Computers & Industrial Engineering 65 (2013) 39–53

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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

A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery

Fatma Pinar Goksal[a], Ismail Karaoglan[b], Fulya Altiparmak[c,*]

[a] Department of Industrial Engineering, Baskent University, Ankara, Turkey
[b] Department of Industrial Engineering, Selcuk University, Konya, Turkey
[c] Department of Industrial Engineering, Gazi University, Ankara, Turkey

ARTICLE INFO

Article history:
Available online 20 January 2012

Keywords:
Vehicle routing problem
Simultaneous pickup and delivery
Particle swarm optimization
Variable neighborhood descent algorithm

ABSTRACT

Vehicle routing problem (VRP) is an important and well-known combinatorial optimization problem encountered in many transport logistics and distribution systems. The VRP has several variants depending on tasks performed and on some restrictions, such as time windows, multiple vehicles, backhauls, simultaneous delivery and pick-up, etc. In this paper, we consider vehicle routing problem with simultaneous pickup and delivery (VRPSPD). The VRPSPD deals with optimally integrating goods distribution and collection when there are no precedence restrictions on the order in which the operations must be performed. Since the VRPSPD is an NP-hard problem, we present a heuristic solution approach based on particle swarm optimization (PSO) in which a local search is performed by variable neighborhood descent algorithm (VND). Moreover, it implements an annealing-like strategy to preserve the swarm diversity. The effectiveness of the proposed PSO is investigated by an experiment conducted on benchmark problem instances available in the literature. The computational results indicate that the proposed algorithm competes with the heuristic approaches in the literature and improves several best known solutions.

1. Introduction

In today's competitive environment, it is obvious that companies should make strategic and operational decisions in order to optimize and manage the processes in their logistics system more efficiently. One of the most important operational decisions concerns to finding optimal vehicle routes since it offers great potential to reduce costs and to improve service quality. The classical vehicle routing problem (VRP) can be defined as the determination of an optimal set of routes for a fleet of vehicles which needs to serve a set of customers. Since it was introduced by Dantzig and Ramser (1959), attention has been devoted to more complex variants of the VRP appearing in real life such as time windows, multiple vehicles and backhauls. The comprehensive review on classical VRP, its variants, formulations, and solution methods can be found in Toth and Vigo (2002).

One of the variants of the VRP is the vehicle routing problem with simultaneous pickup and delivery (VRPSPD). In the VRPSPD, the vehicles are not only required to deliver goods to customer, but also pickup goods at customer locations. A general assumption in the VRPSPD is that all delivered goods must be originated from the depot and all pickup goods must be transported back to the depot. Delivery and pickup goods must be met simultaneously when each customer is visited only once by a vehicle and unloading is carried out before loading at the customers (Chen & Wu, 2006).

The VRPSPD can be defined formally as follows: Let $G = (N,A)$ be a complete directed network, where $N = N_0 \cup \{0\}$ is the set of vertices in which N_0 represents the customers and "0" represents the depot, respectively. $A = \{(i, j): i, j \in N\}$ is the set of arcs, and to each arc (i,j) is associated a nonnegative cost (distance) c_{ij}. In the depot there are identical vehicles with a capacity Q and there is no restriction on the number of vehicles. In the VRPSPD, each customer $i \in N_0$ requires a given quantity to be delivered (d_i) and picked-up (pc_i). The VRPSPD consists of finding a set of routes such that each route starts and ends at the depot, each customer is visited exactly once by one vehicle, the total vehicle load in any arc does not exceed the capacity of the vehicle and the total routing cost is minimized (Montane & Galvão, 2006). The critical feature of the problem is that both pickup and delivery activities should be carried out simultaneously by the same vehicle. Therefore, when a solution approach is developed for the problem, a mechanism that check the fluctuating load on the vehicle at each customer should be imposed into it to prevent the vehicle is overloaded. For mathematical formulations of the VRPSPD, the interested readers are referred the studies of Ai and Kachitvichyanukul (2009a), Dethloff (2001), Nagy and Salhi (2005), and Tang and Galvao (2006). The VRPSDP is related to the vehicle routing

* Corresponding author.
E-mail addresses: pgoksal@baskent.edu.tr (F.P. Goksal), ikaraoglan@selcuk.edu.tr (I. Karaoglan), fulyaal@gazi.edu.tr (F. Altiparmak).

0360-8352/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2012.01.005

problem with backhauls (VRPBs) in which all deliveries to the linehaul customers must be made before any pickup from a backhaul customer. When the delivery or pickup is allowed in any order of sequence, the problem is called the vehicle routing problem with mixed pickup and delivery (VRPMPD). The VRPMPD can be considered as a special case of the VRPSPD because when the delivery or pickup demand of each customer is set to zero, the problem reduces to VRPMPD. Hence, a solution approach developed to solve the VRPSPD can be directly used to solve the VRPMPD (Wassan, Wassan, & Nagy 2007; Wassan, Nagy, & Ahmadi 2008).

The application of the VRPSPD is frequently encountered in the distribution system of grocery store chains. Each grocery store may have both a delivery (e.g., fresh food or soft drink) and pickup (e.g., outdated items or empty bottles) demands and is serviced with a single stop (Chen & Wu, 2006). Reverse logistics is also another application area for the VRPSPD as companies become interested in gaining control over the whole lifecycle of their products. For example, in some countries legislation forces companies to take responsibility for their products during lifetime, especially when environmental issues are involved (Dethloff, 2001; Montane & Galvão, 2006).

The VRPSPD is an NP-hard problem (Salhi & Nagy, 1999), because it can be considered as the VRP, which is well-known NP-hard problem, when only delivery goods or pickup goods are considered. Thus, after Min (1989) introduced the VRPSPD, the research on this problem has been mainly focused on heuristic and meta-heuristic approaches which can produce high-quality solutions within limited computational times (Berbeglia, Cordeau, Gribkovskaia, & Laporte 2007; Parragh, Doerner, & Hartl 2008).

Recently, research attention for combinatorial optimization has turned to hybridization of meta-heuristics. It is assumed that combining features of different heuristics in complementary fashion can result in more robust and effective optimization tools (Talbi, 2002). Thus, previous experiments have showed that the effectiveness and efficiency of hybrid algorithms are often better than those of more simplistic ones. Some examples for hybrid algorithms applied to various VRPs can be given as follows: Tan, Lee, and Ou (2001) apply different hybridizations of TS, simulated annealing (SA) and genetic algorithms (GAs) to solve the VRP with time windows. Lee, Lee, Lin, and Ying (2010) hybridize ant colony optimization (ACO) and SA for the capacitated VRP. Another hybrid algorithm based on evolutionary strategies and memory-based guided local search is proposed by Repoussis, Tarantilis, Braysy, and Ioannou (2010) for the open VRP.

In this paper, our purpose is to present an effective solution approach to deal with the VRPSPD. The solution approach is a hybridization of the particle swarm optimization (PSO) and variable neighborhood descent (VND) which are well known meta-heuristics in the literature. While PSO is implemented to search good quality solutions in the solution space, VND is used to improve solutions which are randomly selected from the population in each iteration of the PSO. Moreover, an annealing-like strategy is employed to preserve the swarm diversity of the PSO. One of the key issues when designing an algorithm based on PSO lies in its solution representation, where particles bear the necessary information related to the problem domain on hand. So the most important issue in applying PSO-based algorithm to VRPSPD is to develop an effective problem mapping and solution generation mechanism. Therefore, in the h_PSO, we utilize permutation encoding that is a giant tour without trip delimiters to represent a solution of the problem. Hence, we adapt the splitting procedure proposed by Prins (2004) for the capacitated VRP in order to obtain a feasible solution from a giant tour for the VRPSPD. To the best of our knowledge, this is the first implementation of giant tours to represent VRPSPD solutions. Further difficulty in solving the VRPSPD lies in checking load feasibility which is a time consuming process. Thus, we utilize a constant-time feasibility checking procedure in the neighborhood structures implemented in VND. The proposed hybrid algorithm (h_PSO) is compared with the existing solution approaches for the VRPSPD and its special case, i.e. the VRPMPD, in the literature. The computational results reveal that the h_PSO competes with existing approaches in the literature and also improves some best known solutions of the benchmark problems.

This paper is organized as follows: Section 2 reviews studies about the VRPSPD and its special case, i.e. VRPMPD. While Section 3 includes brief information about PSO and VND, the proposed heuristic approach for the VRPSPD is given in Section 4. Section 5 gives computational results and conclusion follows in Section 6.

2. Literature review

Recently the VRP with pickup and delivery has received more attention from practitioners as well as researchers because of its great importance in practical applications and reverse logistics. The variants of the problem are: the VRP with backhauls, the VRP with mixed pickup and delivery, the VRP with simultaneous pickup and delivery, the dial-a-ride problem. In this paper, we briefly review the studies related with the VRP with simultaneous pickup and delivery and its special case, i.e. the VRP with mixed pickup and delivery. Thus, for more information about the VRP with pickup and delivery and its variants, we refer the interested readers to the review papers of Berbeglia et al. (2007) and Parragh et al. (2008).

The VRPSPD is introduced two decades ago by Min (1989) to solve a real life problem of transporting books between libraries. The solution approach for the problem consists of the following stages: (i) clients are first clustered in such a way that the vehicle capacity is not exceeded in each group; (ii) one vehicle is assigned to every cluster; and (iii) Traveling Salesman Problem (TSP) is solved for each group of clients. Dethloff (2001) propose an insertion heuristic in which customers are inserted into emerging routes according to three criteria: travel distance, residual capacity and distance from the depot. The heuristic approach does not include an improvement routine. The author considers also the relationship between the VRPSPD and other VRP variants. Nagy and Salhi (2005) develop a composite heuristic approach for single- and multi-depot VRPSPD and VRPMPD. The heuristic approach combines the power of different routines in an organized way. These routines are modified versions of VRP routines such as 2-Opt, 3-Opt, Shift, Exchange, Perturb but also some specially developed for the VRPSPD such as Reverse which is implemented to reverse the direction of a route and Neck which allows customers to be visited twice, once for delivery and then for pickup. Gajpal and Abad (2010) present the saving and parallel saving heuristics based on the Clark and Wright algorithm for the VRPSPD. The authors develop a cumulative net-pickup approach for checking the feasibility when merging two existing routes. An exact algorithm based on branch-and-price approach is developed in Dell'Amico et al. (2006) to solve the VRPSPD. This approach applies two different strategies to solve the sub pricing problem: (i) exact dynamic programming and (ii) state space relaxation. The proposed exact algorithm finds the optimum solution for instances up to 40 customers.

Meta-heuristic approaches have been also successfully applied to solve the VRPSPD and VRPMPD. Crispim and Brandao (2005) are the first authors who present a meta-heuristic approach for the VRPSPD. The proposed approach is a hybrid algorithm based on tabu search (TS) and variable neighborhood descent (VND). The authors use a sweep method to obtain an initial solution. If any route in the initial solution is infeasible because of the overloading of some intermediate arcs, the feasibility is established by exchanging the order of customers on the route. The improvement phase implements insert and swap as moves. Infeasible solutions are allowed by penalizing them according to the level of overload. Ropke and Pisinger (2006) develop a large neighborhood search (LNS) heuristic to solve several variants of the VRP including the VRPSPD. Chen and Wu (2006) also suggest a hybrid scheme incorporating the TS and the record-to-record travel strategies. In this scheme, initial feasible solutions are obtained by an insertion method based on distance- and load-based criteria. Moreover, the authors apply 2-exchange, swap, shift, 2-opt and Or-opt in the improvement phase of the hybrid algorithm. Montane and Galvão (2006) propose a heuristic approach based on TS in which shift, swap, cross and 2-opt are used as neighborhood structures. The authors have also utilized an arc frequency penalization scheme to provide balance between intensification and diversification of the search. Bianchessi and Righini (2007) present constructive and local search heuristics and also a TS algorithm using a variable neighborhood structure, in which the node-exchange-based and arc-exchange-based movements are combined. Wassan et al. (2007) develop a reactive TS with the following neighborhood structures: relocation of a customer, exchanging two customers between two different routes and reversing the route direction (reverse). In order to achieve an effective balance between the intensification and diversification of the search, the authors propose dynamic control of the tabu list size. Another hybrid algorithm based on the TS and guided local search (GLS) is proposed by Zachariadis, Tarantilis, and Kiranoudis (2009). In the hybrid algorithm, an initial solution is obtained by a construction heuristic based on cost savings and GLS is implemented for diversifying the search process. Gajpal and Abad (2009) develop a heuristic approach based on ant colony optimization (ACO). The developed heuristic has two steps: (i) the trail intensities and parameters are initialized using an initial solution obtained by means of a nearest neighborhood constructive heuristic and (ii) an ant-solution is generated for each ant using the trail intensities, followed by a local search in every ant-solution and updating of the elitist ants and trail intensities.

The most recent heuristic approaches based on meta-heuristics have been proposed by Catay (2010), Subramanian, Drummond, Bentes, Ochi, and Farias (2010), and Zachariadis, Tarantilis, and Kiranoudos (2010); Zachariadis and Kiranoudos (2011). The heuristic approach proposed by Zachariadis et al. (2010) is based on the Adaptive Memory Programming approaches. The approach stores the routes in high quality VRPSPD solutions in the Adaptive Memory, where customer sequences are periodically extracted to form new initial solutions for guiding the search. Meantime, the use of an additional memory component drives the algorithm to exploit diverse routing information stored in the Adaptive Memory and eliminates the risk of an elitistic behavior. In the approach, initial solutions are generated by using the weighted savings heuristic. Zachariadis and Kiranoudos (2011) propose a local search approach which can efficiently examine rich solution neighborhoods by statically encoding tentative moves into special data structures. This approach explores the solution space by the use of a diversification component, called promises, which is based on the aspiration criteria of tabu search. The authors also utilize the weighted savings heuristic to generate an initial solution. Subramanian et al. (2010) present a parallel algorithm which is embedded with a multi-start heuristic consisting of the VND integrated in an iterated local search framework. The main features of the proposed approach are the automatic calibration of some parameters and the ability of exploring the high-level of parallelism inherent to recent multi-core clusters. This approach implements a greedy insertion strategy to construct an initial solution for the problem. Catay (2010) proposes another heuristic approach based on ACO in which a new saving-based visibility function and pheromone updating rule are implemented. The proposed approach employs the nearest-neighbor heuristic to generate a solution which is used to initialize the pheromone trials and a local search to improve solutions obtained by ants in each iteration of the algorithm. The local search implements swap and insert moves applied between routes and within routes.

As seen from this brief review, a few attempts have been done to solve the VRPSPD by population-based solution approaches while TS is extensively used for solving the problem. Meanwhile, two main properties of the all solution approaches in the literature can be given as follows: (1) Using a heuristic approach to obtain a good initial solution or population of initial solutions and (2) Hybridizing the meta-heuristic approach with another one to improving the solutions found during search process. In recent years, particle swarm optimization (PSO), which is one of the recent and promising evolutionary meta-heuristics, has been succesfully applied to solve various VRPs. Previous researches on the application of PSO to the capacitated VRP are due to Ai and Kachitvichyanukul (2009a, 2009b), Chen, Yang, and Wu (2006), Marinakis and Marinaki (2010), and Marinakis, Marinaki, and Dounias (2010). Ai and Kachitvichyanukul (2009a, 2009b) are the first researchers who proposed a PSO approach for the VRPSPD. PSO approach of Ai and Kachitvichyanukul (2009a) implements a real-value encoding structure and improves the solution using cheapest insertion heuristic and 2-opt method applied during the route construction. This paper proposes a new solution approach based on PSO and VND for the VRPSPD. The proposed PSO approach (h_PSO) differs from Ai and Kachitvichyanukul's (2009a) PSO in terms of the definition of a particle position, the velocity model used and implementation of local search. The h_PSO implements permutation encoding to represent a solution of the problem and determines new positions of the particles using genetic operators such as crossover and mutation. Moreover, it implements an annealing-like strategy to preserve the swarm diversity of the PSO and a VND algorithm as a local search to improve the randomly selected particles in each iteration. It should be noted that these properties, i.e. representing a solution with permutation encoding and improving only randomly selected solutions by VND are also the main differences of our algorithm from the others given in the VRPSPD literature.

As the VRPMPD is a special case of the VRPSPD, the proposed h_PSO can be directly applied to solve the VRPMPD. In the literature, few researchers deal with the VRPMPD. Deif and Bodin (1984) propose a constructive heuristic based on the classical saving heuristic of Clarke and Wright. Cosco, Golden, and Wasil (1988) hybridize the saving method with load based insertion method to solve the problem. Cosco et al. (1988), Golden, Baker, Alfaro, and Schaffer (1985), and Salhi and Nagy (1999) combine Clarke and Wright saving heuristic with an insertion based heuristic. In these papers, firstly CVRP consisting only of the linehaul customers is solved and then backhaul customers are inserted in the route. Mosheiov (1998) propose a heuristic based on the tour partitioning approach. Wassan et al. (2008) propose a heuristic approach based on reactive TS approach for the VRPMPD.

3. Particle swarm optimization and variable neighborhood descent

In this section, we briefly give information about the particle swarm optimization and variable neighborhood descent metaheuristics which are two counterparts of the proposed hybrid algorithm to solve the VRPSPD.

3.1. Particle swarm optimization

Particle swarm optimization (PSO), proposed by Kennedy and Eberhart (1995), is one of the recent and promising evolutionary meta-heuristics. Its development was based on observations of the social behavior of animals such as bird flocking, fish schooling and swarm theory. PSO was originally developed to solve continuous non-linear optimization problems. However, in recent years several researchers have developed new variants of PSO to solve combinatorial optimization problem such as the traveling salesman problem (Pang, Wang, Zhou, & Dong, 2004), the vehicle routing problems (Ai and Kachitvichyanukul (2009a, 2009b); Chen et al., 2006), and the scheduling problems (Liao, Tseng, & Luarn, 2007; Liu, Wang, & Jin, 2007; Pan, Tasgetiren, & Liang, 2008; Tang & Wang, 2010; Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2007) because of its simple concept, easy implementation and quick convergence.

PSO is initialized with a population of random solutions. Each individual in the population is assigned with a randomized velocity according to its own and its companions' experiences. The individuals are called particles and flown through the hyperspace. The position of the *i*th particle in the *d*-dimensional search space at iteration *t* can be represented as $X_i^t = [x_{i1}^t, x_{i2}^t, \ldots, x_{id}^t]$ and its velocity can be denoted by $V_i^t = [v_{i1}^t, v_{i2}^t, \ldots, v_{id}^t]$. Each particle has its own best position (p_{best}) $P_i^t = [p_{i1}^t, p_{i2}^t, \ldots, p_{id}^t]$ corresponding to the personal best objective value obtained so far at iteration *t*. The global best particle (g_{best}) $P_g^t = [p_{g1}^t, p_{g2}^t, \ldots, p_{gd}^t]$ represents the best particle found so far at iteration *t* in the whole swarm. The new velocity of each particle based on the current velocity, the best experience of the particle itself (p_{best}) and that of the entire population (g_{best}) is calculated as follows:

$$v_{ij}^{t+1} = w v_{ij}^t + c_1 r_1 (p_{ij}^t - x_{ij}^t) + c_2 r_2 (p_{gj}^t - x_{ij}^t) \tag{1}$$

where *w* is called the inertia parameter, c_1 and c_2 are, respectively, cognitive and social learning parameters, r_1 and r_2 are independent random numbers generated uniformly between 0 and 1. The position of each particle is updated in each iteration using Eq. (2):

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} \tag{2}$$

In order to control excessive roaming of particles outside the search space, the velocity values are generally restricted in the range of $[-v_{\max}, v_{\max}]$. Hence, the particle flies toward a new position according to Eq. (2). This process continues until reaching a predetermined stopping criterion. The PSO velocity model given by Eqs. (1) and (2) is called global best model in which each particle flies towards its best previous position and towards the best particle in the whole swarm. In another model, called local best model, each particle flies towards its best previous position and towards the best particle in the current population (Eberhart, Shi, & Kennedy, 2001). In this paper, we apply the global best model in our hybrid algorithm.

3.2. Variable neighborhood descent algorithm

Variable neighborhood descent (VND) is an enhanced local improvement strategy which is commonly used as a subordinate in Variable Neighborhood Search (VNS) and other meta-heuristics (Hansen, Mladenovic, & Perez, 2010). The idea behind VND is to systematically change between different neighborhood structures $N_1, \ldots, N_{kmax}$. Starting with the first structure N_1, VND performs local search until no further improvements are possible. From this local optimum, it continues local search with neighborhood structure N_2. If an improved solution could be found with this structure, VND returns to using N_1 again; otherwise, it continues with N_3, and so forth. If the last structure N_{kmax} has been applied and no further improvements are possible, the solution represents a local optimum with respect to all neighborhood structures and VND terminates.

3.3. Hybridization of PSO with VND

PSO is a population-based searching technique which has a high search efficiency by combining local search (by self experience) and global one (by neighboring experience). It has some attractive characteristics: (i) all particles have knowledge of good solutions since PSO has memory and (ii) PSO has constructive cooperation between particles, and particles in the swarm share information between them. Because of the simple concept, easy implementation, and quick convergence, PSO has received increasing interest from the optimization community and wide application in different fields, especially for continuous optimization problems (Liu et al., 2007; Tang & Wang, 2010). However, PSO has following drawbacks: First, its parameter values are effective on the performance of PSO. Second is that PSO suffers from the problem of being trapped in local optima. Thus, some studies have been conducted in the literature to prevent premature convergence and to balance the exploration (i.e., searching for promising solutions within the entire region) and exploitation (i.e., searching for improved solutions in sub-regions) for achieving better performance. It is known that the performance of PSO can be greatly improved when it is hybridized with a local search algorithm. An explanation of the good performance of a combination of PSO with local search can be found in the fact that these two search methods are complementary. PSO usually performs a rather coarse-grained search. Therefore, its solutions can be locally optimized. On the other hand, a local search algorithm searches in the surroundings of its initial solution. Finding good initial solutions, however, is not easy task. Thus, this paper proposes hybridization of PSO with VND, called h_PSO, for the VRPSPD. In h_PSO, while PSO is used as a global search algorithm, VND is adopted as a local search algorithm. Since VND has stronger ability of the local search, which can overcome the disadvantages of PSO, it is expected that h_PSO exhibits better performance than VND and PSO in terms of solution quality and computation time.

In recent years, different researchers have also used the idea of hybridizing the PSO with VND to solve different combinatorial optimization problems. Pan et al. (2008) propose a hybrid algorithm based on PSO and VND to solve the no-wait flowshop scheduling problem. In this algorithm, VND consisting of two neighborhood structures called swap and insert is applied to improve the global best solution at each iteration of PSO. Czogalla and Fink (2008) propose another hybrid algorithm based on PSO and VND for the continuous flow-shop scheduling problem. The hybrid algorithm applies VND to improve the best solution of the swarm at each iteration of PSO. Kang and He (2011) hybridizes PSO and VND for meta-task assignment in heterogeneous computing systems. The hybrid algorithm generates initial population randomly and improves the global best particle found during the search process of PSO by VND based on two neighborhood structures called transfer and swap. The proposed h_PSO in this paper is different from all the above mentioned studies in terms of the application of VND as a local search algorithm. In the proposed h_PSO, VND is used to improve randomly selected solutions during the search process of the PSO.

4. Proposed hybrid algorithm

This section introduces the proposed hybrid algorithm based on PSO and VND to solve VRPSPD. As mentioned previously, the standard PSO algorithm was developed for solving continuous optimization problems. When it is applied to solve any combinatorial optimization problem, some modifications have been made on the position representation, particle velocity, and particle movement. Our hybrid algorithm (h_PSO) for the VRPSPD is based on the discrete PSO. Thus, firstly, we give brief information about the representation of solutions as particles, generating initial population and updating positions of particles and then introduce local search algorithm based on VND.

4.1. Representation

Representation is one of the important issues that affect the performance of meta-heuristics. In general, different problems have different data structures or representations. In h_PSO, we utilize giant tours, i.e. sequences of customers (called permutation encoding), without trip delimiters to represent solutions for the VRPSPD. As known, Prins (2004) is the first researcher who uses giant tours in his GA developed for the capacitated VRP. After study of Prins (2004), this representation has been successfully applied to heterogeneous VRP (Liu, Huang, & Ma, 2009; Prins, 2009). This study also extends its applicability to the VRPSPD. When this representation is used for any VRP, a decoding scheme which optimally partitions a giant tour into feasible routes is necessary. We adapt Split procedure, as a decoding scheme, developed by Prins (2004) for the capacitated VRP. In the following, the adapted Split procedure is briefly explained.

The evaluation of a given solution $X = \{1, 2, \ldots, |N_0|\}$ is concerned with partitioning customer orders along X into routes so that the maximum load of each route does not exceed vehicle capacity (Q), while the total cost required to pickup and deliver demands of all customers is minimized. An acyclic graph is constructed as follows: let $G(X)$ be a directed acyclic graph with vertex $V(G) = \{i | 0 \leqslant i \leqslant |N_0|\}$, and $E(G)$ be the set of directed arcs on $G(X)$. Each arc $(i,j) \in E(G)$ represents a feasible route, where the vehicle for route (i,j) departs from node 0 (depot) and visits nodes $i+1, i+2, \ldots, j-1$, and j, consecutively, such that maximum load (max_load) on any arc from customers $i+1$ to j does not exceed vehicle capacity. The max_load on any arc of route (i,j) is calculated using:

$$\max\_load_{X(i+1,j)} = \max[pload_{X(i+1,j)}, sload_{X(i+1,j)}] \tag{3}$$

where

$$pload_{X(i+1,j)} = \max\_load_{X(i+1,j-1)} + d_{X(j)} \tag{4}$$

$$sload_{X(i+1,j)} = \sum_{k=i+1}^{j} p_{X(k)} \tag{5}$$

The cost of arc/route (i,j) is calculated by

$$z_{ij} = c_{0,X(i+1)} + \sum_{k=i+1}^{j-1} c_{X(k),X(k+1)} + c_{X(j),0} \tag{6}$$

A minimum cost-path from node 0 to node $|N_0|$ in $G(X)$ defines the optimal partition of the chromosome. An optimal partitioning can be found in $O(|N_0|^2 \log(|N_0|))$ time. Nevertheless, shortest path problem can be solved in $O(|N_0|^2)$ or even $O(|N_0|)$ by using faster algorithms (Prins, 2004).

Fig. 1 depicts an example for a VRPSPD. There are 5 customers and vehicle capacity is equal to 15. The number on arc (i,j) denotes the cost of the arc, and the numbers on node j represent the delivery and pickup demands, respectively. The chromosome to be partitioned is X = 1, 2, 3, 4, 5. Each feasible route extractible from X is modeled by one arc in acyclic graph $G(X)$ (see, Fig. 1b). For example, arc (2,4) in Fig. 1b represents the route (0,3,4,0) with a total cost of 95. Fig. 1c gives the optimal partition into routes, which is obtained by considering the arcs of the shortest path (boldface) in $G(X)$.

In this study, we revise the algorithm given in Prins (2004) to optimally partition the chromosome, i.e. X, to obtain a VRPSPD solution. The revised algorithm for the VRPSPD is given in Fig. 2. The first part of this algorithm computes two labels (i.e. V_j and P_j) for each node j = 1, 2, ..., $|N_0|$ of X instead of generating $G(X)$ explicitly. While V_j represents the cost of the shortest path from node 0 to node j in $G(X)$, P_j is the predecessor of node j on this path. The repeat loop updates V_j and P_j after enumerating all feasible sub-sequences $X(i)$, ..., $X(j)$. It should be noted that for a given node i, j is increased until maximum load (i.e. max_load) of the vehicle exceeds the vehicle capacity (Q). Hence, $V_{|N_0|}$ gives the cost

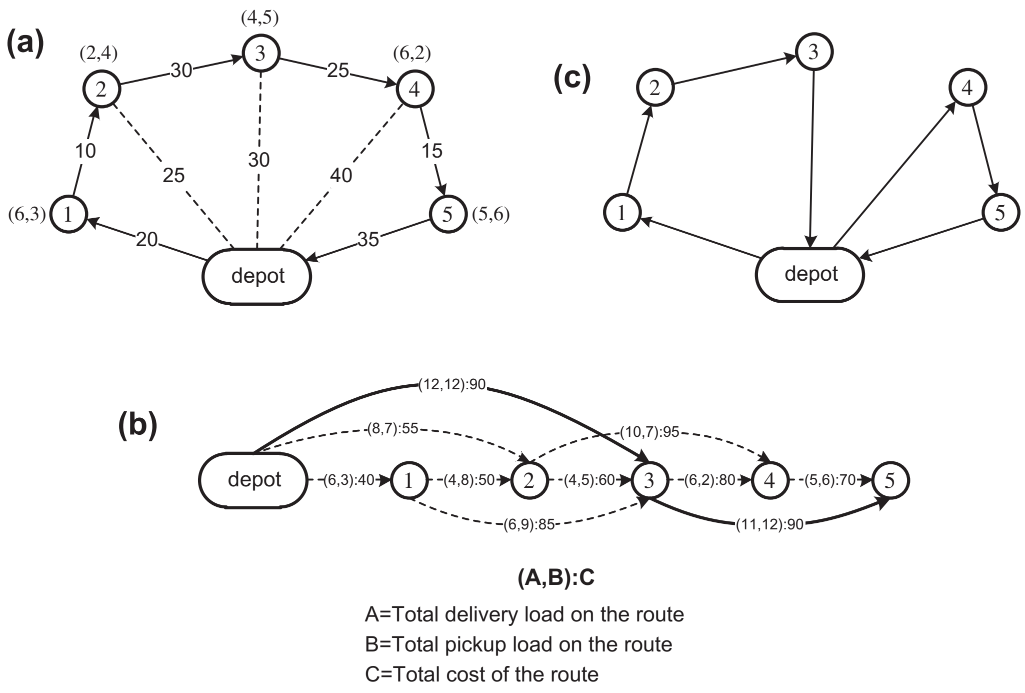

Fig. 1. Decoding of a chromosome. (a) A chromosome X = (1, 2, 3, 4, 5), (b) acyclic graph, and (c) optimal partitioning.

```
Procedure 1: Decoding of chromosome for the VRPSPD
Input: Giant tour $X$
Output: Vehicle routes for $X$
begin
// Creating two labels (i.e. $V_j$ and $P_j$) related shortest path from node 0 to node $j$, $j = 1,2,\dots,|N_0|$ //
    $V_0 \leftarrow 0$; $V_j \leftarrow \infty$  $\forall j \in \{1, 2, \dots, |N_0|\}$
    for $i$=1, 2, ... , $|N_0|$ do begin
        $k \leftarrow 0$
        $max\_load \leftarrow 0$; $pload \leftarrow 0$; $sload \leftarrow 0$
        $j \leftarrow i$
        repeat
            $pload \leftarrow \max\_load + d_{X(j)}$;
            $sload \leftarrow sload + p_{X(k)}$
            if $i = j$ then $cost \leftarrow c_{(0,X(i))} + c_{(X(i),0)}$
                else $cost \leftarrow cost - c_{(X(i),0)} + c_{(X(i),X(j))} + c_{(X(j),0)}$
            if $max\_load \le Q$ then begin
                if ($V_{i-1} + cost < V_j$) then begin
                    $V_j \leftarrow V_{i-1} + cost$
                    $P_j \leftarrow i$-1
                end
            $j \leftarrow j + 1$
            end
        until $j > |N_0|$ or $max\_load > Q$
    end
//Listing routes of the VRPSPD using the vector of labels P//
    for $i$ =1, 2, ...,$|N_0|$ do $route(i) \leftarrow \varnothing$
    $r \leftarrow 0$; $j \leftarrow |N_0|$
    repeat
        $r \leftarrow r + 1$; $i \leftarrow P_j$
        for $k = i + 1$ to $j$ do $route(r) \leftarrow route(r) \cup \{X(i)\}$ end for
        $R \leftarrow R \cup route(r)$
        $j \leftarrow i$
    until $i$=0
    return the set of vehicle routes, $R$
end
```

Fig. 2. The algorithm to obtain optimal partition of a giant tour.

of optimal partition of the chromosome. The second part of the algorithm extracts the VRPSPD solution using the vector of labels *P*. Each route consists of a list of customers. In worst case, it builds *n* routes, i.e. one route per customer.

4.2. Generating the initial population

Usually, the initial population is generated randomly. However, selecting a suitable initial population accelerates the convergence of the PSO. To reach optimal/near optimal solution, therefore, we generate an initial population including heuristic solutions as well as random solutions to explore the different regions of solution space. In h_PSO, while nearest neighborhood heuristic (NNH) is employed to generate the first giant tour of the population, others are randomly generated. The NNH is applied for the VRPSPD as follows: a vehicle starts from the depot and visits the nearest customer in terms of cost. If visiting the nearest customer is feasible, i.e. max_load does not exceed the vehicle capacity, that customer is added to route, otherwise the vehicle returns to the depot, and another vehicle starts from the depot. This procedure continues until all customers are visited and a heuristic solution is obtained. It is worthy to note that this solution is converted into a giant tour by connecting the routes directly. It should be noted that to diversify the initial population, other giant tours are randomly generated in such a way that clones are forbidden and also their tour costs differ from each other at least Δ value. Based on our preliminary experiments, Δ is set to 2.

4.3. Position Updating Rule

Since possible solutions for the VRPSPD are represented by giant tours (i.e. using permutation encoding) in the proposed algorithm, the velocity model and position updating rule given for real values in standard PSO are no longer be applicable for our representation. Different position updating rules for permutation encoding are proposed in the literature (see for example, Liao et al., 2007; Pan et al., 2008). In this paper, we implement a position updating rule proposed by Pan et al. (2008) for the no-wait flowshop scheduling problem. According to this rule, the position of a particle at iteration *t* is updated by considering possible three choices. These are: (i) following its own position (X_i^t), (ii) going towards its personal best position (P_i^t), and (iii) going towards the best position of the particle in the entire swarm population (P_g^t). Thus, the updating equation is given as follows:

$$X_i^{t+1} = c_2 \otimes F_3\left(c_1 \otimes F_2\left(w \otimes F_1\left(X_i^t\right), P_i^t\right), P_g^t\right) \quad (7)$$

This equation consists of three components. The first component, $\lambda_i^{t+1} = w \otimes F_1(X_j^t)$, represents the velocity of the particle. In this component, F_1 is the mutation operator applied to X_i^t with probability of w and λ_i^{t+1} is obtained as follows:

$$\lambda_i^{t+1} = \begin{cases} F_1(X_i^t) & \text{if } u \leqslant w \\ X_i^t & o.w. \end{cases} \quad (8)$$

where $u \approx U(0,1)$.The second component, $\delta_i^{t+1} = c_1 \otimes F_2(\lambda_i^t, P_i^t)$, is related with the private thinking of the particle, where F_2 represents the crossover operator applied to λ_i^t with the probability of c_1. δ_i^{t+1} is obtained as follows:

$$\delta_i^{t+1} = \begin{cases} F_2(\lambda_i^t, P_i^t) & \text{if } u_1 \leqslant c_1 \\ \lambda_i^t & o.w. \end{cases} \quad (9)$$

where $u_1 \approx U(0,1)$. Finally, $X_i^{t+1} = c_2 \otimes F_3(\delta_i^t, P_g^t)$ models the collaboration of the particles, where F_3 represents the crossover operator applied to δ_i^t with the probability of c_2. X_i^{t+1} is obtained as follows:

$$X_i^{t+1} = \begin{cases} F_3\left(\delta_i^t, P_g^t\right) & \text{if } u_2 \leqslant c_2 \\ \delta_i^t & o.w. \end{cases} \quad (10)$$

where $u_2 \approx U(0,1)$. In our implementation, we utilize following well-known mutation and crossover operators developed for permutation encoding in the literature: swap and partially mapping crossover (PMX) operators (Gen & Cheng, 1997). These operators are applied to giant tours.

4.4. Local search

h_PSO implements variable neighborhood descent (VND) algorithm as a local search. VND improves some randomly selected solutions which are obtained by the position updating rule. After that personal and global bests of the swarm are updated. In our implementation, we utilize six neighborhood structures which are well-known in the VRP literature. These neighborhood structures are briefly explained as follows:

Crossover: This structure divides routes r_1 and r_2 into two, namely first part and second part. Then the first part of r_1 is connected to second part of r_2, while the first part of r_2 is connected to second part of r_1. Every pair of routes and every division point are considered to select best improved neighborhood.

Swap(1, 1): In this structure, a customer cs_1 from route r_1 and a customer cs_2 from route r_2 are removed. Then customers cs_1 and cs_2 are put to their best position in r_2 and r_1, respectively. Finding best position for every pair of customers belong to different routes is a time consuming process. Therefore, we use following two criteria to save computation time of this operation: Firstly, k nearest customers of customers cs_1 and cs_2 are determined. Then swapping operation between customers cs_1 and cs_2 is performed if at least one of the k nearest customers of customer cs_1 is in the route r_2 and at least one of the k nearest customers of customer cs_2 is in the route r_1 (Gajpal & Abad, 2009). Secondly, we implement a tabu list and keep customers cs_1 and cs_2, which are justly swapped, in this list to forbid cycling phenomena. In our implementation, k is set to 10 and the length of the tabu list is taken as 7.

Shift(1, 0): This structure removes a customer cs_i from its current route r_1 and relocates it at a position in another route r_2. Every possible insertion position for every customer is considered.

2-Opt: This structure reverses the direction of a path lying between the customers cs_{i-1} and cs_j by replacing non-adjacent arcs (cs_{i-1}, cs_i) and (cs_{j-1}, cs_j) belonging to the same route with (cs_{i-1}, cs_{j-1}) and (cs_i, cs_j) respectively.

Exchange: This neighborhood structure exchanges the position of two customers in the same route.

Reverse: This structure reverses the route direction if the maximum load carried through the corresponding route is reduced. In fact as one can conclude, the distance of the route does not change when its direction is reversed but the decrease in the maximum load will give an opportunity of assigning new customers to the route in proceeding iterations.

As seen in the definitions of the neighborhood structures, while first three structures are applied between different routes, others perform search inside the routes. Fig. 3 illustrates the application of the neighborhood structures. When a VND algorithm is used as a local search, the order of the neighborhood structures affects the performance of the algorithm. Based on our preliminary experiments, we implement the neighborhood structures in their explanation order given above. It is important to note that the first three structures used between routes are perfomed to obtain a new improved solution from the current solution (s). In the case of the improvement, the last ones implemented within routes are applied to obtain further improvement on the solution. A variant of this approach was also implemented in Subramanian et al. (2010). The proposed VND is applied to a real solution which is obtained by decoding corresponding giant tour as explained in Section 4.1 and only feasible movements, in which the vehicle capacity constraint is not violated, are considered. Therefore, whenever an improvement exists, the feasibility of the solution is checked. Nevertheless, the feasibility checking process in the VRPSPD has an additional complexity because of the load fluctuation of vehicles. In order to speed up the feasibility checking process, we utilize special metrics proposed by Bianchessi and Righini (2007). These metrics capture the load fluctuation of vehicles along their routes. The proposed VND algorithm for the VRPSPD is given in Fig. 4.

As mentioned previously, the personal and global bests are updated after randomly selected solutions are improved by the proposed VND. Our preliminary experiments show that the swarm diversity decreases gradually if each personal best is updated whenever an improvement occurs. It is known that the swarm diversity is affected on the swarm exploration ability. Therefore, we utilize an annealing-like strategy for updating personal best of each particle. According to this strategy, Eq. (11) is used to update personal best.

$$P_i^t = \begin{cases} X_i^t & eger\ f(X_i^t) \leqslant f\left(P_i^{t-1}\right) \\ X_i^t & eger\ f(X_i^t) > f\left(P_i^{t-1}\right) ve\ u \leqslant e[-(\Delta s/T)] \\ P_i^{t-1} & d.d. \end{cases} \quad (11)$$

where T is the temperature and $\Delta s = [(f(X_i^t) - f(P_i^{t-1}))/f(P_i^{t-1})] \cdot 100$. In this approach, a worst solution has a high probability to be selected as a personal best in the initial iterations of the algorithm. Thus, this approach initially increases the search ability of the algorithm by providing high diversification in the swarm. Since the temperature decreases gradually in each iteration, the probability of a worst solution to be selected as a personal best also decreases. Hence, the search can be intensified around the good solutions at the last iterations of the algorithm. We utilize geometric cooling schedule in order to reduce temperature in each iteration of the algorithm, i.e. $T^t \leftarrow \alpha T^{t-1}$.

4.5. The steps of the h_PSO

The details of the proposed h_PSO for solving the VRPSPD are presented in Fig. 5. The algorithm starts with an initial swarm of PS particles, where PS is the number of particles in the swarm. Each particle corresponds to a candidate solution to the underlying problem. Then, all of the particles repeatedly move until a stopping condition is satisfied. At each iteration, the personal best particles and global best position achieved so far are determined. The particle adjusts its position based on the individual experience and the swarm's intelligence as described in Eq. (7). To speed up the convergence of h_PSO, VND is applied to solutions which are selected from the current swarm with the

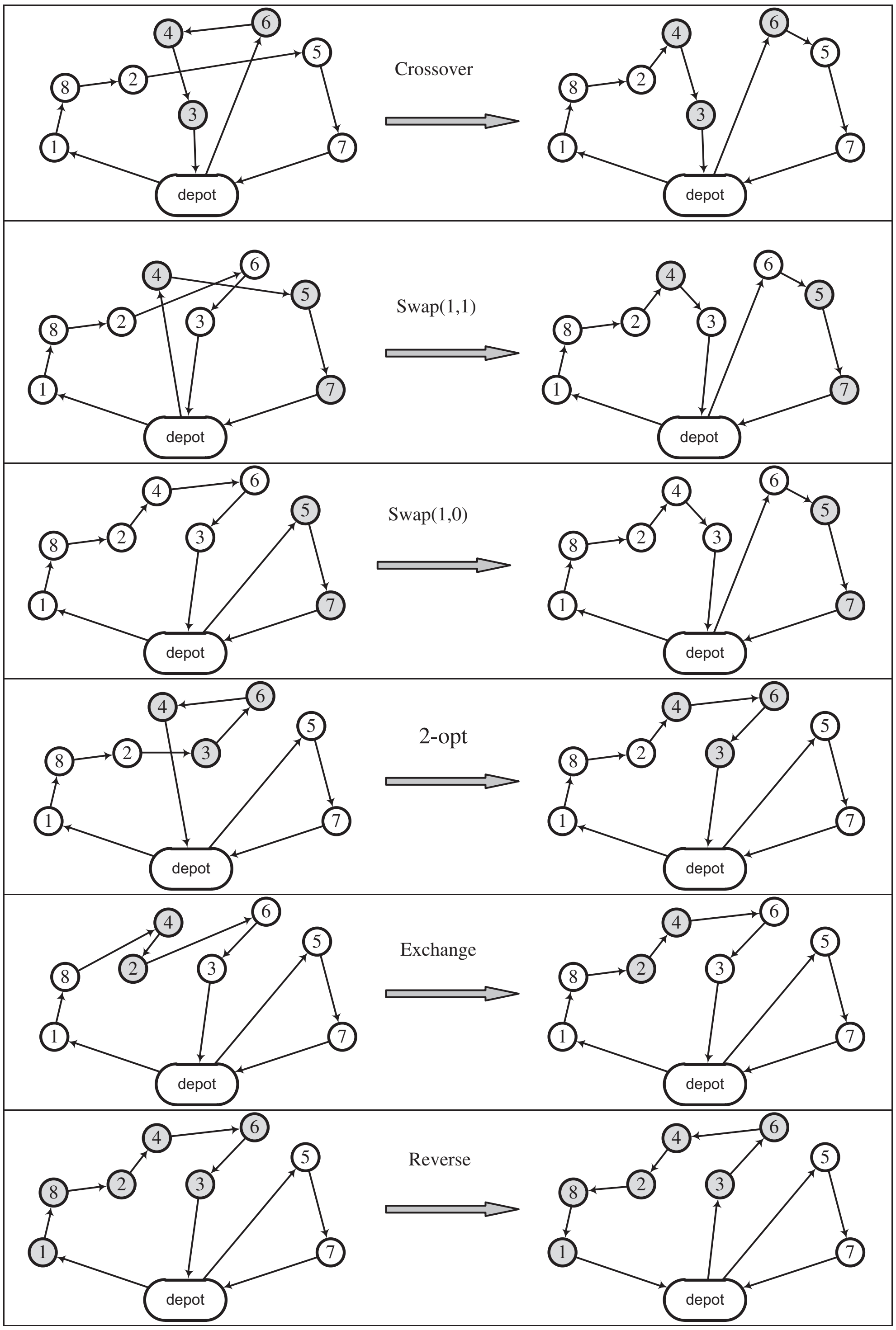

Fig. 3. Neighborhood Structures.

probability of p_{VND} before updating the personal and global bests. When the algorithm is terminated, the global best particle and the corresponding fitness value are considered as a solution for the VRPSPD.

5. Experimental study

We conduct two sets of computational experiments to investigate the effectiveness of h_PSO. The first set compares h_PSO with the algorithms proposed for the VRPSPD in the literature. As mentioned in the literature (see, Wassan, Wassan et al. (2007), Wassan, Nagy, et al. (2008)), the VRPMPD (i.e. VRP with mixed pickup and delivery) is a special case of the VRPSPD. Therefore, the proposed h_PSO can be directly applied to solve the VRPMPD. The second set evaluates the performance of h_PSO against the existing algorithms in the literature for the VRPMPD. The h_PSO is coded in Visual C. All experiments have been performed on Intel Xeon 3.16 GHz equipped (with 1 GB RAM).

```
Procedure 2: Local Search (VND)
Input: Vehicle routes for the VRPSPD
Output: Improved vehicle routes for the VRPSPD
begin
   $k \leftarrow 1$
   while $k \leq nk$ do begin
     find the best neighbor $(s')$ of $s' \in N(s)$ by using kth neighborhood structure
     if $f(s') < f(s)$ then
         begin
           $s \leftarrow s'$, $f(s) \leftarrow f(s')$ and $k \leftarrow 1$
           apply 2-opt to $(s)$ and obtain $(s')$
                 if $f(s') < f(s)$ then $s \leftarrow s'$ and $f(s) \leftarrow f(s')$
           apply exchange to $(s')$ and obtain $(s'')$
                 if $f(s') < f(s)$ then $s \leftarrow s'$ and $f(s) \leftarrow f(s')$
           apply reverse to $(s'')$ and obtain $(s''')$
                 if $\max\_load(s''') < \max\_load(s'')$ then $s \leftarrow s'''$
         end
         else $k \leftarrow k+1$
   end
   return s
end
```

Fig. 4. The proposed VND algorithm.

```
Algorithm :  h_PSO for the VRPSPD
Input     :  VRPSPD data, h_PSO parameters
Output    :  best solution
begin
   $t \leftarrow 0$;
   Generate initial population of swarm, $X^t$
   Call Procedure 1 to evaluate each particle in $X^t$
   Determine the personal best of each particle in $P^t$
   Determine the global best of swarm, $G^t$
   while (stopping criterion is not satisfied) do
       begin
          $t \leftarrow t + 1$
          Update particles using equation (7)
          Call Procedure 1 to evaluate each particle in $X^t$
          for i = 1,…, $PS$ do
             begin
                $u \sim (0,1)$
                if $u < p_{VND}$ then Call Procedure 2 to improve $X_i^t$
                update personal best $P_i^t$ using equation (11)
             end
          update global best $G^t$
          $T \leftarrow \alpha T$
       end
   Return the global best $G^t$ and its fitness function
end
```

Fig. 5. The proposed h_PSO algorithm.

5.1. Computational Analysis for the VRPSPD

This subsection presents the computational results of the proposed h_PSO for the VRPSPD.

5.1.1. Test problems and parameter setting for h_PSO

We investigate the performance of h_PSO using two sets of problem instances available in the literature for the VRPSPD. The first data set is provided by Dethloff (2001). This set consists of 40 problem instances with 50 customers. Instances are classified into four sets, namely SCA3, SCA8, CON3 and CON8. SCA data sets are generated with customers scattered uniformly in the service region of $100 * 100$. CON data sets are generated with half of the customers located uniformly in the service region and the other half are concentrated in the interval [100/3, 100/3]. The delivery demand of customer i (d_i) is generated from uniform distribution in the interval [0,100] and then pickup demand (p_i) is calculated from the eguation of $p_i = (0.5 + r_i)d_i$, where r_i is a random number between [0,1]. The numbers 3 and 8 after SCA or CON indicate the parameter for determining vehicle capacity. There are 10 problems in each group of problems.

The second data set is proposed by Salhi and Nagy (1999). This data set has been adapted from seven problems orginally proposed by Christofides, Mingozzi, and Toth (1979) for the VRP with capacitated VRP, involving from 50 to 199 customers. The cost matrix is obtained by calculating the Euclidean distances between vertices. Salhi and Nagy (1999) utilize following scheme to modify capacitated VRP instances for the VRPSPD: x_i and y_i coordinates for customer i are considered as in the original problem and a ratio r_i is calculated as $r_i = \min(x_i/y_i, y_i/x_i)$. Let dm_i be the demand of customer i. To obatin the first 7 VRPSPD instances, namely X-series, the delivery and pickup quantities for customer i are calculated as $d_i = r_i \cdot dm_i$ and $p_i = (1 - r_i) \cdot dm_i$, respectively. The other seven VRPSPD instances, namely Y-series, are generated by swapping the d_i and p_i values for every customer (Gajpal & Abad, 2009; Subramanian et al., 2010; Zachariadis et al., 2010).

The parameters of h_PSO are: swarm size (PS), initial temperature (T_0), mutation rate (w), crossover rates (c_1 and c_2), probability of VND (p_{VND}). Firstly we investigate the effects of VND and population size on the solution quality and computation time of h_PSO. We consider two levels for PS as 20 and 60, and three levels for p_{VND} as 0.1, 0.3 and 0.5. In order to determine their best combination, we solve 5 instances randomly selected from the data sets for each combination of PS and p_{VND}. Table 1 reports average solution, which is calculated over 5 instances, and average computation time (ACT) for each combination. As seen from Table 1, increasing the population size from 20 to 60 improves solution quality slightly while causing excessive increases in computation times (i.e. more than two times higher than those obtained when $PS = 20$). Meantime, when p_{VND} is set to 0.30, the h_PSO produce slightly better solutions than others (i.e. obtained with $p_{VND} = 0.10$ or 0.50) regardless of population size. Thus, in order to obtain a competitive algorithm with others in the existing literature, we set $PS = 20$ and $p_{VND} = 0.3$ for h_PSO.

We also utilize an experimental design approach to investigate the performance of h_PSO for a set of remaining four parameters. Three levels are selected for each parameter: $T_0 = 10$, 20 and 30, $w = 0.1$, 0.2 and 0.3, and c_1 and $c_2 = 0.6$, 0.8 and 1. Hence the experimental design includes 3^4 design points. Five instances taken from data set of Dethloff (2001) are solved for each design point, resulting 405 observations. At the level of 0.05, statistical analysis performed using ANOVA shows that main effects of the parameters are not significant, while the interaction between mutation and crossover rates are significant. Thus, we set following values for the mutation and crossover rates by considering their interactions: $w = 0.2$, and c_1 and $c_2 = 0.6$. Meantime, T_0 is set to 30.

The proposed h_PSO is run ten times with different random number seeds for each problem instance on data sets to gauge its natural variability. We apply two stopping criteria in the algorithm: (1) the number of successive iterations (n_suc) in which the global best solution is not updated and (2) upper bound of the computation time (CUB). In our implementation, n_suc and CUB are set to 350 iterations and 500 s, respectively.

Table 1
The computational results for swarm siz and probability of VND.

Probability of VND (p_{VND})	Swarm size			
	$PS = 20$		$PS = 60$	
	Average solution	ACT	Average solution	ACT
0.1	844.03	4.8	843.28	9.7
0.3	841.80	21.9	839.83	50.9
0.5	842.60	29.5	840.96	72.3

Before the computational results, it is important to give information about the impact of the annealing-like strategy on the performance of the h_PSO, Thus, we run h_PSO with annealing-like strategy (called h_PSO with SA) and without annealing-like strategy (called h_PSO without SA) on a test instance coded SCA8-9 in Dethloff's data test. Fig. 6 shows the convergence of the h_PSO with SA and h_PSO without SA. As seen from this figure, the h_PSO with SA slowly convergences better solutions than h_PSO without SA. This result supports that annealing-like strategy helps the swarm exploration ability of the algorithm to prevent local optima.

5.1.2. Computational results

In order to investigate performance of the proposed h_PSO for the VRPSPD, we compare it following algorithms presented in the literature:

RTS: Reactive tabu search (Wassan et al., 2007).
LNS: Large neighborhood search (Ropke & Pisinger, 2006).
PSO: Particle swarm optimization (Ai & Kachitvichyanukul, 2009a).
ACS: Ant colony system (Gajpal & Abad, 2009).
PILS: Parallel iterative local search (Subramanian et al., 2010).
AMM: Adaptive memory methodology (Zachariadis et al., 2010).

PSO and ACS are considered in the comparison with the purpose of drawing a conclusion about performances of population-based heuristics (i.e. PSO, ACS and h_PSO) on the VRPSPD. It is also important to test the performance of h_PSO against the PILS and AAM since they are the most effective heuristics recently proposed in the VRPSPD literature.

Since these algorithms have been run on different computers, it is difficult to compare their CPU times. Thus, we utilize table of Dongarra (2007) to estimate relative performance of different computers used in those studies. Table 2 presents Mflops (million floating point calculation per second) values for the computers related with this study. The Mflops values are used to scale running time of an algorithm relative to our computer.

5.1.2.1. Results of h_PSO for Dethloff's (2001) data set.

Table 3 compares the results of h_PSO and PSO over four data sets, namely SCA3, SCA8, CON3 and CON8. Ai and Kachitvichyanukul (2009a) report the average of the best solutions over the 10 instances of each data set in their study. Thus, in order to make a direct comparison, we present our results considering these four data sets. As seen

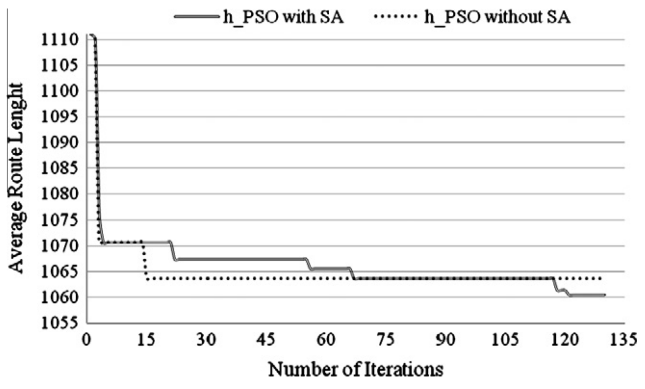

Fig. 6. Convergence of h_PSO with SA and without SA.

Table 2
Mflops values for different computers and conversion ratio (r).

Algorithm	Computer	Equivalent computer[a]	Mflops	r[*]
h_PSO	Intel Xeon 3.16 GHz	Intel Xeon 3.16 GHz	981	1
RTS	Sun Fire V440 server–Ultra SPARC-IIII 1062 MHz	IBM eServer pSeries 630 6C4 1 proc(1000 MHz)	842	0.86
LNS	Pentium IV 1.5 GHz	Intel Pentium M 1.6 GHz	326	0.33
PSO	Intel P4 3.4 GHz	Intel Pentium 4 (2.8 Ghz)	1317	1.34
ACS	Intel Xeon 2.4 GHz	Intel Xeon 2.4 GHz	884	0.90
AMM	Intel P4 2.4 Ghz	Intel Pentium 4 (2.53 GHz)	1033	1.05
PILS[*]	Intel Xeon 2.66 GHz (256 P.I.)	[**]	[**]	[**]

[a] An equivalent computer given in the report for the corresponding computer.
[*] Conversion ratio of the Mflops value of a given computer to the Mflops value of our computer.
[**] PILS has been run in a cluster with a multi-core architecture using 256 cores.

Table 3
Comparison of h_PSO with PSO on VRPSPD instances of Dethloff (2001).

Data set	PSO Average best solution	h_PSO Average best solution
SCA3	675.80	673.40
SCA8	1041.80	1028.15
CON3	569.60	560.95
CON8	798.30	771.65

from Table 3, h_PSO performs better than PSO and it improves the best results of PSO around 1.63% on average.

Table 4 presents the computational results of LNS, ACS, AMM, PILS and h_PSO over 40 instances. It should be noted that reported best solution (Best sol.) and average computation time (ACT) in seconds for each algorithm are the best one and average over multiple runs, respectively. While LNS, ACS, AMM and our h_PSO run 10 times for each instance, PILS run 20 times for each instance. Bold numbers given for each algorithm in the table indicate that the algorithm has reached the best solution. Table 4 reports also average deviation to the best known solution (BKS) in percent (Avg. deviation), number of BKS found, average computation time (Avg. time) and scaled average computation time (Avg. scaled time) over 40 instances in last four rows. From Table 4, it is seen that AMM, PILS and h_PSO reach same results for 40 instances. However, LNS and ACS fail to reach best results for 18 and 2 out of 40 instances, respectively. The average percentage deviations of the solutions of LNS and ACS from the best solutions are around 0.32% and 0.01%, respectively. These results show that algorithms considered in the comparison are very effective to produce less than 1% deviation. It is also important to note that performance comparison only in terms of computation time is not recommended in the literature because of the other influencing factors such as hardware (i.e. cache, main memory and compiler), software and coding. Nevertheless, we can see from Table 4 that h_PSO and AMM need less computational burden to reach best known solution than others. As mentioned previously, the PILS has been run in a cluster with a multi-core architecture using 256 cores. Thus, we ignore its computational times when comparing the algorithms in terms of computational burden.

5.1.2.2. Results of h_PSO for Salhi and Nagy's (1999) data set. Table 5 sumarizes the computational results of RTS, LNS, PSO, ACS, AMM, PILS and h_PSO over 14 instances. As in Table 4, bold numbers for each algorithm in Table 5 indicate that the algorithm has reached the best solution. As seen from the table, PILS is the best to attain 9 out of 14 best solutions compared with other algorithms. While our algorithm h_PSO reaches six best solutions out of 14 possible solutions, RTS and AMM produce five best solutions. Others, i.e. LNS, PSO and ACS, generate only two best solutions for the data set. Regarding the percentage deviation of solution of each algorithm from the best solution, it is seen that PILS, h_PSO, AMM and ACS produce less than 1% deviation. Our algorithm h_PSO having deviation of 0.71% is the second best after PILS with deviation of 0.65% while PSO has the largest deviation (i.e. 6.09%) among the other algorithms. Additionally, h_PSO performs better than RTS and LNS and also competes with PSO in terms of computational burden.

Meantime, we perform statistical analysis using a paired-t test to investigate whether there are statistically significant differences between h_PSO and other heuristic approaches according to solution quality or not. Since it is expected that the h_PSO would give the solutions with lower route length than other heuristics, a one sided alternative hypothesis, H_1, is used as given below:

$$H_1 : \mu_{h\_PSO} - \mu_{HM} < 0 \tag{12}$$

where μ_{h_PSO} and μ_{HM} are population mean for h_PSO and HM, respectively. HM refers to the heuristic approach used in statistical comparison. For example, if h_PSO is compared with RTS, then HM will be RTS. Table 6 shows pairs, mean differences for Nagy and Salhi's test problems and p-values at significance level of $\alpha = 0.05$. As is seen from the table, while h_PSO is statistically significant different from LNS, PSO and ACS with p-value of 0.008, 0.001 and 0.030 and mean differences of −29.428, −45.928 and −1.500, respectively, there is no statistically significant difference between h_PSO and RTS, AMM and PILS. This analysis indicates that the proposed h_PSO performs as well as RTS, AMM and PILS which are the most effective heuristics recently proposed in the literature, and also outperforms existing population-based heuristics, i.e. PSO and ACS, in the VRPSPD literature.

5.2. Computational analysis for the VRPMPD

As mentioned previously, the proposed h_PSO can be directly applied to solve the VRPMPD which is a special case of the VRPSPD. Thus, in this subsection, we compare our algorithm h_PSO with the existing algorithms in the literature for the VRPMPD.

5.2.1. Test problems and parameter setting for h_PSO

We utilize two sets of problem instances available in the literature to investigate the performance of h_PSO for the VRPMPD. The first data set is provided by Salhi and Nagy (1999). They have generated three problem subsets referred to as T, Q and H from the seven problem instnaces of capacitated VRP. In set T, every 10th customer is considered as a pickup customer and is assigned pickup demend equal to the original delivery demand. In a similar fashion, every 4th and every 2nd customer are considered as pickup customers in sets Q and H, respectively. While the proportion of pickup customers is 10% in set T, this value increases to 25% and 50% in sets Q and H, respectively. These three sets, i.e. T, Q and H, are referred as CMTT, CMTQ and CMTH, respectively. This set includes totally 21 test instances.

The second data set has been derived by Wassan et al. (2008) from the VRPSPD data set of Dethloff (2001). Similar to Salhi and Nagy's (1999) approach, they create three mixed instances from each VRPSPD instance by setting every 10th, 4th and 2nd customer to be a pickup customer. The new instances are distiguished by adding the letter T, Q or H after the original instance designator. This set consists of a total of 120 instances in which each subclass has 40 instances.

It is important to note that the parameter settings of h_PSO used for the VRPSPD are also implemented for the VRPMPD.

Table 4
Comparison of h_PSO with LNS, ACS, AMM and PILS on the VRPSPD instances of Dethloff (2001).

Instance	LNS		ACS		AMM		PILS		h_PSO	
	Best sol.	ACT	Best sol.	ACT	Best sol.	ACT	Best sol.	ACT	Best sol.	ACT
SCA3-0	636.1	232	**635.62**	6.0	**635.62**	2.5	**635.62**	2.3	**635.62**	4.9
SCA3-1	**697.8**	218	**697.84**	6.0	**697.84**	2.5	**697.84**	2.3	**697.84**	0.8
SCA3-2	**659.3**	203	**659.34**	6.0	**659.34**	2.9	**659.34**	2.1	**659.34**	0.4
SCA3-3	680.6	241	**680.04**	6.1	**680.04**	2.3	**680.04**	2.5	**680.04**	1.0
SCA3-4	**690.5**	208	**690.50**	5.7	**690.50**	2.9	**690.50**	2.2	**690.50**	0.3
SCA3-5	**659.9**	226	**659.90**	5.1	**659.90**	3.0	**659.90**	2.2	**659.90**	2.0
SCA3-6	**651.1**	233	**651.09**	6.1	**651.09**	3.1	**651.09**	2.5	**651.09**	0.8
SCA3-7	666.1	206	**659.17**	6.8	**659.17**	2.8	**659.17**	2.4	**659.17**	1.6
SCA3-8	**719.5**	190	**719.47**	5.4	**719.47**	3.5	**719.48**	2.3	**719.47**	0.5
SCA3-9	**681.0**	220	**681.00**	6.0	**681.00**	4.7	**681.00**	1.9	**681.00**	0.8
SCA8-0	975.1	98	**961.50**	11.0	**961.50**	2.7	**961.50**	3.4	**961.50**	4.8
SCA8-1	1052.4	95	**1049.65**	11.5	**1049.65**	3.8	**1049.65**	2.9	**1049.65**	6.8
SCA8-2	1044.5	94	1042.69	11.9	**1039.64**	3.9	**1039.64**	2.4	**1039.64**	10.2
SCA8-3	999.1	94	**983.34**	11.3	**983.34**	2.6	**983.34**	3.0	**983.34**	13.0
SCA8-4	**1065.5**	93	**1065.49**	11.1	**1065.49**	2.4	**1065.49**	2.8	**1065.49**	3.0
SCA8-5	**1027.1**	96	**1027.08**	11.3	**1027.08**	3.4	**1027.08**	3.3	**1027.08**	4.1
SCA8-6	977.0	94	**971.82**	12.0	**971.82**	2.7	**971.82**	3.5	**971.82**	1.6
SCA8-7	1061.0	92	1052.17	12.5	**1051.28**	5.1	**1051.28**	3.1	**1051.28**	3.4
SCA8-8	**1071.2**	98	**1071.18**	11.0	**1071.18**	3.6	**1071.18**	2.9	**1071.18**	0.8
SCA8-9	**1060.5**	92	**1060.50**	11.5	**1060.50**	4.8	**1060.50**	2.2	**1060.50**	7.3
CON3-0	**616.5**	215	**616.52**	8.3	**616.52**	4.7	**616.52**	3.1	**616.52**	2.1
CON3-1	**554.5**	245	**554.47**	7.1	**554.47**	2.2	**554.47**	2.8	**554.47**	1.3
CON3-2	521.4	232	**518.00**	6.9	**518.00**	3.1	**518.00**	2.8	**518.00**	1.3
CON3-3	**591.2**	231	**591.19**	7.2	**591.19**	3.2	**591.19**	2.3	**591.19**	0.5
CON3-4	**588.8**	221	**588.79**	6.0	**588.79**	2.3	**588.79**	2.6	**588.79**	3.2
CON3-5	**563.7**	209	**563.70**	6.9	**563.70**	3.7	**563.70**	2.7	**563.70**	0.4
CON3-6	500.8	225	**499.05**	7.3	**499.05**	3.7	**499.05**	2.8	**499.05**	2.3
CON3-7	**576.5**	227	**576.48**	7.0	**576.48**	1.9	**576.48**	2.8	**576.48**	2.6
CON3-8	**523.1**	237	**523.05**	7.4	**523.05**	3.8	**523.05**	2.5	**523.05**	1.0
CON3-9	586.4	207	**578.25**	6.8	**578.25**	2.2	**578.25**	3.4	**578.25**	2.9
CON8-0	**857.2**	94	**857.17**	12.3	**857.17**	4.4	**857.17**	3.7	**857.17**	5.2
CON8-1	**740.9**	94	**740.85**	12.0	**740.85**	3.3	**740.85**	3.0	**740.85**	2.9
CON8-2	716.0	94	**712.89**	13.0	**712.89**	2.7	**712.89**	3.1	**712.89**	2.1
CON8-3	**811.1**	98	**811.07**	13.9	**811.07**	2.8	**811.07**	4.0	**811.07**	2.8
CON8-4	**772.3**	95	**772.25**	11.9	**772.25**	2.8	**772.25**	3.7	**772.25**	3.6
CON8-5	755.7	94	**754.88**	12.4	**754.88**	5.7	**754.88**	4.2	**754.88**	3.4
CON8-6	693.1	96	**678.92**	12.4	**678.92**	3.4	**678.92**	4.1	**678.92**	7.9
CON8-7	814.8	94	**811.96**	13.0	**811.96**	2.5	**811.96**	4.0	**811.96**	3.0
CON8-8	774.0	94	**767.53**	12.5	**767.53**	3.2	**767.53**	3.4	**767.53**	3.2
CON8-9	809.3	92	**809.00**	12.9	**809.00**	3.8	**809.00**	3.5	**809.00**	2.4
Avg. deviation	0.32		0.01		0.00		0.00		0.00	
BKS found	22		38		40		40		40	
Avg. time (s.)		157.9		9.3		3.3		2.92		3.1
Scaled time (s)		52.1		8.4		3.5		*		3.1

Table 5
Comparison of h_PSO with RTS, LNS, PSO, ACS, AMM and PILS on the VRPSPD instances of Salhi and Nagy (1999).

Instance	RTS		LNS		PSO		ACS		AMM		PILS		h_PSO	
	Best Sol.	ACT	Best Sol.	ACT	Best Sol.	ACT	Best Sol.	ACT	Best Sol.	ACT	Best Sol.	ACT	Best Sol.	ACT
CMT1X	468.30	48	**467**	221	**467**	40	**466.77**	5.0	469.80	2.1	**466.77**	2.3	**466.77**	1.3
CMT1Y	–	–	**467**	235	**467**	40	**466.77**	5.0	469.80	3.8	**466.77**	2.3	**466.77**	1.4
CMT2X	**668.77**	94	704	294	710	54	684.21	20.8	684.21	5.4	684.21	6.4	684.21	35.6
CMT2Y	**663.25**	102	685	331	710	54	684.94	22.3	684.21	6.8	684.21	6.4	684.21	36.8
CMT3X	729.63	294	731	863	738	114	721.40	41.3	**721.27**	11.9	**721.27**	12.1	**721.27**	41.7
CMT12X	**644.70**	242	685	684	691	115	663.01	36.3	662.22	9.3	662.22	10.3	662.94	141.4
CMT3Y	745.46	285	738	708	740	113	721.40	43.8	**721.27**	11.0	**721.27**	12.3	**721.27**	55.5
CMT12Y	**659.52**	254	675	539	697	114	663.50	39.3	662.22	4.8	662.22	10.8	663.50	105.4
CMT11X	861.97	504	837	1821	895	226	839.66	57.3	**833.92**	21.2	**833.92**	18.9	**833.92**	244.9
CMT11Y	**830.39**	325	920	1376	900	228	840.19	52.8	833.92	14.4	833.92	19.0	833.92	368.9
CMT4X	876.50	558	879	1676	912	207	854.12	131.8	**852.46**	29.6	**852.46**	30.9	852.83	380.2
CMT4Y	870.44	405	876	1788	913	204	855.76	140.3	**852.46**	27.4	**852.46**	31.6	**852.46**	414.6
CMT5X	1044.51	483	1108	2340	1167	285	1034.87	377.5	1030.55	62.8	**1029.25**	71.5	1033.50	500.0
CMT5Y	1054.46	533	1146	2177	1142	286	1037.34	393.5	1030.55	47.7	**1029.25**	69.6	1036.00	500.0
Avg. deviation	1.31		4.07		6.09		0.92		0.76		0.65		0.71	
BKS found	5		2		2		2		5		9		6	
Avg. time (s.)		299.7		1075.2		148.6		97.6		18.4		21.7		201.9
Scaled time (s)		257.7		354.8		199.1		87.8		19.3		*		201.9

Table 6
Results of paired-t test.

Pairs (h_PSO–HM)	Mean Difference	p-value
h_PSO–RTS	−5.143	0.259
h_PSO–LNS	−29.428	0.008*
h_PSO–PSO	−45.928	0.001*
h_PSO–ACS	−1.500	0.030*
h_PSO–AMM	0.285	0.635
h_PSO–PILS	0.857	0.145

* Statistically significant different at $\alpha = 0.05$.

5.2.2. Computational results

We utilize following algorithms to evaluate the perfomance of the proposed h_PSO for the VRPMPD:

RTS: Reactive tabu search (Wassan et al., 2008).
LNS: Large neighborhood search (Ropke & Pisinger, 2006).
PSO: Particle swarm optimization (Ai & Kachitvichyanukul, 2009a).
ACS: Ant colony system (Gajpal & Abad, 2009).

It should be noted that we exclude the AMM and PILS from the comparison because there is no published results of them for the VRPMPD.

5.2.2.1. Results of h_PSO for Dethloff's (2001) data set. Wassan et al. (2008) are the first researchers who report the computational results of their algorithm, called RTS, on this data set. Thus, we compare the results of h_PSO with those obtained by RTS. Table 7 presents computational results of both algorithms. As seen from table, h_PSO improves 101 out of 120 best-known solutions (see, bold numbers indicating that the algorithm has reached the best solution), where the number of improvements in the first, second and third classes are 31, 32 and 38, respectively. The average improvement on the results of RTS is around 2%. For the 12 instances in which the results of h_PSO are inferior to those of RTS, the results are still within 0.01% of the best-known solutions. Meanwhile, h_PSO needs less computational burden to reach best solutions than RTS.

5.2.2.2. Results of h_PSO for Salhi and Nagy's (1999) data set. Table 8 reports the computational results of RTS, LNS, PSO, ACS and h_PSO over 42 instances. As mentioned previously, bold numbers for each algorithm in the table indicate that the algorithm has reached the best solution. It is important to note that average computation times (ACT) are not given for PSO in the table because Ai and Kachitvichyanukul (2009a) does not report them in their study. As seen from the table, h_PSO reaches 16 best solutions out of 42

Table 7
Comparison of h_PSO with RTS on the VRPMPD instances of Dethloff (2001).

Instance	RTS		h_PSO		Instance	RTS		h_PSO		Instance	RTS		h_PSO	
	Best sol.	ACT	Best sol.	ACT		Best sol.	ACT	Best sol.	ACT		Best sol.	ACT	Best sol.	ACT
SCA3-0T	627.92	3.0	**604.22**	0.6	SCA3-0Q	608.70	17.0	**591.84**	0.4	SCA3-0H	598.27	1.0	**567.20**	0.8
SCA3-1T	662.68	14.0	**657.99**	1.0	SCA3-1Q	678.30	1.0	**652.32**	0.8	SCA3-1H	624.86	1.0	**610.66**	1.0
SCA3-2T	636.57	3.0	**634.70**	1.8	SCA3-2Q	635.45	4.0	**622.10**	0.4	SCA3-2H	611.78	4.0	**584.52**	0.6
SCA3-3T	684.85	1.0	**651.86**	1.6	SCA3-3Q	678.36	9.0	**644.85**	1.1	SCA3-3H	650.03	3.0	**608.15**	2.8
SCA3-4T	666.18	12.0	**648.82**	0.5	SCA3-4Q	667.71	6.0	**642.04**	0.8	SCA3-4H	624.77	1.0	**597.35**	1.6
SCA3-5T	647.49	18.0	**619.31**	1.0	SCA3-5Q	618.01	10.0	**617.42**	0.9	SCA3-5H	602.57	10.0	**585.09**	1.3
SCA3-6T	634.83	1.0	**624.10**	1.5	SCA3-6Q	628.20	1.0	**601.28**	1.3	SCA3-6H	605.00	4.0	**574.25**	1.3
SCA3-7T	647.23	8.0	**631.41**	1.1	SCA3-7Q	623.78	7.0	**616.84**	0.7	SCA3-7H	579.64	12.0	**575.99**	0.7
SCA3-8T	700.52	6.0	**691.13**	0.6	SCA3-8Q	690.50	9.0	**676.85**	0.7	SCA3-8H	676.44	15.0	**635.84**	1.6
SCA3-9T	650.50	2.0	**642.86**	0.7	SCA3-9Q	630.10	25.0	**628.31**	0.8	SCA3-9H	645.69	13.0	**600.92**	1.7
SCA8-0T	911.61	1.0	**889.40**	3.0	SCA8-0Q	**804.92**	5.0	804.93	4.7	SCA8-0H	724.51	1.0	**714.86**	2.1
SCA8-1T	**947.00**	6.0	947.01	6.4	SCA8-1Q	924.52	6.0	**922.00**	3.5	SCA8-1H	787.66	7.0	**780.77**	1.5
SCA8-2T	**897.15**	1.0	897.16	1.7	SCA8-2Q	**830.43**	2.0	830.44	4.2	SCA8-2H	**739.93**	2.0	740.55	1.4
SCA8-3T	922.95	1.0	922.96	6.7	SCA8-3Q	852.47	1.0	**849.00**	1.8	SCA8-3H	764.52	7.0	**761.83**	6.3
SCA8-4T	991.49	4.0	**987.25**	2.4	SCA8-4Q	909.57	4.0	**906.29**	1.1	SCA8-4H	819.39	1.0	**794.69**	2.4
SCA8-5T	**957.44**	4.0	**957.44**	1.3	SCA8-5Q	**880.58**	4.0	**880.58**	1.4	SCA8-5H	788.42	4.0	**783.04**	4.0
SCA8-6T	**916.02**	5.0	916.03	8.6	SCA8-6Q	820.87	3.0	**804.21**	4.9	SCA8-6H	731.41	2.0	**723.21**	2.6
SCA8-7T	**931.48**	1.0	**931.48**	0.7	SCA8-7Q	857.97	1.0	**855.24**	5.7	SCA8-7H	735.98	7.0	**725.79**	1.1
SCA8-8T	1005.01	5.0	**1001.19**	5.5	SCA8-8Q	**927.09**	1.0	927.10	2.4	SCA8-8H	856.20	12.0	**850.91**	5.4
SCA8-9T	951.14	2.0	**938.75**	4.9	SCA8-9Q	845.21	5.0	**841.97**	5.9	SCA8-9H	803.18	1.0	**771.96**	3.2
CON3-0T	617.79	4.0	**601.85**	1.4	CON3-0Q	622.66	20.0	**595.54**	3.0	CON3-0H	604.65	8.0	**579.67**	2.4
CON3-1T	**538.66**	3.0	**538.66**	1.5	CON3-1Q	536.05	9.0	**529.05**	0.6	CON3-1H	537.88	13.0	**513.46**	1.0
CON3-2T	498.98	6.0	**498.15**	0.8	CON3-2Q	499.40	9.0	**493.09**	1.3	CON3-2H	502.23	6.0	**482.74**	1.9
CON3-3T	629.77	4.0	**569.07**	1.0	CON3-3Q	567.49	21.0	**557.18**	0.7	CON3-3H	**549.68**	9.0	**549.68**	1.0
CON3-4T	603.95	1.0	**572.52**	1.0	CON3-4Q	580.89	2.0	**571.57**	1.1	CON3-4H	568.28	4.0	**551.92**	3.2
CON3-5T	555.71	4.0	**549.49**	0.8	CON3-5Q	562.89	3.0	**545.90**	2.2	CON3-5H	534.22	9.0	**529.93**	1.5
CON3-6T	494.11	14.0	**489.54**	1.5	CON3-6Q	486.89	6.0	**482.69**	4.0	CON3-6H	480.00	11.0	**466.61**	4.7
CON3-7T	573.60	8.0	**553.95**	6.8	CON3-7Q	559.99	7.0	**541.61**	0.1	CON3-7H	551.45	15.0	**527.98**	0.5
CON3-8T	514.78	10.0	**509.16**	1.9	CON3-8Q	508.61	3.0	**492.14**	0.3	CON3-8H	500.77	9.0	**475.77**	0.9
CON3-9T	573.52	15.0	**570.38**	3.4	CON3-9Q	**563.88**	1.0	563.89	3.0	CON3-9H	559.57	6.0	**536.89**	1.0
CON8-0T	809.60	1.0	**802.83**	4.5	CON8-0Q	755.44	1.0	**754.50**	7.0	CON8-0H	716.41	7.0	**697.34**	6.3
CON8-1T	702.04	2.0	**682.83**	2.1	CON8-1Q	652.47	1.0	**648.30**	4.2	CON8-1H	611.86	7.0	**610.23**	4.4
CON8-2T	655.97	1.0	**653.29**	10.5	CON8-2Q	596.33	4.0	**595.64**	5.3	CON8-2H	601.73	5.0	**590.43**	4.2
CON8-3T	**740.61**	5.0	740.62	2.6	CON8-3Q	706.86	2.0	**685.99**	3.8	CON8-3H	663.75	6.0	**653.81**	6.9
CON8-4T	726.06	3.0	**720.59**	7.8	CON8-4Q	**705.37**	4.0	**705.37**	3.3	CON8-4H	659.69	5.0	**641.59**	3.9
CON8-5T	**717.21**	1.0	717.22	1.8	CON8-5Q	671.26	1.0	**666.17**	3.4	CON8-5H	623.46	1.0	**608.59**	2.3
CON8-6T	651.39	1.0	**644.75**	3.6	CON8-6Q	585.43	3.0	**584.72**	6.0	CON8-6H	557.52	1.0	**546.06**	3.6
CON8-7T	**755.97**	4.0	**755.97**	2.0	CON8-7Q	**724.57**	1.0	**724.57**	2.5	CON8-7H	682.53	1.0	**652.70**	5.2
CON8-8T	733.04	1.0	**714.17**	3.1	CON8-8Q	652.76	1.0	**648.54**	2.3	CON8-8H	596.18	6.0	**595.60**	3.0
CON8-9T	763.97	2.0	**760.96**	10.4	CON8-9Q	**719.58**	1.0	714.64	6.3	CON8-9H	631.23	1.0	**624.03**	3.0
Avg. dev.	1.59		0.00			1.44		0.00			2.84		0.00	
BKS found	6		31			5		32			1		38	
Avg. time		4.7		3.00			5.5		2.6			6.0		2.6
Scaled time		4.0		3.00			4.7		2.6			5.1		2.6

Table 8
Comparison of h_PSO with LNS, PSO, ACS on the VRPMPD instances of Salhi and Nagy (1999).

Instance	PSO	RTS		LNS		ACS		h_PSO	
	Best sol.	Best sol.	ACT	Best sol.	ACT	Best sol.	ACT	Best sol.	ACT
CMT1T	**520**	**520**	2.9	**520**	34	**520.06**	7.0	**520.06**	2.6
CMT2T	810	789	0.5	**783**	57	**782.77**	26.0	**782.77**	19.4
CMT3T	827	808	5.1	**798**	109	**798.07**	42.6	**798.07**	94.8
CMT12T	792	801	21.4	**788**	96	**787.52**	52.0	**787.52**	28.2
CMT11T	1026	1101	6.0	1000	164	**998.80**	70.2	**998.86**	66.5
CMT4T	1014	1009	51.8	1000	212	**990.39**	166.8	**990.39**	516.2
CMT5T	1297	1265	362.3	**1227**	333	1232.08	460.8	1233.52	500.0
CMT1Q	**490**	498	1.7	**490**	41	**489.74**	6.0	**489.74**	1.1
CMT2Q	739	739	0.6	733	65	732.76	26.2	**726.27**	23.5
CMT3Q	768	766	14.5	**747**	128	**747.15**	39.8	**747.15**	62.6
CMT12Q	733	744	23.7	**729**	108	729.46	42.0	**729.25**	200.6
CMT11Q	964	1038	6.6	**939**	196	**939.36**	66.2	**939.36**	92.1
CMT4Q	938	944	59.5	918	244	913.93	153.0	**913.63**	474.6
CMT5Q	1174	1176	8.5	**1119**	381	1134.72	451.8	1129.37	500.0
CMT1H	**464**	468	0.7	465	51	465.02	5.6	465.02	2.1
CMT2H	668	667	17.1	663	78	662.63	22.0	**661.39**	36.5
CMT3H	**701**	730	23.8	**701**	186	701.31	35.6	**700.94**	78.6
CMT12H	635	646	42.6	**629**	150	629.37	32.8	**629.02**	178.9
CMT11H	830	880	51.9	**818**	303	820.35	45.8	**818.05**	137.2
CMT4H	883	890	376.4	**829**	345	831.39	125.4	831.04	273.7
CMT5H	1044	1078	19.0	**983**	514	992.37	351.4	997.90	500.0
Avg. deviation	2.38	3.75		0.14		0.23		0.17	
BKS found	4	1		15		9		16	
Avg. time (s.)			52.2		180.7		106.1		180.4
Scaled time (s)			44.9		59.6		95.5		180.4

possible solutions. Meantime, h_PSO produces three new best solutions for the test instances referred CMT2Q, CTM4Q and CMT12Q. The second best algorithm in terms of the number of best solutions found is LNS which produces 15 best solutions. ACS, PSO and RTS reach 9, 4 and 1 best solution, respectively. Regarding the percentage deviation of solution of each algorithm from the best solution, we can see that h_PSO, LNS and PSO have less than 0.3% deviation while PSO and RTS produce 2.38% and 3.75% deviation. The comparison of computational burden of the algorithms shows that h_PSO needs slightly greater computation time than other algorithms but this is in the acceptable level.

To see whether differences in solution quality for the pairs of h_PSO–RTS, h_PSO–LNS, h_PSO–PSO and h_PSO–ACS are statistically significant, we implement the paired-*t* test. As in the VRPSPD, a one-sided alternative hypothes is constructed for the solution quality. At the significance level of 0.05, h_PSO performs better than RTS and PSO with *p*-values of <0.000, while there is no statistical significant difference between h_PSO and LNS, ACS. These comparisons also reveal that the proposed h_PSO is capable of finding good quality solutions for the VRPMPD and competes with existing algorithms in the literature.

6. Conclusion

In this paper, we have proposed a hybrid search algorithm based on discrete particle swarm optimization (PSO) and variable neighborhood descent algorithm (VND) to solve vehicle routing problem with simultaneous pickup and delivery (VRPSPD). This algorithm, called h_PSO, is a PSO supplemented with VND as a local search which is employed to improve randomly selected solutions in each iteration. Moreover, h_PSO implements an annealing-like strategy to preserve its swarm diversity. In h_PSO, we have proposed permutation encoding that is a giant tour without trip delimiters to represent a solution of the problem. To the best of our knowledge, this is the first implementation of giant tours to represent VRPSPD solutions. In order to demonstrate the effectiveness of the h_PSO, we have carried out an experimental study into two-stage. The first stage compares h_PSO with the existing algorithms (i.e. PSO, ACS, PILS, AMM, RTS and LNS) in the VRPSPD literature. The computational results over 54 test instances show that while h_PSO improves the results of the population-based heuristic approaches, i.e. PSO and ACS, around 2.49% and 0.05% on average, respectively, it is competitive with the effective heuristic approaches, i.e. PILS and AMM, such that the h_PSO and PILS produce less than 0.2% deviation on average from the best known solutions. Since the VRP with mixed pickup and delivery (VRPMPD) is a special case of the VRPSPD, the proposed h_PSO can be directly applied to solve the VRPMPD. In the second stage, we have evaluated the performance of h_PSO against the algorithms in the literature (i.e. PSO, ACS, RTS and LNS) for the VRPMPD. The computational results over 141 test instances indicate that the proposed h_PSO performs better than the existing algorithms in terms of solution quality and improves 104 best known solutions of the VRPMPD (the improvement is around 2% on average). In the further research, this kind of hybridization of PSO and VND can be used to solve multi-depot VRPSPD and multi-objective vehicle routing problems.

Acknowledgement

The authors are indebted to the anonymous referees for their helpful comments and suggestions. This research is supported by The Scientific and Technological Research Council of Turkey as Scientific Research Project (No: 108E069).

References

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

Ai, T. J., & Kachitvichyanukul, V. (2009b). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering, 56*(1), 380–387.

Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP, 15*, 1–31.

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

Catay, B. (2010). A new saving-based ant algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications, 37*(10), 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.

Chen, A., Yang, G., & Wu, Z. (2006). Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University Science A, 7*, 607–614.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & L. Sandi (Eds.), *Combinatorial optimization*. Chichester, UK: Wiley.

Cosco, D. O., Golden, B. L., & Wasil, E. A. (1988). Vehicle routing with backhauls: Models, algorithms and case studies. In *Vehicle routing: Method and studies* (pp. 127–147). Amsterdam: Elsevier.

Crispim, J., & Brandao, J. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society, 56*, 1296–1302.

Czogalla, J., & Fink, A. (2008). On the effectiveness of particle swarm optimization and variable neighborhood descent for the continuous flow-shop scheduling problem. *Studies in Computational Intelligence, 128*, 61–89.

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

Deif, I., & Bodin, L. (1984). Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling. In A. Kidder (Ed.), *Proceedings of the Babson conference on software uses in transportation and logistic management* (pp. 75–96). USA: Babson Park.

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*, 235–247.

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

Eberhart, R. C., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence*. San Francisco: Morgan Kaufmann Publisher.

Gajpal, Y., & Abad, P. (2009). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers and Operations Research, 36*(12), 3215–3223.

Gajpal, Y., & Abad, P. (2010). Saving-based algorithms for vehicle routing problem with simultaneous pickup and delivery. *Journal of the Operational Research Society, 61*(10), 1498–1509.

Gen, M., & Cheng, R. (1997). *Genetic algorithms & engineering design*. NewYork: Wiley.

Golden, B., Baker, E., Alfaro, J., Schaffer, J. (1985). The vehicle routing problem with backhauling: Two approaches. In R. Hammesfahr (Ed.), Proceedings of the XXI annual meeting of S.E. Times (pp. 90–92).

Hansen, P., Mladenovic, N., & Perez, J. A. M. (2010). Variable neighborhood search: Methods and applications. *Annals of Operations Research, 175*, 367–407.

Kang, Q., & He, H. (2011). A novel discrete particle swarm optimization algorithm for meta-task assignment in heterogeneous computing systems. *Microprocessors and Microsystems, 35*, 10–17.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks* (pp. 1942–1948). NJ: Piscataway.

Lee, C. Y., Lee, Z. J., Lin, S. W., & Ying, K. C. (2010). An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. *Applied Intelligent, 32*, 88–95.

Liao, C. J., Tseng, C. T., & Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research, 34*, 3099–3111.

Liu, S., Huang, W., & Ma, H. (2009). An effective genetic algorithm for fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review, 45*(3), 434–445.

Liu, B., Wang, L., & Jin, Y. H. (2007). An effective PSO-based memetiv algorithm for flow shop scheduling. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics, 37*(1), 18–37.

Marinakis, Y., & Marinaki, M. (2010). A hybrid genetic-particle swarm optimization algorithm for the vehicle routing problem. *Expert Systems with Application, 37*(2), 1446–1455.

Marinakis, Y., Marinaki, M., & Dounias, G. (2010). A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence, 23*(4), 463–472.

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

Montane, F. A. T., & Galvão, 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.

Mosheiov, G. (1998). Vehicle routing with pick-up and delivery: Tour-partitioning heuristics. *Computers & Industrial Engineering, 34*(3), 669.

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*(1), 126–141.

Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operation Research, 35*(9), 2807–2839.

Pang, W., Wang, K. P., Zhou, C. G., & Dong, L. J. (2004). Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In *Proceedings of the 4th international conference on computer and information technology* (pp. 796–800). IEEE CS Press.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft, 58*(1), 21–51.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*, 1985–2002.

Prins, C. (2009). Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence, 22*(6), 916–928.

Repoussis, P. P., Tarantilis, C. D., Braysy, O., & Ioannou, G. (2010). A hybrid evolution strategy for the open vehicle routing problem. *Computers & Operations Research, 37*, 443–455.

Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with bakhauls. *European Journal of Operational Research, 171*(3), 750–775.

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.

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

Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics, 8*, 541–564.

Tan, K. C., Lee, L. H., & Ou, K. (2001). Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. *Engineering Applications of Artificial Intelligence, 14*(6), 825–837.

Tang, F. A., & Galvao, R. D. (2006). Vehicle routing problems with simultaneous pick-up and delivery service. *Journal of the Operational Research Society of India, 39*, 19–33.

Tang, L., & Wang, X. (2010). An improved particle swarm optimization algorithm for the hybrid flowshop scheduling to minimize total weighted completion time in process industry. *IEEE Transactions on Control Systems Technology*. doi:10.1109/TCST.2009.2036718.

Toth, P., & Vigo, D. (Eds.). (2002). *The vehicle routing problem*. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications.

Tasgetiren, M. F., Liang, Y. C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research, 177*, 1930–1947.

Wassan, N. A., Nagy, G., & Ahmadi, S. (2008). A heuristic method for the vehicle routing problem with mixed deliveries and pickups. *Journal of Scheduling, 11*, 149–161.

Wassan, N. A., Wassan, A. H., & Nagy, G. (2007). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization, 15*, 368–386.

Zachariadis, E. E., & Kiranoudos, T. (2011). A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Systems with Applications, 38*, 2717–2726.

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*(2), 1070–1080.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudos, T. (2010). An adaptive methodlogy for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research, 202*(2), 401–411.