Expert Systems with Applications 39 (2012) 5641–5648

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# New best solutions to VRPSPD benchmark problems by a perturbation based algorithm

Yeowoon Jun, Byung-In Kim *

*Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Kyungbuk 790-084, Republic of Korea*

ARTICLE INFO

*Keywords:*
VRPSPD
Vehicle routing problem
Simultaneous pickup and delivery
Sweep algorithm
Perturbation

ABSTRACT

This paper discusses the vehicle routing problem with simultaneous pickup and delivery (VRPSPD), in which both pickup and delivery tasks simultaneously occur at various customer locations. The objective is to design a set of minimum distance routes with the minimum number of vehicles satisfying all the customers. This paper proposes a heuristic algorithm consisting of a route construction procedure, a route improvement procedure, and a solution perturbation procedure. A new sweep-based route construction method is developed to generate better initial solutions. In the route improvement procedure, a series of inter- and intra-route improvement algorithms are applied. The perturbation procedure perturbs a solution by removing some routes and stops from the solution and reinserting them into the solution. The last procedure is used to escape from local optima. Computational experiments on various benchmark instances are performed to evaluate our algorithm against the previously proposed approaches. New best solutions for many benchmark instances are found.



## 1. Introduction

In the classic vehicle routing problem (VRP), a set of delivery customers with known demands is to be served by a fleet of vehicles from a single distribution center, called a depot. Vehicles have identical capacities, and they all start and end their tour at the depot. The objective is to minimize the total distance traveled by the entire fleet.

The problems that need to be solved in real-life situations are usually much more complicated. For example, customers not only have delivery demands, but also pickup demands. Examples of these real-life problems include the VRP with mixed pickup and delivery, the dial-a-ride problem, and the VRP with simultaneous pickup and delivery (VRPSPD). In this paper, we discuss the VRPSPD.

The VRPSPD is a variation of the classic VRP, in which clients have both delivery and pickup demands, and each customer should be serviced by a single vehicle. Pickup and delivery should be performed simultaneously such that each customer is visited only once by a vehicle. For the VRPSPD in this paper, the objective is to minimize the total distance traveled by the entire fleet subject to the following considerations:

- Each customer should be serviced by a single vehicle.
- Each customer has both delivery and pickup demands. Either demand amount may have a value of 0.
- If a customer has both pickup and delivery demands, the delivery task is serviced before the pickup task.
- All vehicles have the same capacity and the same duration limit.
- At any point of a route, the loaded amount, that is, the sum of the up-to-that-point pickup and future-after-that-point delivery amounts cannot exceed the vehicle capacity.
- The route duration of a vehicle cannot exceed the vehicle duration limit.
- There is a single depot.
- All vehicles start from and end at the depot.

The VRPSPD was first introduced by Min (1989), who studied the book delivery and pick-up activity between a central library and 22 local libraries with a given number of capacitated vehicles. He proposed a cluster-first route-second approach, in which customer locations are clustered first and then a route for each cluster is generated by a traveling salesman problem (TSP) algorithm. Dethloff (2001) pointed out the importance of the VRPSPD in reverse logistics operations. His solution approach is based on a cheapest insertion method. The insertion criterion of the approach takes into account three metrics: travel distance, residual capacity, and radial surcharge. More recently, Nagy and Salhi (2005) proposed four insertion-based heuristics to solve VRPSPD. After constructing partial routes for a set of customers, the remaining customers are repetitively inserted into the partial routes by the

* Corresponding author. Tel.: +82 54 279 2371; fax: +82 54 279 2870.
*E-mail address:* bkim@postech.ac.kr (B.-I. Kim).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.11.053

insertion method. Montane and Galvao (2006) proposed a tabu search with a frequency penalization scheme to diversify the search space for VRPSPD. Zachariadis, Tarantilis, and Kiranoudis (2009) proposed a hybrid solution approach incorporating a tabu search and a guided local search. Ai and Kachivichyanukul (2009) presented a mathematical formulation of the VRPSPD and proposed a particle swarm optimization (PSO) algorithm as a solution approach.

Zachariadis, Tarantilis, and Kiranoudis (2010) proposed a local search metaheuristic solution approach for the VRPSPD. The proposed algorithm is capable of exploring wide solution neighborhoods by statistically encoding moves into special data structures. They also apply the adaptive memory algorithmic framework which collects and combines promising solution features to generate high-quality solutions.

Subramanian, Drummond, Bentes, Ochi, and Farias (2010) proposed a parallel metaheuristic for the VRPSPD, which is a master–worker based parallel model, and has three main parts. The first one estimates the number of vehicles; the second part corresponds to the automatic calibration of the parameter gamma, which is a factor that controls the bonus of inserting clients remotely located from the depot; and the third is the optimization phase. Catay (2010) developed an ant colony algorithm equipped with a new saving-based visibility function and a pheromone update procedure.

As the VRP is a very complex NP-hard problem (Golden, Ball, & Bodin, 1981), solving the real-life VRPs to optimality is often not possible within the limited computing time available in practical situations. Therefore, most of the research available has focused on heuristics and metaheuristic solution methods designed to produce high quality solutions within a given time. When all the pick-up demands of customers are set to 0, VRPSPD becomes VRP. The VRPSPD is NP-hard because its special case is NP-hard. Although it has recently drawn increased attention from researchers, it still requires development of better solution approaches.

In this paper, we propose a heuristic algorithm consisting of a route construction procedure, a route improvement procedure, and a solution perturbation procedure. A new sweep-based route construction method is developed to generate better initial solutions. The perturbation procedure perturbs a solution by removing some routes and stops from the solution and reinserting them into the solution. The last procedure is used to escape from local optima. Computational experiments on various benchmark instances are performed to evaluate our algorithm against previously proposed approaches. New best solutions for many benchmark instances are found.

The remainder of this paper is organized as follows. Section 2 summarizes the main idea of the proposed heuristic algorithm and the improvement heuristics applied within our algorithm. Then we proceed to the description of the main structure of the solution algorithm, the initial solution procedure, improvement procedure and finally the perturbation strategies that are used to improve the initial solution. In Section 3 computational experiments on the benchmark data sets are presented. Section 4 provides conclusions and future research directions.

## 2. The proposed algorithm

Our proposed algorithm generates an initial solution with a sweep-based route construction method, improves the solution using various improvement operators, perturbs the solution by removing and reinserting some routes and stops, and then improves the solution yet again. The last two steps are iterated until a stopping criterion is met. The concept of perturbation is borrowed from the adaptive large neighborhood search (ALNS) of Pisinger and Ropke (2007). The overall procedure of the proposed algorithm is as follows:

**<Proposed algorithm>**

Step 1: Generate multiple initial solutions using the sweep variant construction algorithm and choose the best solution as the current solution.
Step 2: Improve the current solution using various improvement operators.
Set the best solution to be the current solution.
Step 3: Perturb the current solution by removing and reinserting some stops.
Step 4: Improve the current solution using various improvement operators.
If the improved solution is better than the best solution, update the best solution.
Step 5: Set the current solution to be the best solution.
Step 6: Repeat Steps 3–5 until a stopping criterion is met.
Step 7: Return the best solution.

The three sub-procedures will be described in detail below.

### 2.1. A sweep variant for the route construction

The *sweep algorithm* (SWA) was first introduced by Gillett and Miller (1974), who named the idea of Wren (1971) and Wren and Holliday (1972) '*the sweep algorithm*.' This algorithm clusters a group of stops to form a route using polar angles between the depot and the stops. The SWA works as follows:

**<The Sweep Algorithm: SWA>**

Step 0. Calculate the polar angle between each stop and the depot.
Sort stops in increasing order of polar angle (the counter-clockwise direction): $s_1, s_2, s_3, \ldots, s_n$. Set $k = 1$.
Step 1. Insertion order is $s_k, s_{k+1}, s_{k+2}, \ldots, s_n, s_1, s_2, \ldots, s_{k-1}$.
Step 2. Choose an unused vehicle.
Step 3. Continue to assign unrouted stops to the vehicle according to the insertion order and construct a route while the sum of the pickup and delivery demand of stops does not exceed the capacity and duration limit of the vehicle.
Step 4. If an unrouted stop exists, go to Step 2.
Else, go to Step 5.
Step 5. Record a solution.
Step 6. If $k < n$, set $k = k + 1$ and go to Step 1.
Else, go to Step 7.
Step 7. Repeat Step 0–Step 6 with decreasing order of polar angles (the clockwise direction).
Step 8. Return the solution with the minimum travel distance.

The original SWA clusters stops according to their polar angles, and assigns a vehicle to visit all the stops within each cluster. Because the original SWA clusters stops solely by polar angles, widely-separated stops may be grouped into the same cluster if they have similar polar angles (see Fig. 1(b)). It does not consider how a stop is nearly located; it only chooses the next stop with the polar angle order. When the distances between the stops are considered, a clustering similar to Fig. 1(c) is more desirable. With this line of idea, we extend the SWA and propose a sweep variant named *sweep nearest algorithm* (SWNA) for the construction procedure. In SWNA,

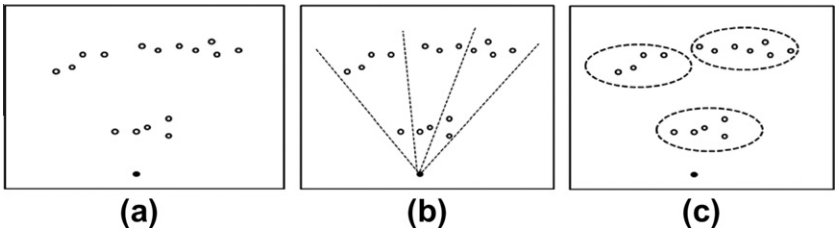


**Fig. 1.** Examples of initial route constructions: (a) initial arrangement, (b) route construction using SWA and (c) route construction using SWNA.

stops are clustered based on the distance between the stops rather than the polar angles. The detailed algorithm is as follows:

**<The Sweep Nearest Algorithm: SWNA>**

| | |
|---|---|
| Steps 0–1: | (Same as Steps 0–1 of the SWA) |
| Step 2: | Choose an unused vehicle.<br>Assign the unrouted stop with the smallest polar angle to the vehicle. |
| Step 3: | Select the nearest stop from the current route.<br>Continue to assign such a nearest stop to the vehicle and construct a route while the sum of pickup and delivery demands does not exceed the capacity and duration limit of the vehicle. |
| Steps 4–8: | (Same as Steps 4–8 of the SWA) |

Because good initial solutions generally result in good final solutions after application of improving algorithms, we attempt to find a better initial solution using various reference points. In a sweep-based route construction method, the stops (or customer locations) are somehow ordered first. Ordering can easily be obtained by numbering the stops in increasing order of polar-angle between the line linking the stop to a reference point and an arbitrary axis passing through the chosen reference point. An equation to evaluate the corresponding polar-angle is defined as follows:

$$polarAngle(i) = \tan^{-1}\left(\frac{y(i)-y(0)}{x(i)-x(0)}\right)$$

where $polarAngle(i)$, $x(i)$, and $y(i)$ are respectively the polar-angle, $x$-coordinates, and $y$-coordinates of customer location $i$, and 0 represents the reference point.

Whereas the original SWA uses the depot as the only reference point, we propose to also use other reference points. Below, distant points $x^+$, $x^-$, $y^+$, and $y^-$ are examples. Fig. 2 compares the orders of the stops arranged using the depot and the reference point $y^-$

$$c_x = \frac{\max_{\forall i}(x_i)+\min_{\forall i}(x_i)}{2}$$
$$c_y = \frac{\max_{\forall i}(y_i)+\min_{\forall i}(y_i)}{2}$$
$$\delta_x = \max_{\forall i}(x_i) - \min_{\forall i}(x_i)$$
$$\delta_y = \max_{\forall i}(y_i) - \min_{\forall i}(y_i)$$
$$reference\ x^+ = (c_x, c_y) + (K^*\delta_x, 0) = (c_x + K^*\delta_x, c_y)$$
$$reference\ x^- = (c_x, c_y) - (K^*\delta_x, 0) = (c_x - K^*\delta_x, c_y)$$
$$reference\ y^+ = (c_x, c_y) + (0, K^*\delta_y) = (c_x, c_y + K^*\delta_y)$$
$$reference\ y^- = (c_x, c_y) - (0, K^*\delta_y) = (c_x, c_y - K^*\delta_y)$$

where $K$ is the distant level parameter.

In our experiment $K$ is set to 5.

In the SWNA, 400 initial solutions are generated using various reference points (the depot, $x^+$, $x^-$, $y^+$, and $y^-$) and the stop ordering methods (clockwise or counter-clockwise direction) and the best solution is selected as the final initial solution. The solution will be improved by the improvement algorithms described below.

### 2.2. Improvement algorithms

After the best initial solution is generated by the sweep-variant method described in Section 2.1 or the current solution is perturbed by the perturbation method described in Section 2.3, our proposed approach applies a series of improvement algorithms until there is no further improvement. This corresponds to Steps 2 and 4 in the proposed overall procedure. The improvement procedure is as follows:

**<Improvement procedure>**

| | |
|---|---|
| Step 1: | Do the following three times<br>Generate random numbers $m$ and $n$ from 0, 1, 2, and 3.<br>Apply inter-route $(m,n)$ exchange improvement to the current solution. |
| Step 2: | Apply inter-route cross exchange to the current solution. |
| Step 3: | Apply intra-route 2-opt to all the routes in the current solution. |
| Step 4: | Apply intra-route Or-opt to all the routes in the current solution. |
| Step 5: | If any of the improvement attempts in Steps 1–4 is succeed, go to Step 1.<br>Otherwise, return the current best solution. |

We use two inter-route algorithms: $(m,n)$ exchange and cross exchange, and two intra-route algorithms: 2-opt and Or-opt. Since $(m,n)$ exchange uses two random numbers $m$ and $n$, one may consider it a type of variable neighborhood method. Fig. 3 shows examples of the improvement algorithms and detailed descriptions are given below.

#### 2.2.1. Inter-route $(m,n)$ exchange

This algorithm first removes $m$ consecutive stops from a route (let's call it $R_a$) and $n$ consecutive stops from another route ($R_b$). The removed $m$ stops are then inserted into $R_b$ one by one and the $n$ stops are inserted into $R_a$. When a stop is inserted into a route, the feasible insertion position that causes the least increment of total distance of the route is selected. When there is no such feasible insertion position, the two routes cannot be improved by the algorithm. The algorithm searches all the possible pairs of the routes and all the possible $m$ and $n$ consecutive stops from the pairs, and accepts the most beneficial exchanges from them. It repeats the procedure until there is no more improvement.

The number of exchange stops $m$ and $n$ are randomly selected from 0, 1, 2, and 3. Depending on the numbers selected, $(m,n)$ exchange can be one of the following: (0,1), (0,2), (0,3), (1,1), (1,2), (1,3), (2,2), (2,3), and (3,3) exchanges.

#### 2.2.2. Inter-route cross exchange

This algorithm exchanges the tails of two routes and checks whether or not the modified routes are feasible and there is

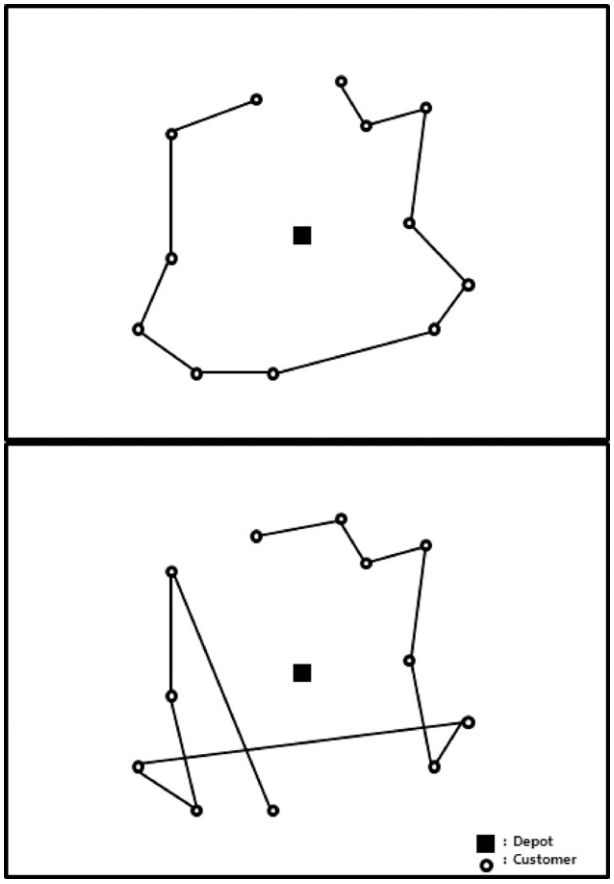


**Fig. 2.** The sequence of stops arranged using the depot (above) and $y^-$ (below) as the reference point.

improvement. For example, suppose a first route has the sequence $(D,1,2,3,4,5,D)$, a second route has the sequence $(D,6,7,8,9,D)$, and $(4,5,D)$ and $(7,8,9,D)$ are current tails of the routes. The routes will be changed to $(D,1,2,3,7,8,9,D)$ and $(D,6,4,5,D)$, respectively. The algorithm searches all the possible pairs of the routes and all the possible tail parts from the pairs, and accepts the most beneficial exchanges from them. It then repeats the procedure until there is no more improvement.

*2.2.3. Intra-route 2-opt*

This is similar to the well-known 2-opt algorithm. This algorithm attempts to reduce the travel distance of a route by exchanging two edges (or links) with two new edges. The algorithm searches all the possible pairs of the edges within a route, and accepts the most beneficial exchanges from them. It then repeats the procedure until there is no more improvement for all the routes.

*2.2.4. Intra-route Or-opt*

This algorithm is due to Or (1976) and it attempts to reduce the travel distance of a route by replacing 1, 2, or 3 consecutive stops between two other stops. The algorithm searches all the possible up-to 3 consecutive stops and all the possible insertion points, and accepts the most beneficial replacement from them. It repeats the procedure until there is no more improvement for all the routes.

*2.3. Perturbation method*

In order to escape from local optima, we use a perturbation mechanism consisting of destroy and repair phases. In the destroy phase, some stops and routes are deleted from the solution. The deleted stops are then reinserted in the repair phase. Through these phases, more diverse solution spaces are searched. This corresponds to Step 3 in the proposed overall procedure. The perturbation procedure is as follows:

**<Perturbation procedure>**

Step 1: Decide the number ($n_d$) of stops that will be removed from the solution.

Step 2: Calculate the number of overlapped routes for each route.
Sort the routes in descending order of the number of overlaps.

Step 3: Select a route from the sorted list of routes in sequence.

Step 4: If the number of stops in the route is less than $n_d$,
  Remove all the stops in the route.
  Remove the route from the solution.
  Set $n_d = n_d$ – the number of stops in the route.
  Go to Step 3.
Otherwise,
  Select $n_d$ stops randomly and remove them from the route.
  Go to Step 5.

Step 5: Set the status of the removed stops to be *unrouted*.

Step 6: Check whether each *unrouted* stop can be inserted into existing routes.
If a stop cannot be inserted into any existing route,
  Initiate a new route with the stop.
  Set the status of the stop to be *routed*.

Step 7: Set an assignment problem (AP) between the *unrouted* stops and the routes.

Step 8: Solve the AP.

Step 9: Select a part of the optimal AP solution.
Add the selected stops to the corresponding routes.
Set the status of the selected stops to be *routed*.

Step 10: If there are remaining *unrouted* stops, go to Step 6.
Otherwise, go to step 11.

Step 11: Return the perturbed solution.

Steps 1–4 are for the destroy phase. In Step 1, the number of stops to be removed from the solution is determined. Two input parameters are used for this: $r_{min}$ (minimum ratio) and $r_{max}$ (maximum ratio). The number of stops is determined randomly between $r_{min} \times N$ and $r_{max} \times N$, where $N$ is the number of stops in the problem. In our experiments, 0.1 and 0.3 are used for $r_{min}$ and $r_{max}$, respectively.

Next, we select some routes in which stops will be removed. When two routes overlap with each other, the routes may be improved. Based on this idea, in Step 2, the number of overlapped routes for each route is calculated and the routes are sorted in descending order of number of overlaps. Thus, the most overlapped route will be the first route in the sorted list. In Step 3, a route from the list is selected. All the stops or a part of the stops from the route are then removed according to the number of stops in the route (Step 4). When there are remaining stops to be deleted, the steps are repeated.

Steps 5–10 are for the repair phase. In Step 6, each unrouted stop is checked whether or not it can be inserted into any existing route. If the stop cannot be inserted into any route, a new route is

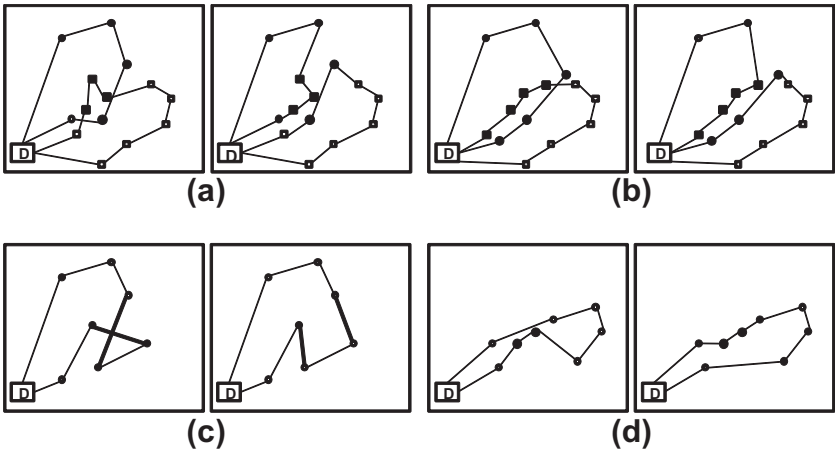


**Fig. 3.** Improvement algorithms: (a) $(m,n)$ exchange with $m = 3$ and $n = 2$, (b) cross exchange over between two routes, (c) 2-opt within a route and (d) Or exchange within a route.

initiated with the stop. Through this procedure, it is guaranteed that all unrouted stops have at least one feasible route for insertion. Then, an assignment problem (AP) is formed between the unrouted stops and the routes (Step 7). The cost between a stop and a route, $c_{stop,route}$, is set to the incremental distance of the route due to the insertion of the stop to the route at the best position if that insertion is feasible. If a stop cannot be inserted into a route, the cost between them is set to a very large number. Note that Steps 6 and 7 can be combined to reduce the computation time.

The AP formed in Step 7 is solved in Step 8. It can be easily solved using the Hungarian method (which is a $O(n^3)$ algorithm), where $n$ is the number of nodes (Papadimitriou & Steiglitz, 1998). The Hungarian algorithm to solve the APs for this research is implemented based on Stachniss' C-language implementation (Stachniss, 2006).

After solving the AP, the algorithm selects part of the optimal solution and adds the selected stops to the selected routes. We select part of the solution rather than all the pairs for this to allow more than one stop to be inserted into a route. From the optimal solution of an AP, a route can take at most one stop. However, the route can take another stop in the next iteration. To select part of the optimal solution, we define the regret measure of a stop as follows:

$$R_{stop} = M - (m_{stop} - c_{stop,route^*})$$

where $M$ = large number, $m_{stop} = \min_{r \in \{allroutes\} - \{route^*\}}(c_{stop,r})$, $c_{stop,route^*}$ = the cost of the optimal pair $(\text{stop},route^*)$.

From the optimal stop-route pairs of AP, a certain number of stop-route pairs that have the largest regret values are selected in Step 9. In our experiments, 3 pairs are used. That is, up to 3 stop-route pairs are selected from the AP optimal solution in each iteration. Steps 6–9 are iterated until all the stops are inserted.

## 3. Computational results

The proposed approach was programmed in Microsoft Visual Studio 2005 C++ and executed on an Intel core i7 CPU920 2.67 GHz computer system, with 6 GB of RAM, under Windows 7. In this section, we describe the benchmark test problems, the effect of SWNA and perturbation, and the computational results obtained by the proposed heuristic approach.

### 3.1. Benchmark data

To evaluate the performance of the proposed approach, we conducted some computational experiments on the well-known and widely used benchmark instances of Nagy and Salhi (2005). The benchmark data set consists of 5 data sets named *T*, *Q*, *H*, *X*, and *Y*, with 14 instances in each set, resulting in 70 problems. For each set, the problems are named 'CMT(number of problem)(name of data set)'. The problems with the same number in different sets have the same number of customers, the same customer locations, and the same vehicle capacity and duration, but have different pickup and delivery demands from each customer. In each instance, the Euclidean distance between two customers can be calculated by their coordinates in the given data, and the objective is to minimize the total distance traveled by the entire vehicle. The problem characteristics are summarized in Table 1.

### 3.2. Effects of SWNA and perturbation

In order to examine the effects of the sweep nearest algorithm (SWNA) and the perturbation method, three combinations of algorithms are tested. In the first two experiments, SWA + Improve and SWNA + Improve, the perturbation method was not applied. While the first experiment uses the original sweep algorithm (SWA) for route construction, the second one uses SWNA. In the third experiment, our final proposed approach, SWNA + Improve + Perturbation (hereinafter called NSP: Nearest Sweep with Perturbation), is applied. Table 2 shows the detailed results for the *T* data set, and Table 3 summarizes the results for other data sets. Compared to SWA, the SWNA provides positive effects on the final solutions. The perturbation method further improves the solution quality

**Table 1**
Statistical summary of the benchmark data.

| Problem | # of customers | Capacity | Duration |
|---|---|---|---|
| CMT1 | 50 | 160 | 999,999 |
| CMT2 | 75 | 140 | 999,999 |
| CMT3 | 100 | 200 | 999,999 |
| CMT4 | 150 | 200 | 999,999 |
| CMT5 | 199 | 200 | 999,999 |
| CMT6 | 50 | 160 | 200 |
| CMT7 | 75 | 140 | 160 |
| CMT8 | 100 | 200 | 230 |
| CMT9 | 150 | 200 | 200 |
| CMT10 | 199 | 200 | 200 |
| CMT11 | 120 | 200 | 999,999 |
| CMT12 | 100 | 200 | 999,999 |
| CMT13 | 120 | 200 | 720 |
| CMT14 | 100 | 200 | 1040 |

**Table 2**
Comparison of various strategies applying the data set *T*.

| Problem | SWA + Improve | | SWNA + Improve | | NSP |
|---|---|---|---|---|---|
| | Total distance | % Gap | Total distance | % Gap | Total distance |
| CMT1T | 530.80 | 2.07 | 524.97 | 0.94 | 520.06 |
| CMT2T | 821.78 | 4.33 | 793.61 | 0.75 | 787.67 |
| CMT3T | 821.30 | 2.56 | 817.36 | 2.06 | 800.83 |
| CMT4T | 1062.56 | 6.64 | 1023.97 | 2.76 | 996.42 |
| CMT5T | 1322.17 | 6.08 | 1270.42 | 1.93 | 1246.34 |
| CMT6T | 570.55 | 2.72 | 565.05 | 1.73 | 555.43 |
| CMT7T | 953.87 | 5.63 | 907.52 | 0.49 | 903.05 |
| CMT8T | 907.09 | 4.80 | 865.54 | 0.00 | 865.54 |
| CMT9T | 1246.37 | 7.00 | 1176.97 | 1.04 | 1164.86 |
| CMT10T | 1484.76 | 5.86 | 1423.30 | 1.48 | 1402.59 |
| CMT11T | 1051.17 | 5.22 | 1026.69 | 2.77 | 999.00 |
| CMT12T | 840.80 | 6.77 | 827.36 | 5.06 | 787.52 |
| CMT13T | 1555.81 | 0.74 | 1550.08 | 0.37 | 1544.37 |
| CMT14T | 857.00 | 3.66 | 847.65 | 2.53 | 826.77 |
| Average | | 4.58 | | 1.71 | |

NSP (our suggested algorithm): SWNA + Improve + perturbation.
% Gap: The % difference of total distance compared to NSP.

**Table 3**
Comparison of % Gap values on data sets *X*, *Y*, *Q*, and *H*.

| Date set | SWA + Improve % Gap | SWNA + Improve % Gap |
|---|---|---|
| *X* | 4.17 | 1.90 |
| *Y* | 5.19 | 2.08 |
| *Q* | 4.44 | 0.93 |
| *H* | 5.58 | 2.92 |
| Average | 4.85 | 1.96 |

% Gap: The % difference of total distance compared to NSP.

**Table 4**
Comparison of best solutions to the NSP solutions to the *X* and *Y* data sets.

| Problem | Best known | | Ref | NSP | |
|---|---|---|---|---|---|
| | # of vehicles | Total distance* | | # of vehicles | Total distance |
| CMT1X | – | **466.77** | S, Z | 3 | **466.77** |
| CMT2X | 7 | **668.77** | W | 6 | 684.75 |
| CMT3X | 5 | 721 | M | 5 | **715.32** |
| CMT4X | 7 | **852.46** | S, Z | 7 | 855.93 |
| CMT5X | 9 | **1029.25** | S, Z | 10 | 1036.36 |
| CMT6X | 6 | 556.06 | W | 6 | **555.43** |
| CMT7X | 11 | **901.22** | R | 11 | **901.22** |
| CMT8X | 9 | 865.51 | C | 9 | **865.50** |
| CMT9X | 14 | 1173.44 | C | 14 | **1161.37** |
| CMT10X | 18 | 1424.06 | C | 18 | **1392.36** |
| CMT11X | – | **833.92** | S, Z | 4 | 866.17 |
| CMT12X | 5 | **644.7** | W | 6 | 662.22 |
| CMT13X | 11 | 1551.25 | C | 11 | **1549.79** |
| CMT14X | 10 | **821.75** | C | 10 | **821.75** |
| CMT1Y | 3 | **458.96** | W | 3 | 467.81 |
| CMT2Y | 6 | **663.25** | W | 6 | 684.75 |
| CMT3Y | 5 | **719** | M | 5 | **719.33** |
| CMT4Y | 7 | 852.35 | CW | 7 | **847.58** |
| CMT5Y | 9 | **1029.25** | S, Z | 10 | 1035.96 |
| CMT6Y | 6 | 556.68 | C | 6 | **555.43** |
| CMT7Y | 11 | 901.22 | C | 11 | **901.10** |
| CMT8Y | 9 | 865.51 | C | 9 | **865.50** |
| CMT9Y | 14 | 1171.95 | C | 14 | **1161.37** |
| CMT10Y | 18 | 1419.79 | W | 18 | **1392.36** |
| CMT11Y | 4 | **830.39** | W | 4 | 839.16 |
| CMT12Y | 6 | **659.52** | W | 5 | 662.99 |
| CMT13Y | 11 | 1547.75 | C | 11 | **1544.37** |
| CMT14Y | 10 | 822.35 | C | 10 | **821.75** |

Ref.: D = Dethloff (2001), R = Ropke (2005), CW = Chen and Wu (2006), M = Montane and Galvao (2006),
W = Wassan et al. (2008), S = Subramanian et al. (2010), Z = Zachariadis et al. (2010), C = Catay (2010).
* The values are reported in the corresponding reference.

from SWNA + Improve. For the remaining experiments, the last combination is used.

### 3.3. Benchmark tests

In our experimental tests, the number of iterations for the initial route construction stage was set to 400 and the number of perturbations was set to 30. NSP results are compared to the current best known solutions. Catay (2010) reports his best solution for the *X* and *Y* data sets compared to the previous best known solutions, which are from Dethloff (2001), Ropke (2005), Chen and Wu (2006), Montane and Galvao (2006) and Wassan, Wassan, and Nagy (2008). We compare the results of NSP to the best known solutions, which are the best in the existing literature, including Catay (2010).

Table 4 shows the comparison of the best known solutions to the NSP solution. Because Ropke (2005) did not report the number of vehicles in his paper, some best known results in Table 4 do not have the number of vehicles. The best solutions are highlighted using bold type. Among 28 instances, the NSP approach could generate new best solutions for 16 instances and similar solutions for 5

**Table 5**
Comparison of best known solutions to the NSP solutions to the *T*, *Q*, and *H* data sets.

| Problem | Best known | | Ref. | NSP | |
|---|---|---|---|---|---|
| | # of vehicles | Total distance* | | # of vehicles | Total distance |
| CMT1T | 5 | **520** | A | 5 | **520.06** |
| CMT2T | 9 | 810 | A | 9 | **787.67** |
| CMT3T | 7 | 827 | A | 7 | **800.83** |
| CMT4T | 11 | 1014 | A | 11 | **996.42** |
| CMT5T | 15 | 1297 | A | 15 | **1246.34** |
| CMT6T | 6 | **555** | A | 6 | **555.43** |
| CMT7T | 12 | 942 | A | 11 | **903.05** |
| CMT8T | 9 | 904 | A | 9 | **865.54** |
| CMT9T | 14 | **1164** | S | 14 | **1164.86** |
| CMT10T | 18 | 1418 | S | 19 | **1402.59** |
| CMT11T | 7 | 1026 | A | 7 | **999.00** |
| CMT12T | 9 | 792 | A | 9 | **787.52** |
| CMT13T | 11 | 1548 | A | 11 | **1544.37** |
| CMT14T | 10 | 846 | A | 10 | **826.77** |
| CMT1Q | 4 | 490 | A | 4 | **489.74** |
| CMT2Q | 8 | 739 | A | 8 | **734.93** |
| CMT3Q | 6 | 768 | A | 6 | **754.57** |
| CMT4Q | 9 | 938 | A | 9 | **915.24** |
| CMT5Q | 13 | 1174 | A | 13 | **1132.33** |
| CMT6Q | 6 | 557 | A | 6 | **555.43** |
| CMT7Q | 12 | 933 | A | 11 | **900.69** |
| CMT8Q | 9 | 890 | A | 9 | **865.50** |
| CMT9Q | 15 | 1178 | S | 14 | **1162.50** |
| CMT10Q | 19 | 1477 | S | 18 | **1406.63** |
| CMT11Q | 6 | 964 | A | 6 | **940.26** |
| CMT12Q | 7 | 733 | A | 7 | **729.25** |
| CMT13Q | 11 | 1570 | A | 11 | **1544.37** |
| CMT14Q | 10 | 825 | A | 10 | **821.75** |
| CMT1H | 3 | 464 | A | 3 | **461.87** |
| CMT2H | 6 | 668 | A | 6 | **660.82** |
| CMT3H | 4 | **701** | A | 4 | 718.81 |
| CMT4H | 6 | 883 | A | 6 | **833.49** |
| CMT5H | 9 | 1044 | A | 9 | **1000.74** |
| CMT6H | 6 | 557 | A | 6 | **555.43** |
| CMT7H | 11 | 943 | A | 11 | **900.12** |
| CMT8H | 9 | 899 | A | 9 | **865.50** |
| CMT9H | 14 | 1164 | S | 14 | **1158.54** |
| CMT10H | 19 | 1499 | A | 18 | **1397.37** |
| CMT11H | 4 | 830 | A | 4 | **819.44** |
| CMT12H | 5 | 635 | A | 5 | **629.98** |
| CMT13H | 11 | 1546 | S | 11 | **1539.79** |
| CMT14H | 10 | 824 | A | 10 | **821.75** |

Ref.: S = Salhi and Nagy (1999) and A = Ai and Kachivichyanukul (2009).
* The values are reported in the corresponding references.

problems. Since some best known solutions were generated by using integer-type distances in previous research, we compare them with approximations. For example, we treat the result of NSP for CMT3Y, 719.33, to be identical to the best known result, 719. The new best solution to CMT4X is provided in the appendix and all the new best solutions are available upon request from the authors.

Results of the application of the proposed algorithm to the problem sets *T*, *Q*, and *H* are compared with the best known results from Salhi and Nagy (1999) and Ai and Kachivichyanukul (2009). Other references do not report the results for the problem sets. Table 5 shows these results. The proposed approach could generate new best solutions for all the instances, except CMT9T and CMT3H. Moreover, in some instances, such as CMT10H, a better solution with huge reduction (more than 5%) from the previous best known solution can be obtained. The new best solution to CMT8Q is provided in the appendix. All the new best solutions are available upon request from the authors.

## 4. Conclusions

This paper presents a heuristic approach consisting of route construction, improvement, and perturbation algorithms to solve VRPSPD. To generate a better initial solution, a new sweep variant, called the nearest sweep algorithm, was proposed. In the route improvement procedure, a series of inter-route $(m,n)$ exchange, cross exchange, intra-route 2-opt, and Or-opt algorithms are applied. The perturbation procedure perturbs a solution by removing some routes and stops from the solution and reinserting them into the solution using the Hungarian method. The last procedure is used to escape from local optima. The computational results on some benchmark problems show that the proposed heuristic outperforms the existing algorithms and many new best solutions have been found. The proposed algorithm could find new best solutions for 53 instances and make tie with the existing best known solutions for 6 instances out of 70 benchmark problem instances. The proposed approach can be further applied to many other variants of the VRP.

## Acknowledgement

This work was supported by a Korea Science and Engineering Foundation (KOSEF) grant funded by the Korean government (MEST) (No. 2010-0015611).

## Appendix A

| Route # | Route sequence | Distance |
|---|---|---|
| *(a) New best solution to CMT4X problem* | | |
| | Number of customers: 199, vehicle capacity: 200, vehicle duration: 200, service time: 10 | |
| 1 | 0–124–168–47–36–143–49–64–11–0 | 112.63 |
| 2 | 0–70–131–32–181–63–126–90–108–10–189–0 | 90.07 |
| 3 | 0–105–40–73–171–74–75–133–22–41–145–115–178–2–0 | 69.98 |
| 4 | 0–146–52–153–106–194–7–182–148–88–31–190–127–167–27–0 | 57.05 |
| 5 | 0–110–4–155–139–187–170–25–55–165–130–195–26–0 | 79.70 |
| 6 | 0–149–179–54–134–24–163–80–150–177–109–12–138–154–0 | 68.67 |
| 7 | 0–18–82–48–123–19–107–175–62–159–162–69–0 | 83.53 |
| 8 | 0–6–96–104–99–93–85–91–193–98–92–151–95–94–183–0 | 53.83 |
| 9 | 0–132–101–30–160–128–66–188–20–122–1–176–0 | 85.33 |
| 10 | 0–152–58–137–144–57–15–43–142–42–172–87–13–0 | 77.13 |
| 11 | 0–28–76–196–116–77–3–158–129–79–185–33–157–102–50–0 | 59.96 |
| 12 | 0–59–37–100–192–119–44–191–141–16–61–173–5–0 | 76.07 |
| 13 | 0–21–72–197–56–186–23–67–39–198–180–0 | 93.25 |
| 14 | 0–118–84–17–113–86–140–38–14–97–117–0 | 98.93 |
| 15 | 0–9–135–35–136–65–71–161–103–51–0 | 104.63 |
| 16 | 0–111–81–120–164–34–78–169–29–121–68–184–0 | 86.83 |
| 17 | 0–147–60–83–199–125–45–46–174–8–114–166–89–0 | 79.16 |
| 18 | 0–156–112–53–0 | 15.62 |
| Total | | 1392.36 |
| *(b) New best solution to CMT8Q problem* | | |
| | Number of customers: 100, vehicle capacity: 200, vehicle duration: 230, service time: 10 | |
| 1 | 0–13–87–42–43–14–44–38–86–16–61–99–0 | 111.40 |
| 2 | 0–94–95–97–92–37–98–100–91–85–93–59–96–6–0 | 59.51 |
| 3 | 0–26–4–56–23–67–39–25–55–54–0 | 107.08 |
| 4 | 0–58–2–57–15–41–22–75–74––72–73–21–40–53–0 | 83.10 |
| 5 | 0–28–76–77–3–79–78–34–29–24–68–80–12–0 | 90.26 |
| 6 | 0–52–7–19–11–64–49–36–47–48–82–18–0 | 117.55 |
| 7 | 0–89–60–83–8–46–45–17–84–5–0 | 88.06 |
| 8 | 0–1–51–20–66–65–71–35–9–81–33–50–0 | 117.93 |
| 9 | 0–31–88–62–10–63–90–32–30–70–69–27–0 | 90.12 |
| Total | | 865.53 |

## References

Ai, T. J., & Kachivichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers and Operations Research, 36*, 1693–1702.

Catay, B. (2010). A new saving-based ant algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications, 37*, 6809–6817.

Chen, J. F., & Wu, T. H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society, 57*, 579–587.

Dethloff, J. (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum, 23*, 79–96.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research, 22*, 340–349.

Golden, B., Ball, M., & Bodin, L. (1981). Current and future research directions in network optimization. *Computers and Operations Research, 8*, 71–81.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A, 23*, 377–386.

Montane, F. A. T., & Galvao, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research, 33*, 595–619.

Nagy, G., & Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research, 162*, 126–141.

Or, I. (1976). Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. New York: Dover.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research, 34*, 2403–2435.
Ropke, S. (2005). Heuristic and exact algorithms for vehicle routing problems. Ph.D. thesis, Computer Science Department, University of Copenhagen.
Salhi, S., & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society, 50*, 1034–1042.
Stachniss, C., 2006. C-implementation of the Hungarian method. <www.informatik.uni-freiburg.de/~stachnis/misc.html> Accessed 01.03.06.
Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S., & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers and Operations Research, 37*, 1899–1911.
Wassan, N. A., Wassan, A. H., & Nagy, G. (2008). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization, 15*, 368–386.
Wren, A. (1971). *Computers in transport planning and operation*. London: Ian Allan.
Wren, A., & Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operations Research Quarterly, 23*, 333–344.
Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications, 36*, 1070–1081.
Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and delivery. *European Journal of Operational Research, 202*, 401–411.