APPROACHES FOR THE VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS DELIVERIES AND PICKUPS

Jeng-Fung Chen
Department of Industrial Engineering and Systems Engineering
Feng Chia University
100 Wenhwa Road, Taichung, Taiwan 40724

ABSTRACT

The vehicle routing problem with backhauls deals with the delivery and pickup of goods at different customer locations. In many practical situations, however, the same customer may require both a delivery of goods from the central depot and a pickup of recycled/outdated items. In this paper, a parallel-insertion procedure is presented to obtain a quick solution for the vehicle routing problem with deliveries and pickups. A hybrid heuristic based on the simulated annealing method, tabu lists, and route improvement procedures is also proposed to improve the obtained solution. Computational characteristics of the proposed insertion-based procedure and the hybrid heuristic are evaluated through computational experiments. Computational experiments show that the parallel-insertion procedure obtains better solutions than those found in the open literature. Computational results also indicate that the proposed hybrid heuristic is able to reduce the gap between the obtained initial solution and optimal solution effectively and outperforms a metaheuristic tested in terms of solution quality, and is capable of obtaining optimal solutions very efficiently for small-scaled problems.

Keywords: simulated annealing, tabu list, vehicle routing problem

1. INTRODUCTION

The vehicle routing problem with backhauls (VRPB) is an extension of the classical vehicle routing problem (VRP). The VRP deals with servicing a set of delivery customers/a set of pickup customers by a set of vehicles stationed at a central depot. The objective of the VRP is to design a set of routes to service all customers while minimizing total travel distance. Furthermore, the capacity of each vehicle must not be violated. In the VRPB, the customers are divided into two subsets. The first subset is the linehaul customers, each requiring a delivery of goods from the central depot. The second subset is the backhaul customers, each requiring a pickup of inbound/recycled goods. The critical assumption of the VRPB was that all deliveries had to be made before any pickups. This assumption was caused by the fact that the in the past vehicles were rear-loaded and rearrangement of delivery loads onboard in order to accommodate new pickup loads was difficult. In modern days, however, most of the vehicles are designed with both rear-loaded and side-loaded functions. Rearranging delivery loads onboard to accommodate new pickup loads is no more necessary. Therefore, the assumption that all deliveries must be made before any pickups can now be relaxed. This results in the vehicle routing problem with mixed-load (VRPM) in which the deliveries and pickups can occur in any sequence.

Although the VRPM allows pickups before completing all deliveries, however the model does not accept the vehicle to deliver and pick up goods at the same customer location simultaneously. When a customer has both a delivery and pickup demand, the VRPM model will cause this customer to be visited twice and result in inconvenience and higher transportation costs. This research perceives the feasibility of a mixed load of deliveries and pickups in the same vehicle and the necessity of a delivery and pickup at the same customer location. The objective of the problem is to minimize the total travel distance to service all customers, each with a single stop, without violating the capacity the vehicles. This type of problem is called the vehicle routing problem with simultaneous deliveries and pickups (VRPSDP). The application of VRPSDP is frequently encountered in the distribution system of grocery store chains. Each grocery store may have both a delivery (e.g., fresh food/soft drink) and pickup (e.g., outdated/recycled items) demand and is serviced with a single stop. The foundry industry is another example [8]. Collection of used sand and delivery of purified reusable sand at the same customer location are also serviced with a single stop.

The VRPSDP is NP-hard [8]. In this paper, a parallel-insertion procedure is presented to obtain a quick solution for the VRPSDP. A hybrid heuristic

based on the simulated annealing method (SA), tabu lists, and route improvement procedures is also proposed to improve the obtained solution. Computational characteristics of the proposed procedure and hybrid heuristic are evaluated through computational experiments.

The rest of this paper is organized as follows: previous studies are reviewed in Section 2; the SA and tabu lists are briefly discussed in Section 3; the proposed insertion procedure and hybrid heuristic are described in Section 4; computational experiments are reported in Section 5; concluding remarks are discussed in Section 6.

2. PREVIOUS STUDIES

If either the delivery or pickup demand of each customer equals zero, the VRPSDP is reduced to the VRPM. If all deliveries must be made before any pickups, the VRPM is reduced to the VRPB. Hence, the previous studies on the VRPB and VRPM are also briefly reviewed. For the VRPB, Deif and Bodin [7] proposed an extension of the Clarke-Wright VRP heuristic, in which the saving of connecting linehauls to backhauls is modified such that the formation of mixed routes is delayed. Goetschalckx and Jacobs-Blecha [12] developed a spacefilling-curve heuristic to obtain an initial solution which was then improved by using search heuristics. Potvin *et al*. [24] proposed a genetic algorithm to identify orderings that produce good routes and presented a greedy route-construction procedure to insert customers one by one into the routes for a given ordering of customers. A two-phase heuristic to the VRPB with time windows was proposed by Thangiah *et al*. [26]. In the first phase, a heuristic based on the insertion procedure of Kontoravdis and Bard [16] is applied to obtain initial solutions. In the second phase, the initial solutions are improved through the applications of λ-interchanges [22] and *2-opt** exchanges [23]. Anily [2] derived a lower bound and developed a heuristic based on the *modified circular regional partitioning procedure* of Anily and Federgruen [1]. The routes produced by this procedure result in that backhaul customers are not served until completing service to all linehaul customers. Duhamel *et al*. [10] proposed a tabu search heuristic in which a greedy insertion procedure is applied to obtain an initial solution. The obtained initial solution is then improved by using link or node exchanges. Toth and Vigo [27] proposed a branch-and-bound approach that uses a Lagrangian lower bound strengthened by adding inequalities in a cutting plane fashion. Toth and Vigo [28] also proposed a cluster-first-route-second heuristic in which the clusters are combined by solving an auxiliary assignment problem and the initial routes are built by using a modified traveling salesman problem heuristic. The final set of routes is then obtained by exchanging the inter-route and outer-route arcs. Mingozzi *et al*. [20] developed an exact solution algorithm based on a new integer formulation. They derived a valid lower bound by combining different heuristics for solving the dual of the linear-programming relaxation.

For the VRPM, Golden *et al*. [13] proposed a heuristic based on inserting backhaul customers into the routes formed by linehaul customers. Their insertion criterion includes a penalty factor that considers the number of linehaul customers left on the route. Casco *et al*. [3] presented a load-based insertion heuristic in which the insertion criterion for backhaul customers takes account into the load remained to be delivered on the route. Yano *et al*. [29] dealt with a special case in which each routes can have at most four customers. They applied set covering model to find an optimal set of routes and obtained the optimal routing sequence by using complete enumeration. Dethloff [9] proposed an insertion heuristic based on the concept of residual capacities.

The VRPSDP was first introduced by Min [19]. He presented a cluster-first-route-second algorithm to solve a problem of transporting books between libraries by two vehicles. The libraries are clustered into two groups and two traveling salesman problems are solved to construct routes. If the solution obtained is infeasible, the overloaded arcs are penalized and the traveling salesman problems are resolved. Salhi and Nagy [25] presented a load-based insertion procedure that extends the idea of 1-insertion to cluster insertion. They tested four insertion methods, 1-insertion, 2-insertion, connected-graph-cluster-insertion, and complete-graph-cluster-insertion, and concluded that cluster insertion method gave positive improvement. Dethloff [8] develop an insertion heuristic based on the residual capacity concept. Crispim and Brandão [6] developed a hybrid algorithm which combines the tabu search and modified variable neighborhood descent method. They generated the initial solution based on the sweeping process of the geographic space of customers. The initial solution was then improved by using insertion and exchange procedures.

The concept of an insertion-based procedure is to successively insert customers into growing routes. In each step either one customer or a cluster of customers is inserted. Routes can be constructed sequentially or in parallel. The insertion procedures proposed by Casco *et al*. [3], Salhi and Nagy [25], and Dethloff [8] used the former way. In this paper, an insertion-based procedure which constructs routes in parallel is proposed to obtain a quick solution for the VRPSDP. In addition, an effective hybrid heuristic based on the SA, tabu lists, and route

improvement procedures is also proposed to improve the obtained solution.

3. SA AND TABU LISTS

The concepts of the SA and tabu lists are briefly discussed in this section. First, the SA is described.

3.1 SA method

The SA algorithm was derived from statistical mechanics. Kirkpatrick *et al*. [15] were the first to apply SA to solve combinatorial optimization problems. Since then SA has been broadly applied to solve many large-scale combinatorial optimization problems. SA is a stochastic hill-climbing search algorithm. It consists of a sequence of iterations, each changes the current solution to a new solution in the neighborhood of the current solution. The neighborhood is defined by the chosen generation mechanism. Once a neighborhood solution is created, the corresponding change in the cost function is computed to determine whether the neighborhood solution is accepted as the current solution. If the cost is improved, the neighborhood solution is directly taken as the current solution. Otherwise, it is accepted according to Metropolis's criterion [18]. Based on Metropolis's criterion, the neighborhood solution is accepted as the current solution only if $r \leq exp(-\Delta E/T)$, in which r is generated from $U[0,1]$, ΔE is the difference between the costs of the two solutions, and T is the current temperature. Three parameters need to be specified in designing the cooling schedule: an initial temperature; a temperature update rule; and the number of iterations to be performed at each temperature stage (i.e., the Markov chain length).

3.2 Tabu lists

Since solutions not leading to improvements may be accepted by SA, it is possible to return to previously visited solutions and cause cycling solutions. Hence, tabu lists from the tabu search [11] are used to overcome this problem. The tabu lists store characteristics that classify certain moves as tabu in the later search. By using tabu lists, the paths previously visited are prevented. Theoretically the tabu list needs to store all previously visited solutions. However, this would require too much memory and computational effort. An alternative way is to store only the moves during the last s iterations (s is known as the tabu size). By choosing an appropriate value of s, the likelihood of cycling can be prevented. An aspiration criterion is used to free a tabu solution if it is of sufficient quality and possibly would not cause cycling solutions. Hence, a solution is considered admissible if its attributes are not tabu or it passes the test of aspiration criterion.

Since SA has the advantage of obtaining good solutions and tabu lists have the capability of avoiding getting cycling solution, we hence combine the characteristics of SA and tabu lists in this research to develop an effective heuristic, *HeuVRPSDP*, to resolve the VRPSDP.

4. APPROACHES

The proposed insertion procedure and hybrid heuristic are described in this section. First, the following notations are defined:

Set

N: set of all customer locations

N_0: set of all locations, i.e., $N_0 = N \cup \{0\}$, in which location 0 is the central depot

Parameters

C_{ij}: distance between locations $i \in N_0$ and $j \in N_0$, $C_{ij} = C_{ji}$, $C_{00} = 0$, $C_{ii} = \infty$

D_j: delivery demand of customer $j \in N$

P_j: pickup demand of customer $j \in N$

Q: vehicle capacity

4.1 Initial solution

An insertion-based procedure, *P_INS*, which constructs routes in parallel is proposed to obtain a quick solution. This procedure is initiated by choosing one customer to serve as the seed customer of the first route. For all unrouted customers, the values of the insertion criterion for all feasible insertion positions are computed. Then, the customer with the best insertion value is inserted to the best insertion position. Note that the feasible insertion positions for an unrouted customer include choosing it to serve as the seed customer for a new route. This route building process is repeated until all customers are routed. Apparently, the insertion criterion is crucial to the solution quality of an insertion-based procedure.

In order to develop an insertion criterion with capability to obtain a good solution, we need to take into account the situation of both a delivery and pickup demand at the same customer location. We expect that the routing position of a customer with a larger net delivery demand is at the front of the route and the routing position of a customer with a larger net pickup demand is at the rear of the route. In addition, the travel distance between any two consecutively visited customers should not be too far away. Let D_{max} be the maximum distance between all pairs of nodes (*i.e.*, $D_{\max} = \max_{i,j \in N_0}\{C_{ij}\}$), $S(k)$ be the predecessors of customer k, SD_k be the sum of the net delivery amounts of customers in $S(k)$ (i.e.,

$SD_k = \sum_{s \in S(k)} (D_s - P_s)$, ISD_k be SD_k plus the net delivery amount of customer k. Then given that inserting customer k between i and j is a feasible route, the proposed insertion criterion is given as follows:

$$\cos t_k(i,j) = \begin{cases} (C_{ik} + C_{kj} - C_{ij}) - 2C_{ok} - \alpha(D_k - P_k)ISD_k / Q + MY & \text{if } D_k \geq P_k \\ (C_{ik} + C_{kj} - C_{ij}) - 2C_{ok} - \beta(P_k - D_k)SD_k / Q + MY & \text{otherwis e,} \end{cases} \quad (1)$$

in which M is a big number; $Y = 0$ if $C_{ik} \leq D_{max}$ and $C_{kj} \leq D_{max}$, and 0 otherwise. In Equation (1), (C_{ik} + C_{kj} - C_{ij}) is the extra travel distance when inserting customer k between customers i and j; $2C_{0k}$ is the distance bonus for inserting customer k on the route, $\alpha(D_k–P_k)ISD_k/Q$ and $\beta(P_k–D_k)SD_k/Q$ are capacity bonus when inserting customer k between customers i and j, MY is the distance penalty. According to Equation (1), a customer with a larger net delivery/pickup amount will have a better chance to be selected early in the insertion process, given that the distance between i and k and between j and k is not too far away. The routing position of the customer with a larger net delivery will be at the front of the route while the routing position of the customer with a larger net pickup amount will be at the rear of the route, such that a vehicle can serve more customers on a route and the travel distance between ay two consecutively visited customers is not too far away.

4.2 Neighborhood solutions

Several search methods that have been proved to be very effective in resolving the VRP and VRPB are applied in the solution approach of the VRPSDP. The *2-opt* and *Or-opt* methods are applied for within-route improvement while the *2-exchange* procedure and *shift/swap move* are applied for between-route improvement.

2-opt

2-opt was introduced by Lin [17]. In this method, two links not adjacent to each other are removed from the same route and the segments are reconnected in all possible ways.

Or-opt

Or-opt was introduced by Or [21]. In this method, a sequence of three consecutive customers, two consecutive customers, or a single customer on a route is removed and inserted to another position in the same route.

2-exchange

2-exchange method was introduced by Potvin *et al*. [23]. It is an extension of the *2-opt* method [17] for problems with multiple vehicles. In this method, two links are removed from two different routes and two new links are produced by connecting the first customer on the first link to the last customer on the second link and connecting the first customer on the second link to the last customer on the first link.

Shift/swap move

Shift/swap move is based on customer interchanging between vehicle routes. For a given solution represented by the set of routes {R_1, R_2, …, R_m, …, R_n, …}, in which R_m is the set of customers serviced by vehicle m, a *shift/swap move* between routes R_m and R_n is to select two sequences of consecutive customers in R_m and R_n and to exchange these two sequences. A *shift/swap move* results in either the chosen sequence being shifted from one route to another or the chosen sequences being exchanged between two routes. For example, operator *(2, 1)* results in the sequence of two consecutive customers on one route is exchanged with the sequence of one customer on another route; operators *(0, 1)*, and *(2, 0*) result in a shift of one or two customers from one route to another.

4.3 Heuristic *HeuVRPSDP*

The proposed hybrid heuristic, *HeuVRPSDP*, is now outlined as follows:

Step 0. Set the initial temperature (T), Markov chain length, tabu size, and stopping criterion.
Step 1. Generate an initial solution based on the proposed insertion procedure.
Step 2. Perform *2-exchange* procedure until the Markov chain length is reached.
Step 3. Perform *or-opt* procedure until the Markov chain length is reached.
Step 4. Perform *(2, 0) shift move* until the Markov chain length is reached.
Step 5. Perform *or-opt* procedure until the Markov chain length is reached.
Step 6. Perform *(2, 1) swap move* until the Markov chain length is reached.
Step 7. Perform *or-opt* procedure until the Markov chain length is reached.
Step 8. Perform *(2, 2) swap move* until the Markov chain length is reached.
Step 9. Perform *or-opt* procedure until the Markov chain length is reached.
Step 10. Perform *(1, 0) shift move* until the Markov

chain length is reached.

Step 11. Perform *or-opt* procedure until the Markov chain length is reached.

Step 12. Perform *(1, 1) swap move* until the Markov chain length is reached.

Step 13. Perform *or-opt* procedure until the Markov chain length is reached.

Step 14. Perform *2-opt* procedure until the Markov chain length is reached.

Step 15. Update the temperature. If the stopping criterion is reached, stop the whole procedure. Otherwise, return to Step 2.

The logic of *HeuVRPSDP* is to apply a between-route improvement procedure whenever a within-route improvement procedure is performed and vice versa. Since *2-exchange* procedure can exchange more customers in each iteration, it is applied before *shift*/*swap move*s. Note that although *2-opt* may be more effective than *or-opt*, however it requires more computational efforts. Therefore, *or-opt* is performed between every two between-route improvement procedures. *2-opt* is only applied at the end of each outer loop to make further improvement.

5. COMPUTATIONAL RESULTS

In this section, two sets of test problems are used to evaluate the computational characteristics of *P_INS* and *HeuVRPSDP*. All procedures described in this research were coded in C language and all computational experiments were conducted on a Pentium IV 1.6 GHz PC. For *P_INS*, α and β were set at 0.3 and 0.4, respectively. Dynamic Markov chain lengths were used in *HeuVRPSDP*. That is, if the best solution was not improved in six consecutive iterations, the inner loop formed by the Markov chain was terminated and the temperature was updated. The initial temperature was set at 100; the size of each tabu list was set at 7; the cooling rate was set at 0.9; and the heuristic was terminated when the best solution was not improved in ten consecutive temperatures. The parameters suggested above were determined after preliminary testing considering both solution quality and computational efficiency. Note that according to our computational experiences, it did not make much difference to the solution quality and runtime if the initial temperature was changed from 100 to 90 or 110, the cooling rate was changed from 0.9 to 0.85 or 0.95, and the size of tabu list was changed from 7 to 5 or 10. However, the solution quality was sensitive to the Markov chain length. If the Markov chain length was reduced, the size of outer loop (i.e., the maximum number of consecutive temperatures without improvement to the best-known solution) needed to be increased.

The first set of test problems of sizes 50, 75, 100, 120, 150, and 199 was taken from the literature [25]. These problems were derived from Christofides *et al*. [5] to generate the delivery-and-pickup case. For each customer j, a ratio $r_j = \min\{(x_j / y_j), (y_j / x_j)\}$ was first computed, where x_j and y_j are the coordinates of customer j. The delivery [pickup] amount of customer j was then set to be $r_j \times t_j$ [$(1 - r_j) \times t_j$], where t_j was the original (delivery) demand. These test problems are referred to as the *X* type. The problems of the *Y* type were then obtained by exchanging the delivery and pickup amount of every other customer.

Table 1 shows the results obtained by the four procedures of Salhi and Nagy [25] and *P_INS*. According to Table 1, the performance of *P_INS* is quite impressive. *P_INS* obtained better solutions than the best solutions of Salhi and Nagy in 10 out of 14 test problems. Although on an overall average the improvement of the total travel distance made by *P_INS* was only 7.50%, however the number of vehicles required was reduced by 48.78%.

The computational comparisons of *P_INS* and *HeuVRPSDP* are given in Table 2. The CPU time required for *HeuVRPSDP* ranged from a few seconds to near 13 minutes. On an overall average, the total travel distance obtained by *P_INS* was reduced by 22.31%. This result indicates that *HeuVRPSDP* is able to reduce the gap effectively between the obtained initial solution and the optimal solution.

Table 3 shows the computational comparisons of the metaheuristic by Crispim and Brandão [6] and *HeuVRPSDP*. According to Table 3, *HeuVRPSDP* obtained better solution than the metaheuristic by Crispim and Brandão in 12 out of 14 test problems. Both heuristics obtained almost the same solutions for the other two test problems. On an overall average, the solution obtained by the metaheuristic of Crispim and Brandão was reduced 7.95% by *HeuVRPSDP*. Although *HeuVRPSDP* obtained better solution than the metaheuristic by Crispim and Brandão, however, it required more CPU time.

In order to further test the computational characteristics of *HeuVRPSDP*, a set of 25 test problems in small sizes was used. These problems of sizes 15, 17, and 20 were derived by Chen and Wu [4] from problems R121, R141, R161, R181, and R1101 in the extended Solomon data set [14] by considering the first 15, 17, and 20 customers and setting the vehicle capacity at 80 and 120. To incorporate a delivery and pickup demand at the same customer, the scheme used by Salhi and Nagy [25] was applied to generate test problems of the *X* type. The solutions obtained by *HeuVRPSDP* were compared with optimal solutions obtained by CPLEX.

Table 1. Computational comparisons of Salhi and Nagy [25] and *P_INS*

Problem no.	No. of customers	Vehicle capacity	Salhi and Nagy [25]								*P_INS*	
			1-INS		*2-INS*		*SC-INS*		*LC-INS*			
			Routing cost	No. of vehicles	Routing cost	No. of vehicles	Routing cost	No. of vehicles	Routing cost	No. of vehicles	Routing cost	No. of vehicles
CMT1X	50	160	601	6	601	6	601	6	601	6	564 (6.16%)[1]	3 (50.00%)[2]
CMT1Y	50	160	603	5	603	5	603	5	603	5	549 (9.00%)	3 (40.00%)
CMT2X	75	140	873	10	873	10	903	11	903	11	789 (9.57%)	6 (40.00%)
CMT2Y	75	140	924	12	924	12	924	12	924	12	904 (2.22%)	6 (50.00%)
CMT3X	100	200	923	10	923	10	923	10	923	10	859 (6.96%)	5 (50.00%)
CMT3Y	100	200	923	10	923	10	923	10	923	10	886 (4.00%)	5 (50.00%)
CMT4X	150	200	1178	15	1178	15	1178	15	1178	15	1204 (-2.18%)	8 (46.67%)
CMT4Y	150	200	1178	15	1178	15	1178	15	1178	15	1109 (5.86%)	7 (53.33%)
CMT5X	199	200	1509	19	1509	19	1509	19	1509	19	1358 (10.03%)	10 (47.37%)
CMT5Y	199	200	1477	19	1477	19	1477	19	1477	19	1521 (-3.00%)	10 (47.37%)
CMT11X	120	200	1546	11	1530	11	1500	11	1500	11	1136 (24.30%)	5 (54.55%)
CMT11Y	120	200	1557	11	1557	11	1500	11	1500	11	1011 (32.59%)	4 (63.64%)
CMT12X	100	200	853	10	853	10	820	10	831	10	940 (-14.68%)	6 (40.00%)
CMT12Y	100	200	888	11	888	11	888	11	873	11	937 (-7.35%)	6 (45.45%)
Overall average			1073.77	11.71	1072.64	11.71	1066.21	11.79	1065.93	11.79	983.36 (7.50%)	6.00 (48.78%)

[1]The best routing cost obtained from Salhi and Nagy [25] improved by *P_INS*
[2]The number of vehicles reduced by *P_INS*

The computational results for the small-sized test problems are given in Table 4. CPLEX did not obtain the optimal solutions for 5 test problems due to running out of memory (these solutions are marked with "-"). For the other tested problems, the runtime required for CPLEX to obtain the optimal solution ranged from 354 seconds to more than 65 hours. In comparisons, it only took 0.35 to 1.47 seconds for *HeuVRPSDP* to obtain the optimal solution. These results indicate that *HeuVRPSDP* is able to obtain the optimal solutions for problems in small sizes very efficiently.

6. CONCLUDING REMARKS

This research has dealt with the vehicle routing problem with simultaneous deliveries and pickups. An insertion-based procedure (i.e., *P_INS*) constructing routes in parallel has been presented to obtain a quick solution. An effective hybrid heuristic (i.e., *HeuVRPSDP*) based on the SA, tabu lists, and improvement procedures has also been proposed to improve the obtained solution. Two sets of problems have been used to test the computational characteristics of *P_INS* and *HeuVRPSDP*. Computational results have shown that both the *P_INS* and *HeuVRPSDP* are capable of obtaining solutions in good quality. *P_INS* obtained better solution than the best solutions of Salhi and Nagy [25] in 10 out of 14 test problems. On an overall average, the total travel distance and the number of vehicles required were reduced by 7.50% and 48.78%, respectively. *HeuVRPSDP* is able to reduce the gap between the obtained initial solution and optimal solution effectively and outperforms a metaheuristic by Crispim and Brandão [6] in terms of solution quality, and is capable of obtaining optimal solutions very efficiently for small-scaled problems.

The solutions obtained by *P_INS* and

HeuVRPSDP can provide the benchmarks for this research problem. As for future research, taking into account that the customer imposes strict earliest and latest time deadline (i.e., service time windows) is an important issue for future research to pursue.

Table 2. Computational comparisons of *P_INS* and *HeuVRPSDP*

Problem no.	No. of customers	Vehicle capacity	*P_INS*		*HeuVRPSDP*		
			Routing cost	No. of vehicles	Routing cost	No. of vehicles	CPU time[1]
CMT1X	50	160	564	3	478 (15.25%)[2]	3	6.23
CMT1Y	50	160	549	3	481 (12.39%)	3	6.75
CMT2X	75	140	789	6	689 (12.67%)	6	21.78
CMT2Y	75	140	904	6	679 (24.89)	6	11.63
CMT3X	100	200	859	5	745 (13.27%)	5	83.77
CMT3Y	100	200	886	5	723 (18.40%)	5	112.98
CMT4X	150	200	1204	8	867 (27.99%)	7	469.78
CMT4Y	150	200	1109	7	852 (23.17%)	7	417.44
CMT5X	199	200	1358	10	1067 (21.43%)	10	775.83
CMT5Y	199	200	1521	10	1046 (31.23%)	10	731.71
CMT11X	120	200	1136	5	859 (24.38%)	4	309.72
CMT11Y	120	200	1011	4	860 (14.94%)	4	221.64
CMT12X	100	200	940	6	676 (28.09%)	6	44.78
CMT12Y	100	200	937	6	673 (28.18%)	5	53.67
Overall average			994.94	6.00	763.93 (22.31%)	5.79 (3.57%)	

[1]The runtime is in seconds
[2]The routing cost obtained from *P_INS* improved by *HeuVRPSDP*

Table 3. Computational comparisons of Crispim and Brandão [6] and *HeuVRPSDP*

Problem no.	No. of customers	Vehicle capacity	Crispim and Brandão [6]			*HeuVRPSDP*		
			Routing cost[1]	No. of vehicles	CPU time[2]	Routing cost[1]	No. of vehicles	CPU Time[2]
CMT1X	50	160	477	3	11.2	478 (-0.21%)[3]	3 (0.00%)[4]	6.23 (44.38%)[5]
CMT1Y	50	160	485	3	9.1	481 (0.82%)	3 (0.00%)	6.75 (25.82%)
CMT2X	75	140	710	6	24.3	689 (2.96%)	6 (0.00%)	21.78 (10.37%)
CMT2Y	75	140	715	6	26.4	679 (5.03)	6 (0.00%)	11.63 (55.95%)
CMT3X	100	200	744	5	37.1	745 (-0.13%)	5 (0.00%)	83.77 (-125.80%)
CMT3Y	100	200	742	5	33.5	723 (2.56%)	5 (0.00%)	112.98 (-237.25%)
CMT4X	150	200	915	7	58.1	867 (5.25%)	7 (0.00%)	469.78 (-708.57%)
CMT4Y	150	200	996	7	47.6	852 (14.46%)	7 (0.00%)	417.44 (-776.97%)
CMT5X	199	200	1136	10	89.4	1067 (6.07%)	10 (0.00%)	775.83 (-767.82%)
CMT5Y	199	200	1129	10	77.1	1046 (7.35%)	10 (0.00%)	731.71 (-849.04%)
CMT11X	120	200	944	4	32.4	859 (9.00%)	4 (0.00%)	309.72 (-855.93%)
CMT11Y	120	200	1035	4	29.6	860 (16.91%)	4 (0.00%)	221.64 (-648.78%)
CMT12X	100	200	731	5	37.1	675 (7.52%)	6 (-20.00%)	44.78 (-20.70%)
CMT12Y	100	200	860	5	33.7	673 (21.74%)	5 (0.00%)	53.67 (-59.26%)
Overall average			829.93	5.71	39.04	763.93 (7.95%)	5.79 (-1.33%)	233.41 (-497.82%)

[1]The solutions are the best available solutions which are not necessarily the optimal solutions
[2]The runtime is in seconds
[3]The routing cost obtained from Crispim and Brandão [6] improved by *HeuVRPSDP*
[4]The number of vehicles reduced by *HeuVRPSDP*
[5]The CPU time reduced by *HeuVRPSDP*

Table 4. Computational comparisons of *HeuVRPSDP* and optimal solutions

Problem no.	No. of customers	Vehicle capacity	Optimal solution			*HeuVRPSDP*		
			Routing cost	No. of vehicles	CPU Time[1]	Routing cost	No. of vehicles	CPU time[1]
R121	15	80	611	3	259234	611	3	0.35
R141			750	3	11940	750	3	0.42
R161			1167	2	27122	1167	2	0.43
R181			1968	2	234229	1968	2	0.79
R1101			2034	2	3407	2034	2	0.86
R121		120	542	2	76708	542	2	0.51
R141			670	2	3821	670	2	0.60
R161			1163	2	14914	1163	2	0.60
R181			1756	2	206703	1756	2	0.51
R1101			1810	2	422	1810	2	0.42
R121	17	80	-	-	-	743	3	0.82
R141			791	3	7426	791	3	0.45
R161			1211	3	95788	1211	3	0.61
R181			-	-	-	1992	3	0.50
R1101			-	-	-	2296	2	0.83
R121		120	564	2	86401	564	2	0.84
R141			758	2	28456	758	2	0.66
R161			1193	2	5442	1193	2	0.69
R181			1787	2	354	1787	2	0.79
R1101			2052	2	2828	2052	2	0.43
R121	20	120	-	-	-	624	2	0.87
R141			798	2	138006	798	2	1.02
R161			1280	2	123550	1280	2	0.56
R181			1866	2	21438	1866	2	1.47
R1101			-	-	-	2119.53	2	1.24

[1]The runtime is in seconds

ACKNOWLEDGEMENT

This material is based on work supported by the National Science Councils on Grant number NSC 91-2213-E-035-040.

REFERENCES

1. Anily, S. and A. Federgruen, "A class of Euclidean routing problems with general route costs functions," *Management Science*, **36**, 92-114 (1990).
2. Anily, S., "The vehicle-routing problem with delivery and backhaul options," *Naval Research Logistics*, **43**, 415-434 (1996).
3. Casco, D. O., B. L. Golden and E. A. Wasil, "Vehicle routing with backhauls: models, algorithms, and case studies." In L. Golden and A. Assad (Eds.), *Vehicle Routing, Methods and Studies*, Amsterdam, North-Holland, 127-147 (1988).
4. Chen, J.-F., and T.-H. Wu, "Vehicle routing problem with simultaneous deliveries and pickups." *To appear in Journal of the Operational Research Society*, (2005)
5. Christofides, N., A. Mingozzi and P. Toth, "The vehicle routing problem," In N. Christofides, A. Mingozzi, P. Toth and C. Sandi (Eds.), *Combinatorial Optimization*, Chichester, Wiley, 315-338 (1979).
6. Crispim, J. and J. Brandão, "Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls," *Journal of the Operational Research Society*, 1-7 (2005).
7. Deif, I. and L. Bodin, "Extension of the Clark and Wright algorithm for solving the vehicle routing problem with backhauling," *Proceedings of the Conference on Computer Software Uses in Transportation and Logistics Management*, Babson Park, USA, 75-96 (1984).
8. Dethloff, J., "Vehicle routing and reverse logistics; the vehicle routing problem with simultaneous delivery and pickup," *OR Spektrum*, **23**, 79-96 (2001).

9. Dethloff, J., "Relation between vehicle routing problems: an insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls," *Journal of the Operational Research Society*, **53,** 115-118 (2002).
10. Duhamel, C., J.-Y. Potvin and J.-M. Rousseau, "A tabu search heuristic for the vehicle routing problem with backhauls and time windows," *Transportation Science*, **31**, 49-59 (1997).
11. Glover, F., "Tabu search-part I," *ORSA Journal on Computing*, **1**, 190-206 (1989).
12. Goetschalckx, M. and C. Jacobs-Blecha, "The Vehicle routing problem with backhauls," *European Journal of Operational Research*, **42**, 39-51 (1989).
13. Golden, B., E. Baker, J. Alfaro and J. Schaffer, "The vehicle routing problem with backhauling: two approaches," *Proceedings of the Twenty-First Annual Meeting of S.E.TIMS*, South Carolina, USA, 90-92 (1985).
14. Homberger, J. and H. Gehring, "Two evolutional metaheuristics for the vehicle routing problem with time windows," *INFOR Journal*, **37**, 297-318 (1999).
15. Kirkpatrick, S., C. Gelatt and M. Vecchi, "Optimization by simulated annealing," *Science*, **220**, 671-680 (1983).
16. Kontoravdis, G. and J. Bard, "Improved heuristics for the vehicle routing problem with time windows," *Working Paper*, The University of Texas at Austin, USA, (1992).
17. Lin, S., "Computer solutions of the TSP," *Bell System Technical Journal*, **44**, 2245-69 (1965).
18. Metropolis, N., A. Rosenbluth and A. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, **21**, 1087-1092 (1953).
19. Min, H., "The multiple vehicle routing problem with simultaneous delivery and pick-up Points," *Transportation Research*, **23A**, 377-86 (1989).
20. Mingozzi, A., S. Giorgi and R. Baldacci, "Exact method for the vehicle routing problem with backhauls," *Transportation Science*, **33**, 315-29 (1999).
21. Or, I., "Traveling salesman-type combinational problems and their relation to the logistics of blood banking," *Disertation*, Northwestern University, USA, (1976).
22. Osman, I. H. and N. Christofides, "Capacitated clustering problem by hybrid simulated annealing and tabu search," *International Transactions in Operational Research*, **41**, 421-51 (1994).
23. Potvin, J. Y., T. Kervahut, B. L. Garcia and J. M. Rousseau, "A tabu search heuristic for the vehicle routing problem with time windows." *Technical Report CRT-855*, Universite de Montreal, Canada, (1992).
24. Potvin, J. Y., G. Duhamel and F. Guertin, "A genetic algorithm for vehicle routing with backhauling," *Applied Intelligence*, **6**, 345-55 (1996).
25. Salhi, S. and G. Nagy, "A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling," *Journal of the Operational Research Society*, **50**, 1034-42 (1999).
26. Thangiah, S. R., J. Y. Potivn and T. Sun, "Heuristic approaches to vehicle routing with backhauls and time windows." *Computers & Operations Research*, **23**, 1043-57 (1996).
27. Toth, P. and D. Vigo, "An exact algorithm for the vehicle routing problems with backhauls," *Transportation Science*, **113**, 528-43 (1997).
28. Toth, P. and D. Vigo, "A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls," *European Journal of Operational Research*, **113**, 528-43 (1999).
29. Yano, C., T. Chan, L. Richter, T. Cutler, K. Murty and D. McGettigan, "Vehicle routing at quality stores," *Interfaces*, **17**, 52-63 (1987).

ABOUT THE AUTHOR

Jeng-Fung Chen is an Associate Professor in the Department of Industrial Engineering and Systems Management at Feng Chia University, TAIWAN. He received his Ph.D. degree in Industrial Engineering from Texas A&M University. His research interests are focused on scheduling, manufacturing systems, supply chain management, and OR applications.

(Received April 2005; revised June 2005; accepted July 2005)

同時送收貨之車輛途程問題的演算法

陳正芳*
逢甲大學工業工程與系統管理系
40724 台中市文華路 100 號

摘要

回程取貨的車輛途程問題乃是在處理對不同顧客點的送貨或取貨。但在很多實務的情況中，同一顧客點可能會同時有送貨和取貨的需求。本研究對同時送收貨的車輛途程問題提出一個插入法爲基的求解程序，並發展結合模擬退火法和禁忌名單的混合演算法來改善插入法所獲得的解。我們以文獻的問題對發展的平行插入法及混合演算法進行測試。測試結果顯示，我門的平行插入法優於文獻的插入法程序，混合演算法可有效地縮小平行插入法所獲得的解和最佳解的差距、可獲得優於文獻之萬用啓發式解法所求的解，且能很有效率地求得測試之小問題的最佳解。

關鍵詞：模擬退火法，禁忌名單，車輛途程
(*聯絡人: jfchen@fcu.edu.tw)