ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics
Bachelor Thesis
Econometrics and Operations Research

# A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery

Assem Mussagulova, 453776
Supervisor: Y.N. Hoogendoorn
Second assessor: K.S. Postek

7 July 2019

**Abstract**

The vehicle routing problem with simultaneous pickup and delivery is the variation of the vehicle routing problem that arises when a vehicle starts and ends its route at the depot and performs both the collection and delivery of goods when visits a customer. In this paper, we solve the problem using the Particle Swarm Optimization (PSO) and reproduce the results obtained by Ai and Kachitvichyanukul (2009). Additionally, we propose incorporating the single move (SM) and customer exchange (CE) heuristics in the algorithm to test if it can improve the solution quality. We compare the performance of the PSO, PSO with SM, PSO with CE and PSO with SM and CE. We find that the PSO with SM and CE gives the best and more robust solution at the cost of increased computational time.

# Contents

**1 Introduction** **3**

**2 Methodology** **4**
2.1 VRPSPD description . . . . . 4
2.2 Particle Swarm Optimization for VRPSPD . . . . . 4
2.3 Decoding algorithm . . . . . 7
2.4 Search for appropriate number of vehicles . . . . . 8
2.5 Single move heuristic . . . . . 9
2.6 Customer exchange heuristic . . . . . 10

**3 Computational experiments** **11**
3.1 Data . . . . . 11
3.2 PSO for VRPSPD . . . . . 12
3.3 PSO with the single move heuristic . . . . . 16
3.4 PSO with the customer exchange heuristic . . . . . 16
3.5 PSO with the single move and customer exchange heuristics . . . . . 17
3.6 Comparison of convergence . . . . . 18

**4 Conclusion** **19**

**References** **21**

# 1 Introduction

The growing awareness of the importance of re-use and recycling gave a rise to reverse logistics (Bianchessi & Righini, 2007). It implies pickup of end-of-life or excess products from customers to manufacturers. The process has found many applications since it helps to reduce the waste and effectively use the materials, diminishing the environmental impact. There are some areas in which the delivery of goods and pick-up of products can be integrated. Such reverse logistics networks can be modelled by a VRPSPD. The practical example of its implementation is the re-use of packages in soft drinks industry when new bottles are delivered and empty bottles are picked up from the same customer.

The VRPSPD is the extension of the vehicle routing problem (VRP). The VRP was first introduced by Dantzig and Ramser (1959) to find optimal routes for delivering gasoline from a bulk terminal to service stations. These days, it is widely applied by logistics and express delivery companies (Laporte et al., 2013) which are essential in businesses' operations. The VRP is a combinatorial optimization problem of constructing a set of cost-minimizing vehicle routes which start from the depot, visit a number of customers, and return to the same depot.

In the VRPSPD, vehicles are required to satisfy clients' pickup and delivery demand simultaneously. Goods go strictly from the depot to the customers and picked up products go from the customers back to the depot. The problem was first introduced by Min (1989) and applied in transporting books between libraries. To solve the VRPSPD, Min (1989) proposed a cluster-first route-second approach. He used this method to find routes for a problem with two vehicles and 22 customers. Later, the first exact method was introduced by Dell'Amico et al. (2006). They applied a branch-and-price technique to solve the problem. However, even for small instances consisting of 40 customers, the computational time was about an hour. Thus, heuristic algorithms are commonly used for solving the VRPSPD as they are more computationally efficient and the literature is focused on them more than on exact methods.

Salhi and Nagy (1999) presented four insertion-based heuristics to solve VRPSPD. They differ by the number of customers that can be inserted simultaneously and by the criteria for insertion. Bianchessi and Righini (2007) developed constructive, local search and tabu search algorithms. Another technique for solving VRPSPD is a parallel heuristic which involves iterated local search combined with multi-start heuristic as proposed by Subramanian et al. (2010).

In this thesis, we focus on a Particle Swarm Optimization (PSO) algorithm for solving the VRPSPD. The PSO was developed by Kennedy and Eberhart (1995) and simulated the social behaviour of organisms. The idea is that particles move in the search space trying to improve a solution with each iteration. Ai and Kachitvichyanukul (2009) extended this technique and introduced a PSO algorithm for solving VRPSPD. They conducted computational experiments on the benchmark instances and showed that the results often outperformed the best-known solutions. Furthermore, a 50-customer problem was solved, on average, within 30 seconds. This proves the efficiency of the proposed method. We discuss it in more detail in Section 2.2.

The aim of our research is to implement a PSO for solving VRPSPD and reproduce the results obtained by Ai and Kachitvichyanukul (2009). Then, we examine if the performance of the method can be enhanced. Our ultimate goal is to attain lower transportation costs. Since it can be done by further optimizing routes, we focus on this step. First, we propose a single move heuristic and examine if using it instead of 2-opt leads to a better solution. Second, we extend the route construction algorithm with a customer exchange heuristic. We test if incorporating one or two of these techniques can improve the solution quality and if the algorithm stays efficient.

# 2 Methodology

In this section, we present a PSO algorithm for solving the VRPSPD based on Ai and Kachitvichyanukul (2009) and the methods of how it can be improved. We first describe the VRPSPD problem in Section 2.1. Then, we explain the PSO for solving the VRPSPD in Section 2.2. In the two subsequent sections, we provide the decoding algorithm for the solution of the PSO and the routine which determines an appropriate number of vehicles. In Sections 2.5 and 2.6, we present a single move and customer exchange heuristics which are used to improve the solution quality of the algorithm.

## 2.1 VRPSPD description

The VRPSPD is defined on a graph $G = (V, A)$, where $V = \{v_0, v_1, ..., v_n\}$ is a vertex set consisting of the depot $v_0$ and customers to be served $v_1, ..., v_n$, and $A = \{(v_i, v_j)|v_i, v_j \in V, i \neq j\}$ is an arc set. Each arc is associated with a distance $d_{ij} > 0$ and travel time $t_{ij} > 0$. In total, there are $m$ homogeneous vehicles and $n$ customers. The ultimate goal is to construct a set of vehicle routes which minimizes transportation costs. The costs are composed of a fixed cost $f$ for using each vehicle, and a variable cost $g$ for each distance unit travelled. Every customer has a delivery demand $q_i$ and pickup demand $p_i$ as well as a service time $s_i$. The total route duration comprises service and transportation times. The load of the vehicle is equal to the total delivery quantities of the corresponding vehicle when it departs from the depot. After serving a customer $i$, its load reduces by the quantity delivered $q_i$ and increases by the picked up quantity $p_i$. The following constraints must be taken into account in solving the problem:

- Each route must originate from the depot and end there.
- Goods are delivered from the depot to the customers and picked up goods end up at the depot.
- Each customer $i = 1, ..., n$ is served exactly once.
- The total vehicle load does not exceed its capacity $Q$ on any arc.
- The route duration is not longer than a preset limit $D$.

## 2.2 Particle Swarm Optimization for VRPSPD

PSO is a population-based search algorithm that simulates physical movements of a swarm of organisms. A swarm consists of $L$ particles and each particle $l \in \{1, ..., L\}$ has its own position, $\Theta_l = [\theta_{l1}, ..., \theta_{lH}]$, decoded into a set of vehicle routes using Algorithm 2. The position vector $\Theta_l$ is composed of $H = n + 2m$ elements presented as real numbers in the range $[\theta^{min}, \theta^{max}]$ (Ai & Kachitvichyanukul, 2009).

The movements of a particle are driven by its velocity $\Omega_l = [\omega_{l1}, ..., \omega_{lH}]$. The velocity vector also consists of $H$ dimensions and is updated at each iteration $\tau$. In the standard PSO algorithm, velocity depends on inertia, cognitive learning, and social learning terms. A particle moves in the same direction as in the previous iteration due to the inertia term, $w(\tau)$. At the beginning, the inertia term is high meaning that the particles are likely to follow their current direction. It decreases with each iteration making the particles follow their cognitive and social terms. The cognitive learning term forces a particle to walk to its personal best position (pbest), while the social term pushes a particle to make a movement toward the global best position (gbest). The pbest of a particle $l$, $\Psi_l = [\psi_{l1}, ..., \psi_{lH}]$, is always kept and updated when a particle reaches a better objective function than the previous pbest. We also keep track of the global best position (gbest), $\Psi_g = [\psi_{g1}, ..., \psi_{gh}]$, which gives the best objective function among all positions visited by $L$ particles of a swarm. It is updated when a particle attains a better objective value than the previous gbest (Kennedy & Eberhart, 1995).

The PSO algorithm for solving the VRPSPD is based on the GLNPSO and uses two additional social learning terms. Apart from pbest and gbest, it keeps and updates the local best position (lbest) and the near neighbor best position (nbest). Lbest, $\Psi_l^L = [\psi_{l1}^L, ..., \psi_{lH}^L]$, is the best position found by $K$ adjacent neighbors of the $l$th particle based on the indices of the particles in the swarm. As an illustration, in Figure 1, we show five adjacent neighbors of particle 1 which are particles $L-1$, $L$, 1, 2 and 3. Nbest, $\Psi_l^N = [\psi_{l1}^N, ..., \psi_{lH}^N]$, is the position chosen to maximize the fitness-distance-ratio ($FDR$) (Veeramachaneni et al., 2003). Its dimensions are found independently and the formula for $FDR$ is shown on line 27 of Algorithm 1. The cognitive and social learning terms are computed by taking a product of a uniform random number ($u_p, u_g, u_l$ or $u_n$) defined in the interval [0,1], an acceleration constant ($c_p$, $c_g$, $c_l$ or $c_n$) and the difference between cognitive or social component ($\Psi_l$, $\Psi_g$, $\Psi_l^L$ or $\Psi_l^N$) and current position $\Theta_l$. Overall, the velocity of a particle is updated based on the five terms.

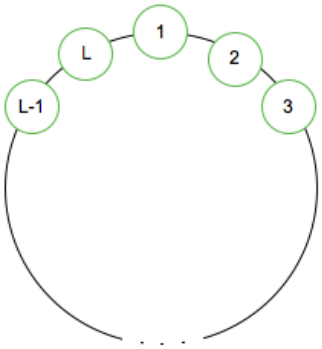

Figure 1: Adjacent neighbors of particle 1

The PSO for solving the VRPSPD is described in Algorithm 1. On lines 1 – 4, we initialize a number of customers to be served, and determine an appropriate number of vehicles using Algorithm 3 which is explained in detail in Section 2.4. Then, on lines 5 – 13, we initialize $L$ particles and their velocity, their personal best position as well as the global best position and its fitness value. In total, the algorithm performs $T$ iterations and at each iteration $\tau \in \{1, ..., T\}$, we update the inertia term with the formula presented on line 15. The inertia terms for the first and last iteration are the parameters.

On lines 16 – 27, we evaluate the position of a particle $l$ at iteration $\tau$, $\Theta_l(\tau)$. First, we decode it into a set of vehicle routes $R_l$ by applying Algorithm 2. After that, we compute its fitness value $Z(\Theta_l)$ representing the objective function of the corresponding solution which is equal to the transportation costs. Additionally, we determine the number of customers that are not served by the set of routes $R_l$ and add a penalty $p$ to the fitness value $Z(\Theta_l)$ for each such client. Due to the fact that a particle $l$ searches a position with a lower fitness value, this helps to avoid further movements toward an infeasible solution. We proceed by checking if the current position of particle $l$ is better than its pbest or gbest, and update them if a better solution was attained. Next, we update the particle's lbest and each dimension of nbest.

On lines 28 – 35, we update every dimension of particle's velocity $\Omega(\tau + 1)$ and determine the position for the next iteration. If any dimension $h$ of the particle's position moves beyond the bounds, we set it to the minimum or maximum value and the corresponding dimension of its velocity to 0 (Ai & Kachitvichyanukul, 2009).

**Algorithm 1** PSO algorithm for VRPSPD

$v \leftarrow$ available number of vehicles
$n \leftarrow$ number of customers
generate particle $a$ with random position $\Theta_a$ consisting of $H = 2v + n$ dimensions bounded by $[\theta^{min}, \theta^{max}]$
$m \leftarrow$ appropriate number of vehicles found by implementing Algorithm 3 using $a$
$S \leftarrow$ swarm of $L$ particles
**for** $l = 1 : L$ **do**
    initialize particle $l$ with $H = 2m + n$ dimensions
    add $l$ to swarm $S$
    $\Theta_l \leftarrow$ random position of particle $l$ in the range $[\theta^{min}, \theta^{max}]$
    velocity $\Omega_l \leftarrow 0$
    pbest $\Psi_l \leftarrow \Theta_l$
initialize gbest $\Psi_g$
$Z(\Psi_g) \leftarrow 0$
**for** $\tau = 1 : T$ **do**
    $w(\tau) \leftarrow w(T) + \frac{\tau - T}{1 - T}[w(1) - w(T)]$
    **for** $l = 1 : L$ **do**
        $R_l \leftarrow$ set of vehicle routes obtained from decoding $\Theta_l(\tau)$ using Algorithm 2
        $Z(\Theta_l) \leftarrow$ transportation costs of $R_l$
        $b \leftarrow$ number of customers not served in $R_l$
        $Z(\Theta_l) \leftarrow Z(\Theta_l) + b * p$
        **if** $Z(\Theta_l) < Z(\Psi_l)$ **then**
            pbest $\Psi_l \leftarrow \Theta_l$
        **if** $Z(\Psi_l) < Z(\Psi_g)$ **then**
            gbest $\Psi_g \leftarrow \Psi_l$
        lbest $\Psi_l^L \leftarrow$ pbest with the lowest $Z(\Psi_l)$ among $K$ neighbors of $l$
        **for** $h = 1 : H$ **do**
            $h$th dimension of nbest $\psi_{lh}^N \leftarrow \psi_{oh}$ that maximizes $FDR = \frac{Z(\Theta_l) - Z(\Psi_o)}{|\theta_{lh} - \psi_{oh}|}$, for $o = 1, ..., L$ where $o \neq l$
            $h$th dimension of velocity $\omega_{lh}(\tau + 1) \leftarrow w(\tau)\omega_{lh}(\tau) + c_p u_1(\psi_{lh} - \theta_{lh}(\tau)) + c_g u_2(\psi_{gh} - \theta_{lh}(\tau)) + c_l u_3(\psi_{lh}^L - \theta_{lh}(\tau)) + c_n u_4(\psi_{lh}^N - \theta_{lh}(\tau))$
            $h$th dimension of $l$'s position $\theta_{lh}(\tau + 1) \leftarrow \theta_{lh}(\tau) + \omega_{lh}(\tau + 1)$
            **if** $\theta_{lh}(\tau + 1) > \theta^{max}$ **then**
                $\theta_{lh}(\tau + 1) \leftarrow \theta^{max}$
                $\omega_{lh}(\tau + 1) \leftarrow 0$
            **if** $\theta_{lh}(\tau + 1) < \theta^{min}$ **then**
                $\theta_{lh}(\tau + 1) \leftarrow \theta^{min}$
                $\omega_{lh}(\tau + 1) \leftarrow 0$

## 2.3 Decoding algorithm

As mentioned before, at each iteration of the PSO algorithm (Ai & Kachitvichyanukul, 2009), we update the position of each particle, and decode it into a set of vehicle routes. The position vector is composed of $H = n + 2m$ elements presented as real numbers which can be split into two parts. Figure 2 shows the components of the position vector. The first part consists of $n$ real numbers corresponding to $n$ customers. The numbers indicate their priority to enter a route, where a small value means a high priority of a client.

The second part of the vector is composed of $2m$ real numbers related to the route orientation of $m$ available vehicles. Each vehicle corresponds to two dimensions, one for the x-coordinate and the other for the y-coordinate value. Customers to be served are located in two-dimensional Euclidean space and their coordinates are known. The route orientation indicates a point in the service map around which a vehicle is likely to move and perform a search. It ensures that a vehicle serves clients located close to each other and, consequently, helps to minimize the route distance.

To limit the particles' search space, we determine the bounds of the area on the service map covered by customers. The minimum value is found by taking the minimum of the x-coordinates and y-coordinates of all customers' locations. The maximum value of x-axis and y-axis gives the maximum boundary to the search space. We use these values as the range for the vehicles' route orientation points, $[\theta^{min}, \theta^{max}]$. Since the bounds do not affect the first $n$ dimensions related to customers, we use them to limit each dimension $h \in H$ of a particle's position vector $\Theta_l$ (Ai & Kachitvichyanukul, 2009).

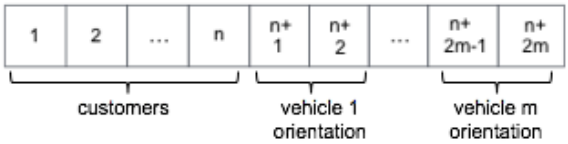


Figure 2: Particle position representation

The decoding procedure is presented in Algorithm 2. The idea is to construct a priority list of customers $U$ based on the values of the first $n$ dimensions where a smaller value represents a higher priority. This is done on lines 1 – 6 of Algorithm 2. Next, for every customer $i = 1, ..., n$, we determine the priority list of vehicles $W_i$ on lines 7 – 18 of Algorithm 2. We compute the Euclidean distance between the location of customer $i$ and the route orientation point of each vehicle $j = 1, ..., m$. Then, we add vehicles one by one to $W_i$ where the vehicles with the minimum distance to the customer $i$ are added first. Therefore, priority to serve a customer is given to the vehicles closest to the customer.

Finally, when the customer priority list $U$ and the vehicle priority lists $W_1, ..., W_n$ are constructed, we create vehicle routes on lines 19 – 29 of Algorithm 2. Customers are assigned one by one to the routes starting from the top of the customer priority list $U$. When a customer $c$ is extracted from $U$, we try to insert it to the route of the vehicle with the highest priority for this customer , $W_{c,1}$, using the cheapest insertion heuristic and check if the candidate route is feasible. If it is, we update the route with a candidate route re-optimized with the 2-opt method and proceed with the next client. Otherwise, we repeat this step until the customer is not assigned to a route (Ai & Kachitvichyanukul, 2009).

**Algorithm 2** Decoding algorithm

initialize priority list of customers $U \leftarrow \emptyset$
initialize set of customers $S \leftarrow \{1, 2, ..., n\}$
**while** $S \neq \emptyset$ **do**
    select customer $c$ from $S$ with $\theta_{lc} = \min_{h \in S} \theta_{lh}$
    add $c$ to the last position in $U$
    remove $c$ from $S$
initialize vehicle priority matrix $W$
**for** $i = 1 : n$ **do**
    **for** $j = 1 : m$ **do**
        vehicle orientation coordinate $x_j \leftarrow \theta_{l,n+2j-1}$
        vehicle orientation coordinate $y_j \leftarrow \theta_{l,n+2j}$
        $d_j \leftarrow \sqrt{(xpos_i - x_j)^2 + (ypos_i - y_j)^2}$
    $W_i \leftarrow \emptyset$
    $S \leftarrow \{1, 2, ..., m\}$
    **while** $S \neq \emptyset$ **do**
        select customer $c$ from $S$ with $d_c = \min_{j \in S} d_j$
        add $c$ to the last position in $W_i$
        remove $c$ from $S$
**for** $k = 1 : n$ **do**
    $c \leftarrow U_k$
    **for** $b = 1 : m$ **do**
        **if** feasible insertion is not found **then**
            $j \leftarrow W_{c,b}$
            $r \leftarrow R_j$, route of vehicle $j$
            insert customer $c$ to the best position in route $r$ using cheapest insertion heuristic
            **if** $r$ is feasible **then**
                re-optimize $r$ using 2-opt method
                $R_j \leftarrow r$

## 2.4 Search for appropriate number of vehicles

In the PSO algorithm for VRPSPD (Ai & Kachitvichyanukul, 2009), we use a fixed number of vehicles $m$. As there is a fixed cost $f$ for using each vehicle, the objective is to find the number of vehicles that minimizes the fixed costs and keeps the variable costs low. This number is determined by implementing Algorithm 3. It is performed as the first step of Algorithm 1.

We start the search for an appropriate number of vehicles $m$ with the number of available vehicles $v$. First, on lines 1 – 3, we generate a random particle $l$ and decode it into a set of vehicle routes $R_l$ by applying Algorithm 2. Then, we determine the fitness value of the particle, $Z$. We proceed by removing the dimensions corresponding to the vehicle which serves the lowest number of customers and reduce the size of the particle by two dimensions on lines 8 – 12. Next, we decode an updated particle into a new set of vehicle routes and compute its fitness value, $Z'$. If $Z'$ is lower than the fitness value of the original particle, then we set $Z$ to $Z'$, decrease the number of vehicles by one and try to remove another vehicle in the same way. Otherwise, we stop the search. The appropriate number of vehicles is $m$ (Ai & Kachitvichyanukul, 2009).

**Algorithm 3** Search for appropriate number of vehicles

**Input:** initial number of vehicles $v$
**Output:** appropriate number of vehicles $m$

1: generate random particle $l$ consisting of $H = 2v + n$ dimensions
2: $m \leftarrow v$
3: $R_l \leftarrow$ set of vehicles routes obtained from decoding $l$ using Algorithm 2
4: $Z \leftarrow$ fitness value of $l$
5: $repeat \leftarrow$ true
6: **while** $repeat =$ true **do**
7: $repeat =$ false
8: **for** $i = 1 : m$ **do**
9: $s_i \leftarrow$ number of customers served by vehicle $i$
10: $k \leftarrow \text{argmin}_i\, s_i$
11: remove the dimensions of particle $l$ corresponding to the vehicle $k$
12: reduce size of $l$ by two dimensions
13: $R_l \leftarrow$ set of vehicles routes obtained from decoding $l$ using Algorithm 2
14: $Z' \leftarrow$ fitness value of $l$
15: **if** $Z' < Z$ **then**
16: $Z \leftarrow Z'$
17: $m \leftarrow m - 1$
18: $repeat \leftarrow$ true

## 2.5 Single move heuristic

To improve the solution quality of the PSO algorithm for solving the VRPSPD proposed by Ai and Kachitvichyanukul (2009), we try to replace the 2-opt method on line 27 of Algorithm 2 with a method which we call the single move heuristic. The idea of the 2-opt heuristic is to change the order in which the customers are visited in the route. If the route crosses over itself, the algorithm removes two edges and reconnects the two paths (Nilsson, 2003). In the single move heuristic given in Algorithm 4, we also try to re-optimize a route by changing the order in which the customers are visited. The difference is that in this heuristic, we consider each client one by one, try to move it to every possible position in the route and keep it at the position which results in the lowest transportation cost.

On lines 1 – 9 of Algorithm 4, we consider every possible move of customer at position $i$ to position $j$ in the route $r$. Then, on lines 10 – 13, we check if the move results in a shorter distance and if it does, we update the route $r$. An example of a successful movement is illustrated in Figure 3. It shows that by moving a customer at the third position to the second position, we obtain a shorter route.

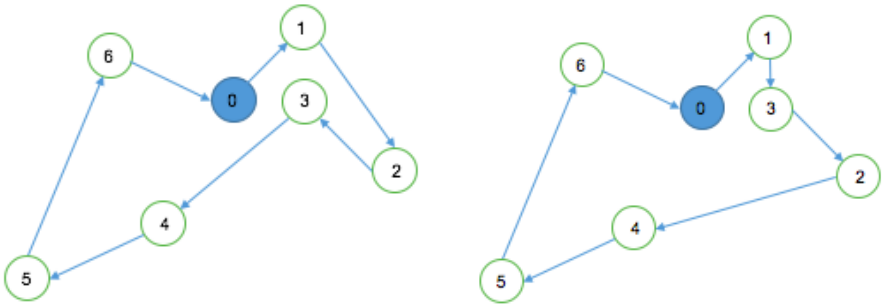


Figure 3: A single movement of a customer at position 3 to position 2

**Algorithm 4** Single move heuristic

**Input:** route $r$
**Output:** optimized route $r$

1: $n \leftarrow$ number of customers in route $r$
2: $f \leftarrow$ fitness value of route $r$
3: **for** $i = 1 : n$ **do**
4:  **for** $j = 1 : n$ **do**
5:  **if** $i \neq j$ **then**
6:  $c \leftarrow$ customer at position $i$ in $r$
7:  $r_{new} \leftarrow r$
8:  remove customer at position $i$ from $r_{new}$
9:  insert customer $c$ to position $j$ in $r_{new}$
10:  $f_{new} \leftarrow$ fitness value of $r_{new}$
11:  **if** $f_{new} < f$ **then**
12:  $r \leftarrow r_{new}$
13:  $f \leftarrow f_{new}$

## 2.6 Customer exchange heuristic

In Algorithm 2, we re-optimize each route separately. However, there might be a possibility that the routes can be further improved by interchanging customers between them. Therefore, as an additional step of the decoding algorithm, when all routes are constructed, we try to perform a so-called customer exchange heuristic given in Algorithm 5. In this procedure, we consider every customer of every route $i$ and try to move client $j$ to a different route $l$ using the cheapest insertion heuristic. At the same time, we try to remove each customer one by one from route $l$ and insert them to the route $i$ at the position which gives the lowest additional cost. These steps are shown on lines 1 – 19 of Algorithm 5. After the customers are exchanged, we check the feasibility of new routes and their transportation costs on lines 20 – 23. If the new routes are feasible and cheaper than the original routes, we update the route set with the obtained routes, go back to line 3 and proceed with the next customer of route $i$. Otherwise, we return the customers back to the initial routes and continue the search for the current customer $j$ from line 12. An example of a customer exchange resulting in a decreased distance is represented in Figure 4.

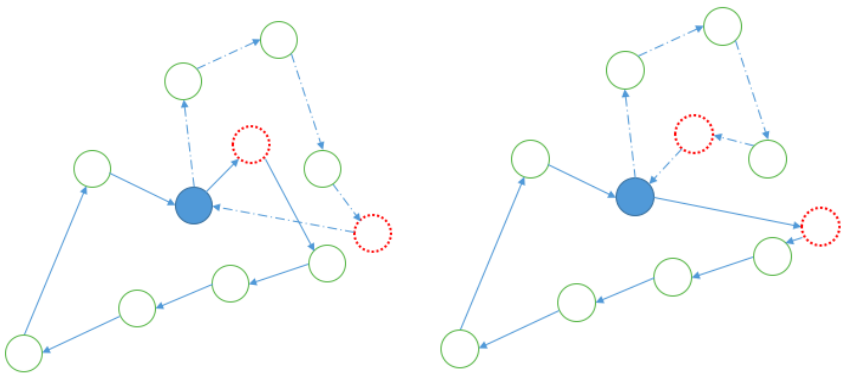

Figure 4: Customer exchange between routes

**Algorithm 5** Customer exchange heuristic

**Input:** set of routes $R = \{1, ..., r\}$
**Output:** optimized set of routes $R$

**for** $i = 1 : r$ **do**
  $n \leftarrow$ number of customers visited in route $i$
  **for** $j = 1 : n$ **do**
    $c \leftarrow$ customer at position $j$ of route $i$
    $route_1 \leftarrow$ route $i$ of set $R$
    $cost_1 \leftarrow$ cost of $route_1$
    remove $c$ from $route_1$
    $exchanged \leftarrow$ false
    **for** $l = 1 : r$ **do**
      **if** $i \neq l$ **then**
        $m \leftarrow$ number of customer visited in route $l$
        **for** $p = 1 : m$ **do**
          **if** $exchanged =$ false **then**
            $k \leftarrow$ customer at position $p$ of route $l$
            $route_2 \leftarrow$ route $l$ of set $R$
            $cost_2 \leftarrow$ cost of $route_2$
            remove $k$ from route $route_2$
            add $k$ using the cheapest insertion heuristic to $route_1$
            add $c$ using the cheapest insertion heuristic to $route_2$
            $cost_1^{new} \leftarrow$ cost of $route_1$
            $cost_2^{new} \leftarrow$ cost of $route_2$
            **if** $(cost_1 + cost_2) > (cost_1^{new} + cost_2^{new})$ **then**
              **if** $route_1$ and $route_2$ are feasible **then**
                route $i$ of set $R \leftarrow route_1$
                route $l$ of set $R \leftarrow route_2$
                $exchanged \leftarrow$true

# 3 Computational experiments

## 3.1 Data

To test the performance of the PSO algorithm for solving the VRPSPD, we solve benchmark instances obtained from Salhi and Nagy (1999). They include five data sets named H, Q, T, X and Y, each consisting of 14 instances. In our computational experiments, we test H, Q, T and X. The results for these problems are known from Ai and Kachitvichyanukul (2009), Salhi and Nagy (1999), Dethloff (2001), Montané and Galvao (2006), and are used for the comparison with the results obtained in this paper.

As the initial number of available vehicles, we use the number of vehicles found in the best-known solution of Salhi and Nagy (1999), Dethloff (2001), Montané and Galvao (2006) as it was done in Ai and Kachitvichyanukul (2009). The number of customers in the instances varies between 50 and 199. The variable cost per distance unit is $g = 1$, the fixed cost for using a vehicle is $f = 0$. The X problems are also tested for $f = 100$. Distance and traveling time are equal. The parameters used for the PSO algorithm are given in Table 1.

Table 1: PSO parameters

| Parameter | Notation | Value |
|---|---|---|
| Number of particles | $L$ | 50 |
| Number of neighbors | $K$ | 5 |
| Number of iterations | $T$ | 1000 |
| First inertia weight | $w(1)$ | 0.9 |
| Last inertia weight | $w(T)$ | 0.4 |
| Penalty for not serving a customer | $p$ | 1000 |
| Pbest acceleration constant | $c_p$ | 1 |
| Gbest acceleration constant | $c_g$ | 0 |
| Lbest acceleration constant | $c_l$ | 1 |
| Nbest acceleration constant | $c_n$ | 2 |

### 3.2 PSO for VRPSPD

We perform the computational experiments using the data sets discussed in Section 3.1 and compare the results to the best-known solutions. The PSO algorithm is programmed in Java using Mac OS. We run the program 10 times for each instance and pick the best solution with the lowest fitness value. First, we replicate the results attained by Ai and Kachitvichyanukul (2009) and run the PSO algorithm for H, Q, T and X instances. We compare the results with the best solutions achieved by Ai and Kachitvichyanukul (2009), Salhi and Nagy (1999), Dethloff (2001), and Montané and Galvao (2006). The obtained number of vehicles and total costs together with their percentage deviation $\%dev$ from the best-known solutions for the case with the fixed cost per vehicle $f = 0$ are shown in Table 2 for instances H, Q and T. The results for X instances are given in Table 3. The percentage deviation $\%dev$ of the obtained result from the best-known cost is computed as follows:

$$\%dev = \frac{Z - Z^*}{Z^*} 100\% \tag{1}$$

where $Z$ is the value obtained from our computational experiment, and $Z^*$ is the reference value.

The replicated results are close to the best-known solutions with the median of $\%dev$ equal to 1.1%. Only for two instances the percentage deviation $\%dev$ exceeds 70%. This might be because the experiments are implemented on different operating systems and these instances include the highest number of customers (199 and 150). Thus, the program may need more iterations for problems with bigger size to find a better solution.

Table 2: Comparison of best-known solution and best PSO solution of H, Q and T instances with fixed cost $f = 0$ and variable cost $g = 1$

| Instance | Best-known solution | | Best PSO solution | | $\%dev$ |
|---|---|---|---|---|---|
| | Number of vehicles | Total cost (distance) | Number of vehicles | Total cost (distance) | |
| CMT1H | 3 | 464 | 4 | 468 | 0.9 |
| CMT2H | 6 | 668 | 6 | 680 | 1.8 |
| CMT3H | 4 | 701 | 5 | 731 | 4.3 |
| CMT4H | 6 | 883 | 9 | 923 | 4.5 |
| CMT5H | 9 | 1044 | 12 | 1139 | 9.1 |
| CMT6H | 6 | 557 | 6 | 559 | 0.4 |
| CMT7H | 11 | 943 | 11 | **931** | -1.3 |
| CMT8H | 9 | 899 | 9 | **892** | -0.8 |
| CMT9H | 14 | 1164 | 14 | 2212 | 90 |
| CMT10H | 19 | 1499 | 18 | 1509 | 0.7 |
| CMT11H | 4 | 830 | 4 | 854 | 2.9 |
| CMT12H | 5 | 635 | 5 | 641 | 0.9 |
| CMT13H | 11 | 1546 | 11 | 1573 | 1.7 |
| CMT14H | 10 | 824 | 10 | 841 | 2.1 |
| CMT1Q | 4 | 490 | 4 | 494 | 0.8 |
| CMT2Q | 8 | 739 | 8 | 754 | 2.0 |
| CMT3Q | 6 | 768 | 6 | 776 | 1.0 |
| CMT4Q | 9 | 938 | 9 | 956 | 1.9 |
| CMT5Q | 13 | 1174 | 13 | 1204 | 2.6 |
| CMT6Q | 6 | 557 | 6 | 559 | 0.4 |
| CMT7Q | 12 | 933 | 12 | 945 | 1.3 |
| CMT8Q | 9 | 890 | 9 | **888** | -0.2 |
| CMT9Q | 15 | 1178 | 15 | 1268 | 7.6 |
| CMT10Q | 19 | 1477 | 19 | 1546 | 4.7 |
| CMT11Q | 6 | 964 | 6 | 991 | 2.8 |
| CMT12Q | 7 | 733 | 7 | 751 | 2.5 |
| CMT13Q | 11 | 1570 | 11 | 1578 | 0.5 |
| CMT14Q | 10 | 825 | 10 | 827 | 0.2 |
| CMT1T | 5 | 520 | 5 | 520 | 0.0 |
| CMT2T | 9 | 810 | 9 | 811 | 0.1 |
| CMT3T | 7 | 827 | 7 | 828 | 0.1 |
| CMT4T | 11 | 1014 | 12 | 1047 | 3.3 |
| CMT5T | 15 | 1297 | 16 | 1333 | 2.8 |
| CMT6T | 6 | 555 | 6 | 559 | 0.7 |
| CMT7T | 12 | 942 | 11 | **930** | -1.3 |
| CMT8T | 9 | 904 | 9 | **881** | -2.5 |
| CMT9T | 14 | 1164 | 14 | 1199 | 3.0 |
| CMT10T | 18 | 1418 | 18 | 2480 | 74.9 |
| CMT11T | 7 | 1026 | 7 | 1035 | 0.9 |
| CMT12T | 9 | 792 | 9 | 810 | 2.3 |
| CMT13T | 11 | 1548 | 11 | 1565 | 1.1 |
| CMT14T | 10 | 846 | 10 | **841** | -0.6 |

Note: in bold are the results that outperformed the best-known solutions.

Table 3: Comparison of best-known solution and best PSO solution of X instances with fixed cost $f = 0$ and variable cost $g = 1$

| Instance | Best-known solution | | Best PSO solution | | $\%dev$ |
|---|---|---|---|---|---|
| | Number of vehicles | Total cost (distance) | Number of vehicles | Total cost (distance) | |
| CMT1X | 3 | 467 | 3 | 472 | 1.1 |
| CMT2X | 7 | 695 | 7 | 713 | 2.6 |
| CMT3X | 5 | 721 | 5 | 740 | 2.6 |
| CMT4X | 7 | 880 | 7 | 938 | 6.6 |
| CMT5X | 11 | 1098 | 11 | 1172 | 6.7 |
| CMT6X | 6 | 557 | 6 | 559 | 0.4 |
| CMT7X | 11 | 919 | 11 | **916** | -0.3 |
| CMT8X | 9 | 896 | 9 | **879** | -1.9 |
| CMT9X | 15 | 1215 | 14 | 1227 | 1.0 |
| CMT10X | 19 | 1520 | 19 | **1503** | -1.1 |
| CMT11X | 4 | 895 | 4 | 913 | 2.0 |
| CMT12X | 6 | 675 | 5 | **674** | -0.1 |
| CMT13X | 11 | 1560 | 11 | 1578 | 1.2 |
| CMT14X | 10 | 826 | 10 | **823** | -0.4 |

Note: in bold are the results that outperformed the best-known solutions.

In Table 4, we present the statistical results obtained in 10 iterations for the data set X, including the average total cost, the standard deviation of the total cost, percentage of standard deviation over the average, and the average computational time in seconds. It can be observed that the bigger the size of the instance, the less robust the results are. For example, for the problem CMT1X with 50 customers standard deviation in percentage terms is only 0.9%, while for the instance CMT5X with 199 clients, it is equal to 28.0%. Furthermore, there is a positive correlation between computational time and the size of the problem. This relationship is depicted in Figure 5.

Table 4: Statistical summary of PSO results for X instances

| Instance | Number of customers | Average total cost | St. dev. | % St. dev. | Average comp. time (sec) |
|---|---|---|---|---|---|
| CMT1X | 50 | 473 | 0.98 | 0.9 | 44 |
| CMT2X | 75 | 718 | 3.13 | 0.9 | 40 |
| CMT3X | 100 | 742 | 2.47 | 2.3 | 231 |
| CMT4X | 150 | 950 | 11.68 | 11.1 | 406 |
| CMT5X | 199 | 1191 | 29.51 | 28.0 | 484 |
| CMT6X | 50 | 560 | 1.64 | 1.6 | 38 |
| CMT7X | 75 | 923 | 5.68 | 5.4 | 68 |
| CMT8X | 100 | 897 | 11.08 | 10.5 | 106 |
| CMT9X | 150 | 1245 | 10.33 | 9.8 | 196 |
| CMT10X | 199 | 1521 | 11.93 | 11.3 | 326 |
| CMT11X | 120 | 926 | 8.82 | 8.4 | 489 |
| CMT12X | 100 | 680 | 5.61 | 5.3 | 185 |
| CMT13X | 120 | 1587 | 6.36 | 6.0 | 134 |
| CMT14X | 100 | 832 | 9.61 | 9.1 | 99 |

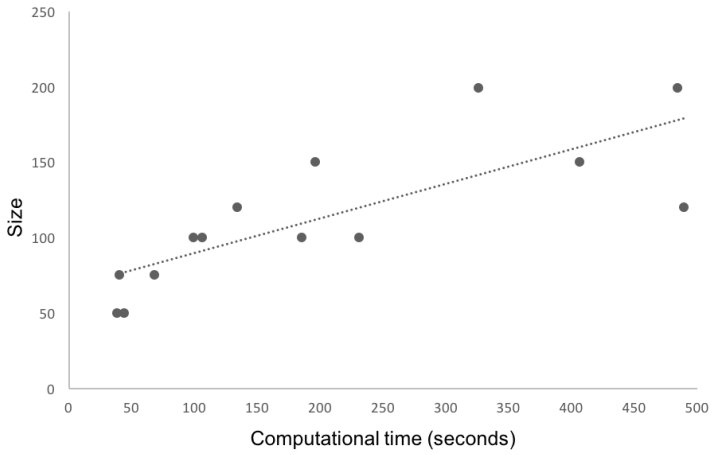


Figure 5: Correlation of the problem size and computational time of X instances

In the previous computational experiments, we used fixed cost per vehicle $f = 0$, so the transportation costs were equal to the variable costs and the number of vehicles was chosen by the PSO in a way that minimizes the distance traveled. In reality, we often need to pay for each vehicle involved in transporting products. Therefore, to check the effect of including the fixed cost to the objective function, we solve the problem for the X data set with $f = 100$ and present the results in Table 5. We can notice that the number of vehicles decreased by one for CMT8X and stayed the same for other instances comparing to the solutions obtained with $f = 0$ from Table 3. Moreover, for 8 out of 14 instances, the total distance of the solution with $f = 100$ is slightly lower than the total distance of the solution with $f = 0$. The percentage deviation of distance varies between -3.0% and 2.2% and its median is equal to -0.2%.

Table 5: Comparison of best-known solution and best PSO solution of X instances with fixed cost $f = 100$ and variable cost $g = 1$, and the percentage deviation of best PSO distance with $f = 100$ from best PSO distance with $f = 0$

| Instance | Best-known solution | | | Best PSO solution | | | $\%dev$ of total cost from best-known | $\%dev$ of total distance from best PSO with $f = 0$ |
|---|---|---|---|---|---|---|---|---|
| | Number of vehicles | Total distance | Total cost | Number of vehicles | Total distance | Total cost | | |
| CMT1X | 3 | 467 | 767 | 3 | **471** | 771 | 0.5 | -0.2 |
| CMT2X | 6 | 707 | 1307 | 7 | 717 | 1417 | 8.4 | 0.6 |
| CMT3X | 5 | 721 | 1221 | 5 | 745 | 1245 | 2 | 0.7 |
| CMT4X | 7 | 880 | 1580 | 7 | 939 | 1639 | 3.7 | 0.1 |
| CMT5X | 10 | 1150 | 2150 | 11 | **1158** | 2258 | 5 | -1.2 |
| CMT6X | 6 | 557 | 1157 | 6 | 559 | 1159 | 0.2 | 0.0 |
| CMT7X | 11 | 931 | 2031 | 11 | **913** | **2013** | -0.9 | -0.3 |
| CMT8X | 9 | 902 | 1802 | 9 | 891 | **1791** | -0.6 | 1.4 |
| CMT9X | 15 | 1215 | 2715 | 14 | **1190** | **2590** | -4.6 | -3.0 |
| CMT10X | 19 | 1499 | 3399 | 19 | **1489** | **3389** | -0.3 | -0.9 |
| CMT11X | 4 | 898 | 1298 | 4 | **911** | 1311 | 1 | -0.2 |
| CMT12X | 5 | 682 | 1182 | 5 | 689 | 1189 | 0.6 | 2.2 |
| CMT13X | 11 | 1570 | 2670 | 11 | **1574** | 2674 | 0.1 | -0.3 |
| CMT14X | 10 | 824 | 1824 | 10 | **822** | **1822** | -0.1 | -0.1 |

Note: in bold are the costs that are better the best-known solutions and the distances that are better than best PSO solutions with $f = 0$.

In Table 5, we also compare the PSO results of X instances with $f = 100$ with the best-known solutions of Ai and Kachitvichyanukul (2009), Salhi and Nagy (1999), Dethloff (2001), and Montané and Galvao (2006). The replicated results do not deviate significantly from the best-known solutions. The median of the percentage deviation is 0.4%. The PSO gave lower costs than in the best-known solution in 5 out of 14 cases.

### 3.3 PSO with the single move heuristic

To test if the PSO algorithm for solving the VRPSPD can be refined, we try to replace the 2-opt method with the single move heuristic discussed in Section 2.5 and perform computational experiments on instances X. For each instance, we run the program 10 times and compare the result of the best replication with the best PSO solution from Section 4.1 and the best-known solution of Ai and Kachitvichyanukul (2009), Salhi and Nagy (1999), Dethloff (2001), and Montané and Galvao (2006). In Table 6, we present the obtained results, statistical summary and the percentage deviation of total cost from the best PSO and best-known solution and the percentage deviation of the computational time in seconds.

The PSO with the SM heuristic attained almost the same results as the initial PSO with $\%dev$ varying between -1.2% and 2.6%. However, it is worth drawing the attention that the PSO with the SM is more robust compared to the general PSO for solving the VRPSPD with the 2-opt method. The median of standard deviation in percentage terms is equal to 2.55% for the PSO with SM and 7.2% for the general PSO. $\%dev$ of computational time fluctuates and there are both positive and negative deviations from the average computational time of the general PSO.

Table 6: Best obtained solution of PSO with single move heuristic, its statistical summary and comparison with best PSO solution for X instances with fixed cost $f = 0$ and variable cost $g = 1$

| Instance | Best obtained solution | | | | Statistical summary | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Number of vehicles | Total cost | $\%dev$ from best PSO | $\%dev$ from best-known | Aver. comp. time (sec) | $\%dev$ of comp. time | Average cost | St. dev. | % St. dev. |
| CMT1X | 3 | 472 | 0 | 1.1 | 102 | 133.6 | 472 | 0 | 0 |
| CMT2X | 7 | **710** | -0.5 | 2.2 | 113 | 183.7 | 713 | 2.2 | 2.1 |
| CMT3X | 5 | **735** | -0.6 | 1.9 | 334 | 44.3 | 740 | 2.7 | 2.6 |
| CMT4X | 7 | 946 | 0.9 | 7.5 | 546 | 34.3 | 949 | 2.6 | 2.5 |
| CMT5X | 11 | **1165** | -0.6 | 6.1 | 579 | 19.5 | 1169 | 3.9 | 3.7 |
| CMT6X | 6 | 559 | 0 | 0.4 | 26 | -32.2 | 560 | 0.6 | 0.4 |
| CMT7X | 11 | 917 | 0.1 | 0.1 | 42 | -38.2 | 922 | 2.1 | 2 |
| CMT8X | 9 | 882 | 0.4 | 0.3 | 98 | -7.7 | 886 | 2.9 | 2.7 |
| CMT9X | 15 | 1236 | 0.8 | 1.7 | 145 | -31.6 | 1246 | 7.7 | 7.3 |
| CMT10X | 19 | 1543 | 2.6 | 2.7 | 221 | -32.3 | 1550 | 9.9 | 9.4 |
| CMT11X | 4 | **902** | -1.2 | 0.8 | 362 | -26 | 905 | 2.5 | 2.3 |
| CMT12X | 6 | 688 | 2 | 2.1 | 219 | 18.5 | 690 | 1.6 | 1.6 |
| CMT13X | 11 | **1558** | -1.3 | -0.1 | 118 | -11.6 | 1562 | 7.8 | 8 |
| CMT14X | 10 | **823** | -0.1 | 0.0 | 71 | -27.6 | 826 | 3.3 | 3.1 |

Note: in bold are the results that outperformed the best-known or the best PSO solutions.

### 3.4 PSO with the customer exchange heuristic

To improve the solution of the PSO for solving the VRPSPD, as the last step of the decoding algorithm we introduced the customer exchange heuristic in Section 2.6. The computational results of the PSO incorporated with the CE algorithm for all X instances are demonstrated in Table 7. We compare the best solution obtained in 10 iterations with the best PSO solution and the best-known solution of Ai and Ka-

chitvichyanukul (2009), Salhi and Nagy (1999), Dethloff (2001), and Montané and Galvao (2006). The PSO with the CE outperformed the general PSO in 10 out of 14 cases and improved the best-known solution of 4 instances. The median percentage deviation of the PSO with CE from the general PSO is -0.8% what indicates that the method tends to improve the solution of the VRPSPD.

Furthermore, the CE makes the PSO more robust decreasing the median of standard deviation in percentage terms from 7.2% to 2%. The method also has a major drawback. Its average computational time is up to 17 times higher comparing to the general PSO with the average increase of about 10 times. This is caused by the fact that the algorithm needs to check many variations of customer exchanges every time when the algorithm constructs a set of routes.

Table 7: Best obtained solution of PSO with customer exchange heuristic, its statistical summary and comparison with best PSO solution for X instances with fixed cost $f = 0$ and variable cost $g = 1$

| Instance | Best obtained solution | | | | Statistical summary | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Number of vehicles | Total cost | $\%dev$ from best PSO | $\%dev$ from best-known | Aver. comp. time (sec) | $\%dev$ of comp. time | Average cost | St. dev. | % St. dev. |
| CMT1X | 3 | **471** | -0.2 | 0.9 | 492 | 1018.6 | 471 | 0.2 | 0.2 |
| CMT2X | 7 | 712 | 0.3 | 2.5 | 729 | 1723.4 | 714 | 2.2 | 2.0 |
| CMT3X | 5 | 737 | 0.3 | 2.2 | 2743 | 1087.4 | 739 | 1.3 | 1.3 |
| CMT4X | 7 | **906** | -4.2 | 3.0 | 7366 | 1714.4 | 908 | 1.4 | 1.3 |
| CMT5X | 10 | **1133** | -2.7 | 3.2 | 10974 | 2167.3 | 1135 | 2.1 | 2.0 |
| CMT6X | 6 | 562 | 0.5 | 0.9 | 192 | 405.3 | 565 | 2.0 | 1.9 |
| CMT7X | 11 | **906** | -1.2 | -1.1 | 316 | 364.7 | 908 | 3.2 | 3.0 |
| CMT8X | 9 | **876** | -0.7 | -0.3 | 1104 | 941.2 | 879 | 2.1 | 2.0 |
| CMT9X | 15 | **1220** | -1.3 | 0.4 | 1856 | 846.9 | 1230 | 7.8 | 7.4 |
| CMT10X | 19 | **1503** | -2.6 | 0.0 | 3683 | 1029.8 | 1515 | 8.5 | 8.0 |
| CMT11X | 4 | **894** | -0.9 | -0.2 | 6778 | 1286.1 | 897 | 3.4 | 3.2 |
| CMT12X | 6 | **679** | -1.3 | 0.7 | 2172 | 1074.1 | 680 | 1.3 | 1.3 |
| CMT13X | 11 | **1555** | -0.2 | -0.3 | 1495 | 1015.7 | 1561 | 6.2 | 5.9 |
| CMT14X | 10 | 824 | 0.1 | 0.1 | 784 | 691.9 | 825 | 1.4 | 1.4 |

Note: in bold are the results that outperformed the best-known or the best PSO solutions.

### 3.5 PSO with the single move and customer exchange heuristics

The last computational experiments are conducted on data set X using the PSO algorithm incorporated with both SM and CE heuristics. Table 8 gives the comparison of the best solution attained in 10 iterations with the best PSO and best-known solution of Ai and Kachitvichyanukul (2009), Salhi and Nagy (1999), Dethloff (2001), and Montané and Galvao (2006). The PSO with SM and CE reduced the fitness value of the general PSO with the 2-opt heuristic for 13 out of 14 instances. The median of the percentage deviation of the total cost from the best PSO is -0.8% with the maximum decrease of 5.1% and the only increase in the objective function of 0.1%.

Table 8 also provides the statistical summary and the percentage deviation of the computational time in seconds. Its median of standard deviation in percentage terms is the lowest among all considered techniques and is equal to 1.85%. The good indicators are reached at the cost of high computational times which are considerably higher comparing to the general PSO. The average increase in computational time of the PSO with CE rose from 10 to 12 times after changing the 2-opt method to the SM heuristic.

Table 8: Best obtained solution of PSO with single move and customer exchange heuristics, its statistical summary and comparison with best PSO solution for X instances with fixed cost $f = 0$ and variable cost $g = 1$

| Instance | Best obtained solution | | | | Statistical summary | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Number of vehicles | Total cost | $\%dev$ from best PSO | $\%dev$ from best-known | Aver. comp. time (sec) | $\%dev$ of comp. time | Average cost | St. dev. | % St. dev. |
| CMT1X | 3 | **470** | -0.4 | 0.6 | 602 | 1279.2 | 471 | 0.1 | 0.1 |
| CMT2X | 6 | **707** | -0.8 | 1.7 | 860 | 2059.9 | 711 | 1.6 | 1.5 |
| CMT3X | 5 | **734** | -0.7 | 1.8 | 3106 | 1242.2 | 735 | 0.8 | 0.8 |
| CMT4X | 7 | **889** | -5.1 | 1.0 | 7814 | 1823.4 | 895 | 1.5 | 1.4 |
| CMT5X | 11 | **1122** | -4.2 | 2.2 | 11823 | 2340.8 | 1124 | 2.7 | 2.7 |
| CMT6X | 6 | **558** | -0.1 | 0.2 | 234 | 511.4 | 559 | 0.8 | 0.8 |
| CMT7X | 11 | **906** | -1.1 | -1.1 | 331 | 387.2 | 914 | 2.9 | 2.7 |
| CMT8X | 9 | **870** | -1 | -1.0 | 1200 | 1029.6 | 874 | 2.1 | 1.9 |
| CMT9X | 15 | **1217** | -0.8 | 0.2 | 2104 | 973.4 | 785 | 5.9 | 5.5 |
| CMT10X | 19 | **1500** | -0.2 | -0.2 | 3638 | 1014.2 | 1511 | 12.5 | 1.8 |
| CMT11X | 4 | **891** | -2.4 | -0.4 | 6848 | 1300.3 | 892 | 2.99 | 2.82 |
| CMT12X | 6 | 675 | 0.1 | 0.1 | 2453 | 1227.3 | 677 | 1.9 | 1.8 |
| CMT13X | 11 | **1554** | -1.5 | -0.4 | 1651 | 1136.5 | 1557 | 4.4 | 4.1 |
| CMT14X | 10 | **822** | -0.2 | -0.1 | 849 | 761.2 | 822 | 1 | 0.9 |

Note: in bold are the results that outperformed the best-known or the best PSO solutions.

### 3.6 Comparison of convergence

In Figures 6 and 7, we present the convergence of the fitness value of the best-found solution obtained with four methods: PSO, PSO with SM heuristic instead of 2-opt, PSO with CE heuristic and PSO with SM and CE. The results are shown for the problems CMT2X with 75 customers and CMT8X including 100 clients. It can be seen that the PSO incorporated with SM and CE and PSO with CE give the fastest convergence with respect to the number of iterations. The other two methods have a similar to each other pattern and their fitness value declines more linearly. The figures show that the PSO with SM and CE and the PSO with CE give the lowest fitness value at the first iteration and have an inverse relationship with the number of iterations.

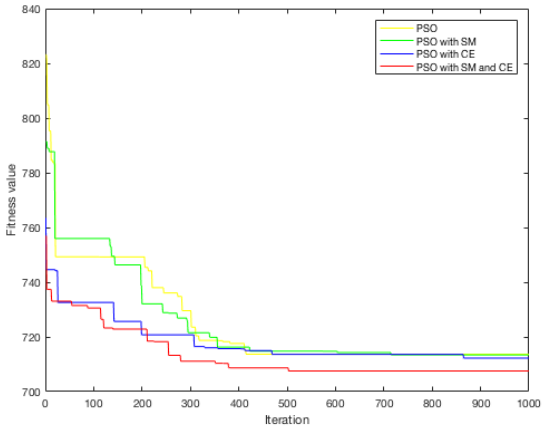


Figure 6: Convergence of the fitness value of CMT2X

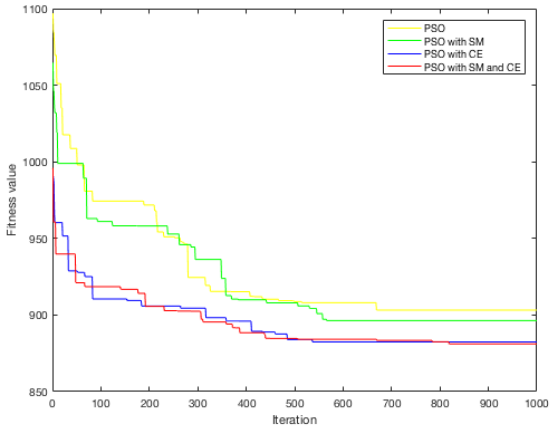


Figure 7: Convergence of the fitness value of CMT8X

As we found, the PSO with SM and CE has the highest computational time and its inverse trend with respect to the number of iterations can be explained by the fact that each iteration of this method requires more time. We check the convergence of the fitness value of CMT2X and CMT8X with respect to time when solving with the PSO with SM and CE and compare it to the general PSO in Figures 8 and 9. It is demonstrated that when we consider two methods with respect to the computational time, the general PSO converges faster. However, the PSO with SM and CE converges to its optimal solution in the middle of the running process. Thus, the actual $\%dev$ of computational time of the PSO with SM and CE is lower because it requires around twice less iterations to reach its optimal point. From Figures 8 and 9, we can see that it takes around four times longer to reach the lowest fitness value attained by the PSO for the PSO with SM and CE. The deviation is still high, but decreasing the number of redundant iterations when running the PSO with SM and CE can save a considerable amount of time.

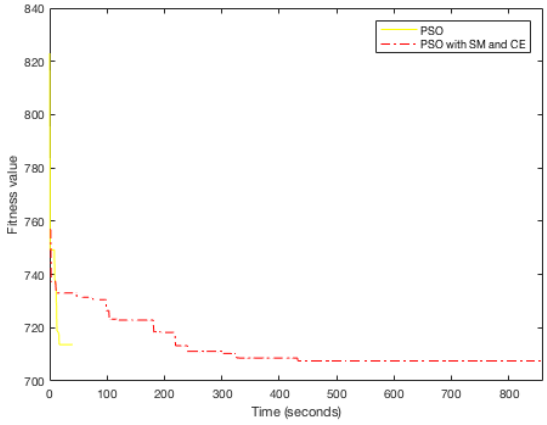


Figure 8: Convergence of the fitness value of CMT2X with respect to time

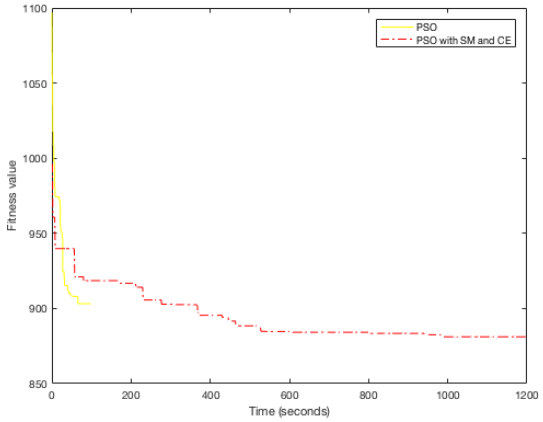


Figure 9: Convergence of the fitness value of CMT8X with respect to time

# 4 Conclusion

In this thesis, we considered four methods for solving the VRPSPD. The main algorithm we analyzed is the Particle Swarm Optimization by Ai and Kachitvichyanukul (2009). We implemented the algorithm and found the solutions for four data sets. The results have shown that the PSO is an efficient technique that often outperforms the best-known solutions obtained with other methods.

To improve the performance of the PSO for solving the VRPSPD further, we replaced the 2-opt method with the single move heuristic and incorporated the customer exchange heuristic as the last step of the decoding algorithm. The heuristics were tested separately and then simultaneously. In total, four variations of the PSO were compared which are the general PSO, the PSO with SM, the PSO with CE, and the PSO with SM and CE. We have determined that the PSO with SM and CE is, on average, the most robust among the four methods and attains the lowest fitness value. However, these come with a significantly higher computational time comparing to the general PSO. It was represented on few instances that the computational time can be lowered by reducing the number of iterations since the PSO with SM and CE converges faster to its optimal solution with respect to the number of iterations. This could be investigated more on other instances and then, the optimal number of iterations that saves time and does not lead to the deterioration in the quality of the solution, could be found.

More research could be done on the customer exchange heuristic as the impact of incorporating it into the PSO algorithm is considerable, especially together with the single move algorithm. The algorithm could be more efficient if it tried to exchange the customers corresponding to the longest arcs first as it could

help to lower the number of exchanges that the algorithm tests. Furthermore, He et al. (2016) conducted a research on the dependence of the performance and efficiency of the PSO on the parameter selection and found that the parameters have a key influence on the performance of the algorithm. Hence, optimizing the parameters like acceleration constants and the first and last inertia weight may be also beneficial for the PSO algorithm.

## References


Ai, T. J., & Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Computers  Operations Research, 36(5), 1693-1702.

Bianchessi, N., & Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. Computers  Operations Research, 34(2), 578-594.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management science, 6(1), 80-91.

Dell'Amico, M., Righini, G., & Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. Transportation science, 40(2), 235-247.

Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. OR-Spektrum, 23(1), 79-96.

He, Y., Ma, W. J., & Zhang, J. P. (2016). The parameters selection of PSO algorithm influencing on performance of fault diagnosis. In MATEC Web of conferences (Vol. 63, p. 02019). EDP Sciences.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization (PSO). In Proc. IEEE International Conference on Neural Networks, Perth, Australia (pp. 1942-1948).

Laporte, G., Toth, P., & Vigo, D. (2013). Vehicle routing: historical perspective and recent contributions.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. Transportation Research Part A: General, 23(5), 377-386.

Montané, F. A. T., & Galvao, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Computers  Operations Research, 33(3), 595-619.

Nilsson, C. (2003). Heuristics for the traveling salesman problem. Linkoping University, 38, 00085-9.

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(10), 1034-1042.

Subramanian, A., Drummond, L. M. D. A., Bentes, C., Ochi, L. S., & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Computers  Operations Research, 37(11), 1899-1911.

Veeramachaneni, K., Peram, T., Mohan, C., & Osadciw, L. A. (2003, July). Optimization using particle swarms with near neighbor interactions. In Genetic and evolutionary computation conference (pp. 110-121). Springer, Berlin, Heidelberg.

# Appendix

A Zip file with the following programs is enclosed:

- *Particle.java*, a class that models a particle and stores its position
- *PSO.java*, a class that performs the Particle Swarm Optimization algorithm for solving the VRPSPD
- *PSO_SM.java*, a class that performs the Particle Swarm Optimization algorithm with the single move heuristic for solving the VRPSPD
- *PSO_CE.java*, a class that performs the Particle Swarm Optimization algorithm with the customer exchange heuristic for solving the VRPSPD
- *PSO_SM_CE.java*, a class that performs the Particle Swarm Optimization algorithm with the single move and customer exchange heuristics for solving the VRPSPD.