simulated annealing. European Journal of Operational Research 86, 480–490.

Wade, A.C., Salhi, S., 2002. An investigation into a new class of vehicle routing problem with backhauls. Omega 30, 479–487.

Wade, A.C., Salhi, S., 2004. An ant system algorithm for the mixed vehicle routing problem with backhauls. Metaheuristics: Computer Decision-making. Kluwer Academic Publishers, pp. 699–719.

Wassan, N., 2004. Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls Canterbury Business School, University of Kent, Working Paper No. 56.

Yano, C., Chan, T., Richter, L., Cutler, T., Murty, K., McGettigan, D., 1987. Vehicle routing at quality stores. Interfaces 17, 52–63.

Please cite this article in press as: Gajpal, Y., Abad, P.L., Multi-ant colony system (MACS) for a vehicle routing problem ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.02.025