Expert Systems with Applications 42 (2015) 3551–3561

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups

Jian Li [a,*], Panos M. Pardalos [b], Hao Sun [c], Jun Pei [d], Yong Zhang [e]

[a] College of Engineering, Nanjing Agricultural University, P.O. Box 64, 40 Dianjiangtai Road, Pukou District, Nanjing 210031, China
[b] Department of Industrial and Systems Engineering, Center for Applied Optimization, University of Florida, 303 Weil Hall, Gainesville, FL 32611, USA
[c] School of Management Science and Engineering, Qingdao University, Qingdao 266071, China
[d] School of Management, Hefei University of Technology, Hefei 230009, China
[e] School of Transportation, Southeast University, Nanjing 210096, China

ARTICLE INFO

Article history:
Available online 10 December 2014

Keywords:
Vehicle routing problem
Simultaneous deliveries and pickups
Multi-depot
Iterated local search
Adaptive neighborhood selection

ABSTRACT

Although the multi-depot vehicle routing problem with simultaneous deliveries and pickups (MDVRPSDP) is often encountered in real-life scenarios of transportation logistics, it has received little attention so far. Particularly, no papers have ever used metaheuristics to solve it. In this paper a metaheuristic based on iterated local search is developed for MDVRPSDP. In order to strengthen the search, an adaptive neighborhood selection mechanism is embedded into the improvement steps and the perturbation steps of iterated local search, respectively. To diversify the search, new perturbation operators are proposed. Computational results indicate that the proposed approach outperforms the previous methods for MDVRPSDP. Moreover, when applied to VRPSDP benchmarks, the results are better than those obtained by large neighborhood search, particle swarm optimization, and ant colony optimization approach, respectively.

1. Introduction

The vehicle routing problem with simultaneous deliveries and pickups (VRPSDP) is one of the variants of the vehicle routing problem (VRP). In VRPSPD, customers may simultaneously receive and send goods. In addition, all delivered goods must originate from the depot and all pickup goods must be transported back to the same depot. Since it was introduced by Min (1989), VRPSDP has received increasing attention owing to its commercial importance and the high computational complexity. In practice, there are numerous applications. The grocery stores often have both delivery (e.g., fresh food or soft drink) and pickup (e.g., outdated items or empty bottles) demands (Chen & Wu, 2006). Furthermore, pro-environmental practices like recycling of empty packaging and other reusable materials or equipment lead to the necessity of reverse product flows (Dethloff, 2001; Montane & Galvão, 2006; Zachariadis et al., 2010). Additionally, express customers may have both delivery and pick-up demands. In theory, VRPSDP is a NP-Hard problem and the fluctuating carrying load of vehicles increases the difficulty in checking the feasibility (Zachariadis, Tarantilis, & Kiranoudis, 2009).

In this paper, we deal with the extension of VRPSDP called the multi-depot vehicle routing problem with simultaneous deliveries and pickups (MDVRPSDP). Due to the development of communication and information technology, and the increasing pressure of transportation cost, many enterprises select the joint distribution of multiple depots instead of traditional fixed zone service of single depot since the joint distribution of multiple depots can obtain more savings of cost. For example, as is shown in Figs. 1 and 2, customer C is far away from major customers of its zone but close to the major customers of depot B. Obviously, assigning customer C to depot B can save many travel distances. Meanwhile, the response time to customers will be reduced and service level correspondingly be enhanced. MDVRPSDP can be applied to the distribution of chain supermarkets, soft drink and food companies in large cities. For Example, to timely meet the demands of their customers under the heavy traffic conditions, some chain supermarkets usually construct or rent several supply warehouses in the skirts of large cities in China.

From the theoretical point of view, MDVRPSDP is an extension of VRPSDP, therefore, like VRPSDP, MDVRPSDP is also an NP-Hard

* Corresponding author.
E-mail addresses: telorance61@gmail.com (J. Li), pardalos@ise.ufl.edu (P.M. Pardalos), rivaldoking@gmail.com (H. Sun), feiyijun.ufl@gmail.com (J. Pei), zhang7678@126.com (Y. Zhang).

http://dx.doi.org/10.1016/j.eswa.2014.12.004
0957-4174/

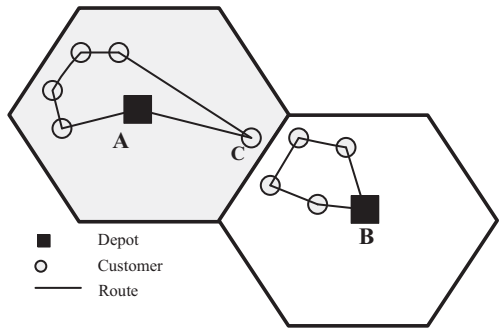

Fig. 1. Vehicle routing under separate distribution of each depot.

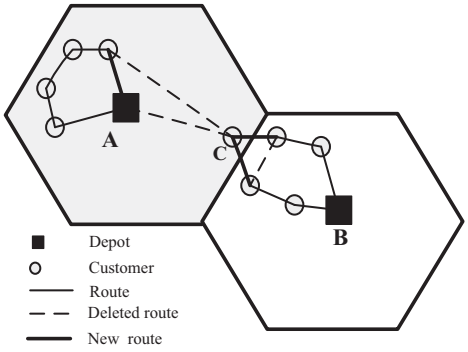

Fig. 2. Vehicle routing under joint distribution of multiple depots.

problem. Furthermore, the MDVRPSDP is more complicated than the VRPSDP considering that it needs to tackle customers' assignment and the VRPSDP problem simultaneously. Because the practical large-scale MDVRPSPD instances are difficult to be tackled efficiently by exact solution approaches, the purpose of the paper is to propose an effective metaheuristic for MDVRPSDP. To the best of our knowledge, this is the first metaheuristic developed for it. To make the implementation simpler, we employ the iterated local search (ILS) as the algorithm framework. Different structural neighborhood methods are used in the improving and perturbation steps of ILS to broaden the exploration of the search space. Meanwhile, an adaptive neighborhood selection mechanism (ANS) is incorporated into the framework of ILS, denoted by ILS_ANS, to manage the neighborhood methods effectively in improvement and perturbation steps of ILS, respectively. The main idea of ANS is that a neighborhood method is selected according to a probability depending on its success (Ropke & Pisinger, 2006a). In addition to integrating ANS into the framework of ILS, the second contribution of our work is the development of new perturbation neighborhood methods. The effectiveness of the algorithm is tested through 50 benchmarks instances for MDVRPSDP and its variant, VRPSDP.

The remainder of this paper is structured as follows: Section 2 is problem description and formulation. Section 3 is a review of the related literature. The overall structure and the details of the proposed approach are shown in Section 4. Computational results are provided in Section 5. Section 6 is conclusions and some suggestions for future work.

2. Problem description and formulation

MDVRPSDP is defined as follows: Let $G = (V, E)$ be a graph where V is the vertex set and E is the edge set. The vertex set V is partitioned into two subsets $V_c = \{v_1, \ldots, v_n\}$ and $V_d = \{v_{n+1}, \ldots, v_{n+p}\}$, which represent the set of customers and the set of depots, respectively. Among other things, n is the number of customers and p is the number of depots. Each vertex $v_i \in V_c$ has several nonnegative weights associated to it, namely, a non-negative pickup demand p_i and delivery demand d_i and a service time s_i. Furthermore, in the depot vertex $v_i \in V_d$, there are no demands and service times, i.e. $p_i = d_i = s_i = 0$. Associated with E are a distance matrix (d_{ij}) and a travel time matrix (t_{ij}), and $d_{ij} = t_{ij}$ for all $i, j \in V$. The distance matrix is symmetric and satisfies the triangle inequality, that is, $d_{ij} = d_{ji}$, and $t_{ij} = t_{ji}$. A fleet of m_d identical vehicles of capacity Q is available at each depot $v_{n+d} \in V_d$. The sum of vehicles of all the depots is m. In MDVRPSDP, the following constraints must be met:

(1) Each route starts and ends at the same depot.
(2) Each customer is only visited once by a vehicle or a route.
(3) The maximum load of each route does not exceed the vehicle capacity at each point of the route.
(4) The total duration of each route (including travel and service time) does not exceed a preset limit.
(5) All the vehicles are homogeneous.

2.1. Notations

Sets:

V_d: the depot set.
V_c: the customer set.
V: the vertex set, $V = V_c \cup V_d$.
K: the vehicle set.

Parameters:

d_i: the delivery demand of customer i.
p_i: the pick-up demand of customer i.
s_i: the service time of customer i.
d_{ij}: the distance between customer i and j.
Q_k: the capacity of vehicle k.
T_k: the maximum duration of vehicle k.
m_d : the number of vehicles at depot d.

Decision variables:

x_{kij} is the 0–1 decision variable. If vehicle k travels directly from node i to node j, then $x_{kij} = 1$, otherwise, $x_{kij} = 0$.
t_{kij} denotes the load on arc (i,j) of route k.

2.2. Mixed-integer linear programming formulation

The objective of MDVRPSDP is to determine the optimal routes by minimizing the weighted sum of the fixed cost related to the number of vehicles and the total travel cost of all the vehicles, where α and β are coefficients. The formulation for MDVRPSDP is given as follows:

$$\min \ \alpha \sum_{k\in K}\sum_{d\in V_d}\sum_{j\in V_c} x_{kdj} + \beta \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} d_{ij}x_{kij} \tag{1}$$

subject to

$$\sum_{k\in K}\sum_{j\in V_c} x_{kdj} \leqslant m_d, \quad \forall d \in V_d \tag{2}$$

$$\sum_{k\in K}\sum_{j\in V} x_{kij} = \sum_{k\in K}\sum_{j\in V} x_{kji} = 1, \quad \forall i \in V_c \tag{3}$$

$$\sum_{j\in V_c} x_{kdj} = \sum_{i\in V_c} x_{kid} \leqslant 1, \quad \forall k \in K,\ d \in V_d \tag{4}$$

$$\sum_{i\in R}\sum_{j\in R}x_{kij} \leqslant |R|-1, \quad \forall R \subseteq V_c,\ k \in K \tag{5}$$

$$\sum_{i\in V_d}\sum_{j\in V_d}x_{kij} = 0, \quad \forall k \in K \tag{6}$$

$$\sum_{k\in K}\sum_{j\in V}t_{kij} - \sum_{k\in K}\sum_{j\in V}t_{kji} = d_i - p_i, \quad \forall i \in V_c \tag{7}$$

$$t_{ijk} \leqslant Q_k, \quad \forall k \in K \tag{8}$$

$$\sum_{i\in V}\sum_{j\in V}(c_{ij}+s_i)x_{ijk} \leqslant T_k, \quad \forall k \in K \tag{9}$$

$$x_{kij} \in \{0,1\}, \quad \forall i,\ j \in V,\ k \in K \tag{10}$$

$$t_{ijk} \geqslant 0, \quad \forall i,j \in V,\ k \in K \tag{11}$$

Constraints (2) require that the number of vehicles departing from each depot is not more than the number of available vehicles. Constraints (3) ensure that each customer must be visited exactly once by exactly one vehicle. Constraints (4) represent that each vehicle starts from a depot, and ends at the same depot. Constraints (5) are the subtour elimination constraints that ensure the solution is connected. Constraints (6) impose that the vehicle cannot travel directly from depot i to depot j. Constraints (7) are flow conservation equations. The next two inequalities (8) and (9) represent capacity, and maximum duration of vehicles constraints, respectively. Constraints (10) and (11) denote the range of decision variables.

3. Related literature review

MDVRPSDP has received very limited attention, although it is often encountered in real-life scenarios of transportation logistics. For MDVRPSDP can be viewed as an extension of the VRPSDP, we briefly discuss the methodological contributions that have been reported for MDVRPSDP and VRPSDP in the following.

3.1. Multi-depot vehicle routing problem with simultaneous deliveries and pickups

To the best of our knowledge, there are no papers that use metaheuristics to solve MDVRPSDP and only two papers that attempted to treat MDVRPSDP by heuristics. The first one is that of Salhi and Nagy (1999) who suggested an extension to the classical insertion-based heuristic for MDVRPSDP. It permitted inserting more than one backhaul a time. The second one is that of Nagy and Salhi (2005) which proposed a method that firstly found a solution to the corresponding VRP problem and then modified the solution to make it feasible for VRPSDP. They both adopted the idea of borderline customers, that is, customers were divided into two subsets, namely borderline and non-borderline customers. The non-borderline customers were assigned to their nearest depots, then the borderline customers were inserted into the single depot vehicle routing one at a time.

3.2. Vehicle routing problem with simultaneous deliveries and pickups

With the development of reverse logistics, VRPSDP has received more and more attention. Exact algorithms only solve small size instances, for example, Dell'Amico et al. (2006) only found the optimum solution for instances up to 40 customers by a proposed exact algorithm based on branch-and-price approach. Therefore, many researches focus on using heuristics or metaheuristics to tackle this problem.

Min (1989) is the first to introduce VRPSDP, tackling an instance of 22 customers and 2 vehicles. He proposed a three-phase method which contained clustering customer nodes, assigning vehicles to clusters, and creating the route of each vehicle. Afterwards, several heuristics were presented for VRPSDP by Salhi and Nagy (1999), Dethloff (2001), Nagy and Salhi (2005) and Gajpal and Abad (2010). Salhi and Nagy (1999) and Dethloff (2001) proposed different insertion-based heuristics according to different criterions. Nagy and Salhi (2005) developed a composite heuristic approach which combines different routines. These routines were modified versions of VRP routines such as 2-opt, 3-opt, shift, exchange, perturb but also some specially developed for VRPSPD, such as Reverse and Neck. Gajpal and Abad (2010) presented the saving heuristic and the parallel saving heuristic for VRPSPD. They used a cumulative net-pickup approach for checking the feasibility when two existing routes were merged.

Recently, many metaheuristics have been successfully applied to VRPSDP. Crispim and Brandão (2005) proposed a hybrid metaheuristic which comprised of tabu search and variable neighborhood descent. Infeasible solutions are allowed by penalizing them according to the level of overload. Chen and Wu (2006) designed an insertion-based procedure for the initial solution, and a metaheuristic based on the record-to-record travel, tabu lists, and route improvement procedures for the improvement of the initial solution. Tang and Galvão (2006) presented a tabu search algorithm for VRPSDP, which used three types of inter-route movements and one type of intra-route improvement. Ropke and Pisinger (2006b) developed a large neighborhood search (LNS) heuristic to solve several variants of VRP including VRPSPD. Bianchessi and Righini (2007) presented 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. Zachariadis et al. (2009) developed a TS-based algorithm which explored the solution space by hybridizing the TS and guided local search (GLS) strategies. Ai and Kachitvichyanukul (2009) proposed a particle swarm optimization, whereas Gajpal and Abad (2009) presented an ant colony optimization approach for VRPSDP. Subramanian, Drummond, Bentes, Ochi, and Farias (2010) presented a parallel algorithm embedded with a multi-start heuristic consisting of a variable neighborhood descent procedure integrated in an iterated local search (ILS) framework. Zachariadis et al. (2010) proposed an adaptive memory (AM) algorithmic framework which collected and combined promising solution features to generate high-quality solutions. Çatay (2010) proposed an ant colony algorithm, employing a new saving-based visibility function and a pheromone updating procedure.

The most recent articles for VRPSPD are from Zachariadis and Kiranoudos (2011), Tasan and Gen (2012), Goksal, Karaoglan, and Altiparmak (2013) and Subramanian, Uchoa, and Ochi (2013). Zachariadis and Kiranoudos (2011) proposed a local search approach which efficiently explored rich solution neighborhoods by statically encoding tentative moves into special data structures. Tasan and Gen (2012) developed a genetic algorithm for VRPSDP. Goksal et al. (2013) presented a heuristic solution approach based on particle swarm optimization (PSO) in which a local search was performed by a variable neighborhood descent algorithm (VND). Moreover, it implemented an annealing-like strategy to preserve the swarm diversity. Subramanian et al. (2013) proposed a hybrid algorithm for a class of vehicle routing problems with homogeneous fleet, including VRPSDP. The hybrid algorithm hybridized an iterated local search based heuristic approach and a set partitioning formulation, called ILS-RVND-SP. The ILS-RVND heuristic used insertion heuristics in the constructive phase, a variable neighborhood descent with random neighborhood ordering (RVND) in the local search phase and simple moves in perturbation phase. Every time a local search was performed, the routes associated to the local

optimal solution may be added to a pool of routes. Then, to extract the best combination of routes, a MIP solver was employed to solve the set partitioning (SP) problem.

4. The proposed solution method

Iterated local search (ILS) was introduced by Lourenço, Martin, and Stützle (2001). It can help the local optimizer escape from being trapped in a local minimum while keeping many good properties of the local minimum. As it is simple and effective, it has been successfully applied to vehicle routing problem by Cordeau and Maischberger (2012) recently. In this paper we also use the iterated local search as the algorithm framework.

The basic ILS consists of the improvement and perturbation steps. To enhance the solution quality, a number of neighborhood methods are generally used in the improvement and perturbation steps. As different neighborhood methods obtain different quality solution and vary with different instances, we need to decide which neighborhood method to be applied for the next step of the search. Instead of traditional fixed order search of neighborhood methods and stochastic neighborhood selection (assigning equal probabilities to each neighborhood method), we embed the adaptive neighborhood selection mechanism within the ILS framework (ILS_ANS).

ILS_ANS is constructed in algorithm 1. It starts from an initial solution s_0, iteratively, and then the algorithm applies an improvement neighborhood method to this solution to obtain an improved solution s_w. If s_w satisfies an acceptance criterion, it replaces current solution s_c Otherwise, the search returns to the previous solution s_c. If s_c cannot be improved after the consecutive preset iterations *iter*2, the best solution s_b found during the search replaces current solution s_c. Then a perturbation neighborhood method is selected to apply to this solution s_c and obtains a new perturbation solution s_p. The next improvement is applied to this solution s_p, where $f(s)$ denotes the cost of solution s. Note that in the step of perturbation s_p is required to a new solution which has not been accepted since the procedure begins.

In order to identify whether a solution is new, we implement long term memory structures to record all the accepted solutions. Generally speaking, it is difficult to keep track of all previous solutions directly and also it is very time consuming to check all the solutions. Hence we explore a simple and fast tracing method based on the objective function. The method for recording solution is as follows in detail: firstly, we transform the travel distance objective value $f'(s)$ of solution s into an integer F by magnifying it. Next, a one-dimensional array Ar is constructed to record the times that the solution is accepted. Thus we update the record of the solution, namely $Ar(F) = Ar(F) + 1$. An example is given to identify whether a solution s with the travel distance value 816.1 is a new solution: we calculate F, where $F = f(s) * 10 = 8161$. If $Ar(F)$ is less than 1, then s is a new solution, otherwise, it is not a new one.

Algorithm 1. The proposed algorithm (ILS_ANS)

Generating the initial solution s_0, $s_b \leftarrow s_c \leftarrow s_p \leftarrow s_0$
Do While the termination condition (specified number *iter*1 of iterations) is not true
  select an improvement neighborhood by adaptive selection mechanism
  $s_w \leftarrow$ improve (s_p) by applying the selected improvement neighborhood
  If s_w satisfies the acceptance criterion then
    $s_c \leftarrow s_w$
  End If
  If $f(s_c) < f(s_b)$ then
    $s_b \leftarrow s_c$
  End If
  $s_p \leftarrow s_c$
  If s_c has not been improved after the consecutive preset iterations *iter*2 then
    $s_c \leftarrow s_b$
    select a perturbation neighborhood by adaptive selection mechanism
    $s_p \leftarrow$ perturb (s_c) by applying the selected perturbation neighborhood
  End If
Loop
Return s_b

Compared with the ILS of Cordeau and Maischberger (2012), our proposed algorithm is different in three ways: (1) when there are more neighborhood methods, it is more difficult to explore and

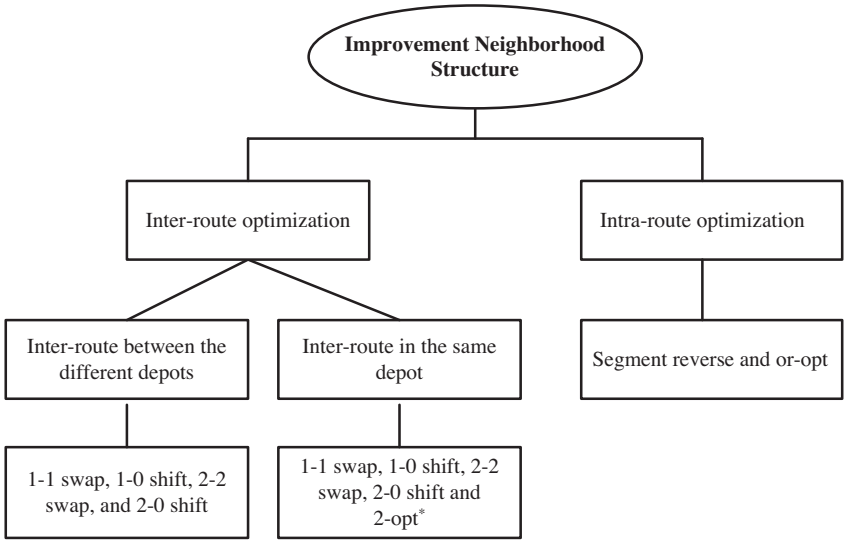

Fig. 3. Improvement neighborhood structure.

check the tabu list for tabu search algorithm. Therefore, instead of tabu search algorithm, adaptive neighborhood selection is applied to improve the current solution in the improvement step; (2) multiple perturbation neighborhoods with adaptive selection mechanism are applied in the perturbation step to strengthen the diversity, as opposed to single perturbation neighborhood; (3) the probability that the worse solution is accepted is related to the times that the solution is accepted, which is described in Section 4.6 in detail, instead of current iteration number.

In the following sections, we describe the construction of an initial solution, the improvement neighborhood structure, the perturbation neighborhood structure, the adaptive neighborhood selection mechanism, the objective function and diversification mechanism, and the acceptance and termination criteria. We start by describing the construction of an initial solution. Note that all constraints are satisfied except for constraints (2) in the initial solution, improvement solutions and perturbation solutions.

4.1. Construction of an initial solution

We modify the saving algorithm proposed by Clark and Wright (1964) to get an initial solution for MDVRPSDP. This method is described as algorithm 2.

Algorithm 2. Initial solution method

Step 1: Assign the customers to their nearest depot
Step 2: Apply the saving algorithm to each depot v_{n+k}
Step 2.1: Create a separate route for each customer
Step 2.2: Calculate the savings $\Delta_{ij} = d_{v_{n+k},i} + d_{v_{n+k},j} - d_{i,j}$ for all pairs of customers i and j
Step 2.3: Order the savings in a non-increasing way and form the saving list
Step 2.4: Start at the arc (i,j) with the maximum Δ_{ij} in the saving list, if the combined route satisfies the constraint conditions, then they can be combined into a new feasible route with the highest savings. The saving of the arc (i,j) is deleted from the saving list
Step 2.5: Try the next combination in the list and repeat step 2.4 until no more combinations are feasible
Step 3: Merge all the routes into an initial solution

4.2. Improvement neighborhood structure

In MDVRPSDP, there are three types of optimizations which are inter-route among the different depots, inter-route in the same depot, and intra-route optimization. The improvement neighborhood structure is shown in Fig. 3. In the improvement step, standard neighborhoods are selected to improve the current solution: 1–1 swap, 1–0 shift, 2–2 swap, and 2–0 shift are selected to apply to inter-route improvement among different depots; 1–1 swap, 1–0 shift, 2–2 swap, 2–0 shift and 2-opt* are selected for inter-route improvement within the same depot; segment reverse and or-opt are selected for intra-route improvement.

To improve the solution efficiently, each of inter-route neighborhood methods is bonded with each of intra-route neighborhood methods, namely, every bond is regarded as a composite neighborhood operator. All the combinations between inter-route neighborhoods methods and intra-route neighborhood methods compose the new neighborhood set, that is, the new neighborhood set is composed of 1–1, segment reverse; 1–1, or-opt; 1–0, segment reverse; 1–0, or-opt; 2–2, segment reverse; 2–2, or-opt; 2–0, segment reverse; 2–0, or-opt; 2-opt*, segment reverse; 2-opt*, or-opt. The generation of the composite neighborhood operators in the improvement step of ILS_ANS is shown in Fig. 4.

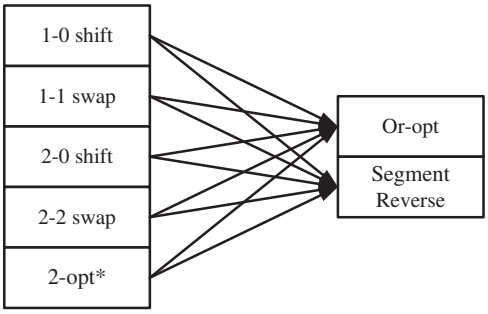

Fig. 4. Composite neighborhood operators in the improvement step of ILS_ANS.

Swap/shift move was introduced by Chen and Wu (2006). It is based on customer interchange between vehicle routes. Operators 1–0 and 2–0 result in a shift of one or two customers from one route to another; operators 1–1 and 2–2 result in an exchange that the sequence of one or two consecutive customers on one route is exchanged with the sequence of one or two customers on another route. The computational complexity of these moves is $O(n^3)$.

2-opt* was introduced by Potvin, Kervahut, Garcia, and Rousseau (1992). In this procedure, two links are removed from two different routes and two new links are introduced by connecting the first customers on the first route to the last customers on the second route and connecting the first customers on the second route to the last customers on the first route. The computational complexity of 2-opt* is $O(n^3)$.

Segment Reverse is an extension of the reverse operator introduced by Nagy and Salhi (2005). It reverses a segment of a route whereas the reverse operator is to reverse the whole route. The computational complexity of Segment Reverse is $O(n^2)$.

Or-opt was introduced by Or (1976). In this procedure, a sequence of three consecutive customers, two consecutive customers, or a single customer on a route is removed and inserted to another location in the same route. The computational complexity of or-opt is $O(n^2)$.

4.3. Perturbation neighborhood structure

The perturbation neighborhood method in our ILS_ANS is used to generate the new solution to diversify the search. To explore the broad space of the solution, different structural neighborhoods (large neighborhood search, LNS) are employed to perturb the local optima. LNS is composed of two parts: the removal of customers and the reinsertion of the removed customers. To generate more new solutions, we improve the traditional LNS. The main differences between our proposed LNS and traditional LNS lie in the reinsertion of the removal customers. Two new insertion methods are proposed: (1) insertion method based on the tournament; (2) probability assignment-first, insertion-second method.

The difference between the insertion method based on the tournament and the basic insertion method lies in the different number of their selected objects for comparison. In order to introduce it conveniently, we compare the greedy insertion based on the tournament (simply denoted by greedy_T) with the basic greedy insertion (simply denoted by greedy_B). The pseudo-code of the basic greedy insertion is shown in algorithm 3 introduced by Campbell and Savelsbergh (2004), while the pseudo-code of greedy insertion based on the tournament is given in algorithm 4. The customers of M' are randomly selected from the set M, namely, $M' \subseteq M$. When the customers of M' include all the customers of M, namely,

$M' = M$, the greedy insertion heuristic based on tournament is transformed into the basic greedy insertion heuristic. Hence, the greedy insertion heuristic based on tournament extends the basic greedy insertion heuristic, as the basic greedy insertion heuristic can be viewed as a special case of greedy insertion heuristic based on tournament. The insertion method based on the tournament supplies the more combinations, therefore, it can produce more new solutions than the basic insertion method. Additionally, when only one customer is in the set M', namely $|M'| = 1$, the greedy insertion heuristic based on tournament (simply denoted by greedy_T_1) results in a time complexity of $O(n^3)$ under the situation that checking the feasibility of an insertion can be done in $O(n)$, whereas the basic greedy insertion heuristic results in a time complexity of $O(n^4)$ under the same checking of the feasibility.

In a similar way, we construct the regret insertion method based on the tournament (simply denoted by regret_T). The regret-2 insertion method introduced by Ropke and Pisinger (2006a) is employed. The basic greedy insertion method selects an unrouted customer with least insertion cost to insert at each iteration, whereas the regret-2 insertion method selects an unrouted customer with largest regret value. The regret value is the difference in the cost of inserting the request in its best position and its second-best position. When only one customer is in the set M', the insertion method regret_T is the same as the greedy_T_1.

Algorithm 3. Basic greedy insertion heuristic

```
M = set of removal customers
R = set of routes
While M ≠ ∅ do
  p^0 ← +∞
  For j ∈ M do
    For r ∈ R do
      For (i − 1, i) ∈ r do
        If Feasible(i,j) and Cost(i,j) < p^0 then
          r^0 ← r
          i^0 ← i
          j^0 ← j
          p^0 ← Cost(i,j)
        End If
      End For
    End For
  End For
  Insert(i^0, j^0)
  M = M\j^0
  Update(r^0)
End While
```

The probability assignment-first, insertion-second method is described in the followed way: a customer is firstly assigned to the depot according to the probability related to the distances between the customer and each depot. The probability that customer i is assigned to depot k is p_{ik}, where $p_{ik} = \lambda_{ik}/\sum_{j\in V_d}\lambda_{ij}\ \forall i \in V_c,\ k \in V_d$, and $\lambda_{ij} = 1/d_{ij}^2\ \forall i \in V_c,\ j \in V_d$. A roulette wheel selection rule is employed to decide which depot a customer is assigned to. Then, the customers assigned are inserted in the corresponding depots by the basic insertion methods, such as the greedy insertion and the regret insertion. The process is repeated until all the unrouted customers are arranged to the routes. The probability assignment-first, basic greedy insertion-second method is simply denoted by A_greedy_B. Likewise, the probability assignment-first, regret-2 insertion-second method is simply denoted by A_regret_2.

Algorithm 4. Greedy insertion heuristic based on tournament

```
M = set of removal customers
R = set of routes
While M ≠ ∅ do
  p^0 ← +∞
  M' = subset of M
  For j ∈ M' do
    For r ∈ R do
      For (i − 1, i) ∈ r do
        If Feasible(i,j) and Cost(i,j) < p^0 then
          r^0 ← r
          i^0 ← i
          j^0 ← j
          p^0 ← Cost(i,j)
        End If
      End For
    End For
  End For
  Insert(i^0, j^0)
  M = M\j^0
  Update(r^0)
End While
```

The customers are removed by three methods which are random removal introduced by Shaw (1998), relatedness removal introduced by Schrimpf, Schneider, Stamm-Wilbrandt, and Dueck (2000) and long-arc-broken removal, described as follows in detail.

In random removal, for each customer of the current solution, if a number r generated randomly is less than threshold level r_0, and $r, r_0 \in (0,1)$, then the customer is removed. For relatedness removal, a customer i is firstly selected randomly, then customer j is selected if the distance d_{ij} between customer i and customer j is in the interval $(0, r * Average(i))$, where $Average(i)$ is the average distance value of all the arcs which are adjacent to vertex i in E, and r is random number in the interval $(0,1)$.

Next, the long-arc-broken removal heuristic is designed. Firstly, we compute the sum of longest arc and second-longest arc of each route. At the second step we rank these routes by the sum of longest arc and second-longest arc of each route in a non-increasing way. At the third step we select stochastically a route from first φ routes, and then remove the customers between the longest arc and the second-longest arc. Fig. 5 illustrates when the long-arc-broken removal heuristic can be useful.

As the different combinations of the removal of customers and the insertion of customers exist, many perturbation neighborhood methods are generated. Ten perturbation neighborhood methods are selected to constitute the perturbation neighborhood set. They are random-greedy_T_1, relatedness-greedy_T_1, random-greedy_T, relatedness-greedy_T, random-regret_T, relatedness-regret_T, relatedness- A_greedy_B, relatedness-A_regret_2, long-arc-broken-greedy_T, and long-arc-broken – regret_T, respectively. The generation of the perturbation neighborhood operators is shown as Fig. 6.

4.4. An adaptive neighborhood selection mechanism

As the performance of different neighborhood methods may vary significantly among different problems and instances (Chakhlevitch & Cowling, 2008), we embed an adaptive mechanism for the choice of improvement neighbor methods and perturbation neighborhood methods. The adaptive mechanism employs a scoring rule. According to the score of each neighborhood method, the roulette wheel selection is used to decide which neighborhood

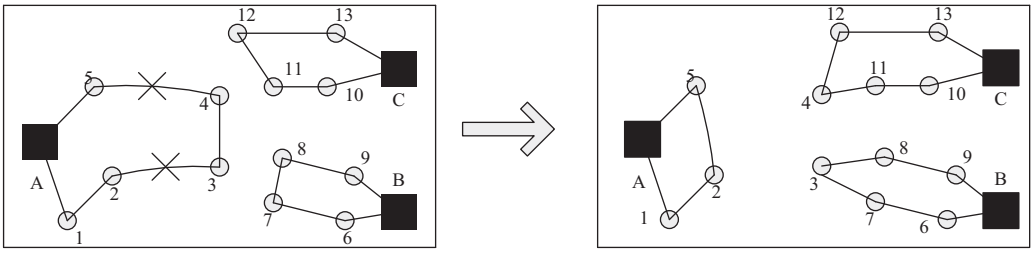

Fig. 5. Long-arc-broken removal.

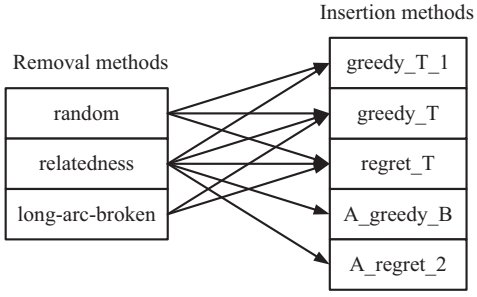

Fig. 6. The generation of perturbation neighborhood operators.

method is applied, which has ever been successfully applied to simulated annealing by Ropke and Pisinger (2006a) for pickup and delivery problem.

In ILS_ANS, the adaptive neighborhood selection mechanism is divided into two parts: the adaptive selection of improvement and the perturbation neighborhood methods. The adaptive selection of improvement neighborhood methods is as follows.

Each improvement neighborhood method is selected by the roulette wheel selection procedure with a probability that is proportional to its known empirical quality. If there are k improvement neighborhood methods with weights w_i, $i \in \{1,2,\ldots,k\}$, then the improvement neighborhood method j is selected with probability $\frac{w_j}{\sum_{k=1}^{k} w_i}$.

At the beginning of the search, the equal weights are assigned to all improvement neighborhood methods. During the search, the weights of the improvement neighborhood methods are updated every η iterations depending on the success of each method. The profitability of each method is calculated by a scoring system. The weight of each improvement neighborhood method is computed and adjusted automatically by three steps:

Step1: dividing the search process into many segments, and each segment consists of η iterations.
Step2: computing the scores of each improvement neighborhood method after segment j is completed. A score of θ_1 is added to an improvement neighborhood method whenever a new overall best solution is found after applying the method in improving step of ILS and a score of θ_2 if the current solution is improved and the solution is a new solution. Otherwise, a score of θ_3 is assigned if the solution is accepted.
Step3: updating the weight of each improvement neighborhood method in segment $j+1$ by the following formula:

$$w_{i,j+1} = (1-\lambda)w_{i,j} + \lambda\frac{\mu_i}{\phi_i}$$

where $w_{i,j}$ is the weight of method i used in segment j, μ_i is the score of the method i obtained during the last segment, ϕ_i is the number of times of method i used during the last segment, and $\lambda \in (0,1)$ is a system parameter that allows for controlling the adaptive behavior of the algorithm to the recent trend.

In the perturbation step, the adaptive selection of perturbation neighborhood is the same as the one of the improvement neighborhood methods except for the value of λ', η' and score parameters. The score of θ_4 is rewarded if a new overall best solution is obtained, otherwise, a score of θ_5 is only assigned after applying a perturbation neighborhood method. If a new overall best solution is obtained after applying it, the perturbation neighborhood method is directly selected as the next perturbation neighborhood method, otherwise, the roulette wheel selection procedure is employed to select a new perturbation neighborhood method for next perturbation.

4.5. Objective function and diversification mechanism

To strengthen the search, the objective function (1) described in Section 2 is augmented with two weighted penalty terms. One is the assessment value related to distances, and the other is related to the maximum load of each route. It is assumed that arcs $(i-1,i)$ and $(i,i+1)$ belong to the current solution. The new objective function is reformulated as (12),

$$Z' = Z + \gamma d' + \delta w' \tag{12}$$

where $d' = \sum_{i\in V_c}\left(d_{i,i-1} + d_{i,i+1} - 2*\overline{d_l}\right)*\overline{d_l}$, $\overline{d_l} = (d_{i,i^*} + d_{i,i'})/2$, customer i^* is the nearest to customer i, while customer i' is the second-nearest to customer i. $w' = \sum_{r\in R}(maxload(r)/\max(load\_p(r), load\_d(r)))$, where R is the route set, $maxload(r)$ is the maximum load of route r, $load\_p(r)$ is the total delivery of route r , and $load\_d(r)$ is the total pickup of route r.

4.6. Acceptance and stopping criterion

Whenever a new overall best solution is found in the improvement step, it replaces the current solution. Besides, when a solution is only better than the current solution, it is still accepted if the solution has not been accepted before. Otherwise, the solution is accepted with probability $1/\sqrt[\rho]{f}$, where f is the times that the solution is accepted, and ρ is a parameter. The probability rule can broaden the search space by changing the search trajectory, on the other hand, it can avoid losing all the other solutions with the same objective value.

When the preset consecutive iteration number $iter2$ is reached whereas the current solution s_c has not been improved, the perturbation operator will be applied again. The algorithm stops when a specified number $iter1$ of iterations are reached.

5. Computational results

The algorithm is coded in visual basic 6.0, and run on a laptop computer with an Intel Core i7 2.9 GHz processor with 8 GB RAM and Windows® 7 Professional edition. ILS_ANS is run 10 times on

each instance. All the computational times are indicated by seconds.

In the following section, we first introduce the data sets in Section 5.1. The parameter values are given in Section 5.2. In Section 5.3 we compare the adaptive neighborhood selection with the stochastic neighborhood selection for MDVRPSDP. In Section 5.4 we compare ILS_ANS with other heuristics for MDVRPSDP. In Section 5.5 ILS_ANS is applied to VRPSDP.

5.1. Data sets

In order to assess the effectiveness of ILS_ANS, it is tested on MDVRPSDP and its special case VRPSDP. We adopt the data set generated by Salhi and Nagy (1999) as the tested instances. In total, the computational instances include 22 multi-depot problem instances (2–5 depots, 50–249 customers) and 28 single-depot problem instances (50–199 customers). The multi-depot instances are derived from Gillett and Johnson (1976). Each of the multi-depot instances is partitioned as *X* and *Y* types on the basis of the difference of deliveries and pickups. The single-depot instances are derived from Christofides, Mingozzi, and Toth (1979), and each of them is also sorted as *X*, and *Y* types, respectively. The basic characteristics of the multi-depot and single-depot instances are shown in Tables 1 and 2, respectively. Items without routing duration constraints are indicated by dashes.

5.2. Parameter setting

The proposed solution approach contains four kinds of parameters: (1) the parameters of LNS, including random removal probability r_0 and the number of routes with longer arcs φ; (2) the parameters of adaptive neighborhood selection mechanism, which include the scoring parameters θ_1, θ_2, θ_3, θ_4, θ_5, reaction factors λ, λ', and the number of iterations updating the weight of the neighborhood method η, η', at improvement steps and perturbation steps, respectively; (3) the parameters of objective function, containing distance penalization coefficient γ and overload penalization coefficient δ; (4) the probability acceptance parameter ρ, the number of stopping iteration *iter*1, and the number of threshold iteration of the perturbation *iter*2.

The main parameters of the ILS_ANS approach are scoring parameter θ and the probability acceptance parameter ρ. As for the scoring parameter θ, the scores of a neighborhood method represent its effectiveness at the current stage. The more its scores are, the higher its selected probability at next stage is, when the reaction factor is given. The probability acceptance parameter ρ controls the search trajectory. The smaller it is, the lower the acceptance probability of the repetition solution is, and vice versa. Meanwhile, the lower acceptance probability of the repetition solution would result in the weakness of local search. However, the higher acceptance probability of the repetition solution would have the search trapped in a local optima. Therefore, it is important for the probability acceptance parameter ρ to obtain a proper value.

To obtain the values of these parameters, the medium size MDVRPSDP instances GJ3X and GJ3Y are employed to test the parameters by changing the value of one parameter while keeping the other parameters fixed. The tested results show that in the improvement step, a score of 40 is added to an improvement neighborhood method whenever a new overall best solution is found by it. If the current solution is improved and the solution is new, then a score of 20 is added. Otherwise, a score of 5 is added if the solution is accepted. In the perturbation step, a score of 10 is rewarded to a perturbation neighborhood method if a new overall best solution is obtained, otherwise, a score of 1 is assigned to it. To determine the value of ρ, we conducted a series of algorithm executions with ρ values taken from the range [1,15] with 1 increments. The value of 8 proved to be rather robust yielding the best average solution cost for test problems. Following these results, parameter ρ is set equal to 8.

The final values of other parameters and ranges are shown in Table 3. The same parameters configuration is used on all instances.

5.3. Comparison of the adaptive neighborhood selection with the stochastic neighborhood selection for MDVRPSDP

In MDVRPSDP, the objective is to minimize a weighted sum of the number of vehicles and the travel distances. We set fixed cost α equal to 100 and variable cost β equal to 1.

Table 4 reports the best solutions of the ILS embedded adaptive neighborhood selection and the ILS embedded stochastic neighborhood selection (ILS_SNS). Table 4 shows that ILS_ANS obtains all the best solutions whereas ILS_SNS only obtains the best solution of 4 out of 22 instances within nearly equal average computational time.

Table 1
Basic characteristics of data sets for the MDVRPSDP.

No. of instance	*n*	*p*	*T*	*Q*
GJ1	50	4	–	80
GJ2	50	4	–	160
GJ3	75	5	–	140
GJ4	100	2	–	100
GJ5	100	2	–	200
GJ6	100	3	–	100
GJ7	100	4	–	100
GJ8	249	2	310	500
GJ9	249	3	310	500
GJ10	249	4	310	500
GJ11	249	5	310	500

Table 2
Basic characteristics of data sets for the VRPSDP.

No. of instance	*n*	*T*	*Q*
CMT1	50	–	160
CMT2	75	–	140
CMT3	100	–	200
CMT4	150	–	200
CMT5	199	–	200
CMT6	50	200	160
CMT7	75	160	140
CMT8	100	230	200
CMT9	150	200	200
CMT10	199	200	200
CMT11	120	–	200
CMT12	100	–	200
CMT13	120	720	200
CMT14	100	1040	200

Table 3
Values of the Parameters used by ILS_ANS.

Parameter	Range	Final value	Parameter	Range	Final value
r_0	[0.0,1.0]	0.1	η	–	100
φ	[1,5]	3	λ'	[0.0,1.0]	0.9
θ_1	[0,50]	40	η'	–	50
θ_2	[0,50]	20	γ	[0.1,1.0]	0.2
θ_3	[0,50]	5	δ	[0.1,2.0]	1.0
θ_4	[0,50]	10	ρ	[1,15]	8
θ_5	[0,50]	1	*iter*1	–	[16,000 * *n*/50][a]
λ	[0.0,1.0]	0.1	*iter*2	–	200

[a] [*x*] denotes the nearest integer to *x*.

Table 4
The best solutions of ILS_ANS and ILS_SNS for MDVRPSDP.

No. of instance	ILS_ANS				ILS_SNS			
	Total cost	Routing cost	No. of vehicles	CPU time(s)	Total cost	Routing cost	No. of vehicles	CPU time(s)
GJ1X	1109.13	509.13	6	14	1110.82	510.82	6	15
GJ2X	748.31	448.31	3	18	748.31	448.31	3	18
GJ3X	1204.33	604.33	6	20	1205.53	605.53	6	18
GJ4X	1731.64	831.64	9	55	1743.29	843.29	9	55
GJ5X	1186.80	686.80	5	70	1186.80	686.80	5	61
GJ6X	1670.21	770.21	9	63	1670.92	770.92	9	47
GJ7X	1686.01	786.01	9	71	1686.28	786.28	9	47
GJ8X	4776.97	3376.97	14	604	4790.7	3390.70	14	623
GJ9X	4479.82	3079.82	14	552	4490.61	3090.61	14	456
GJ10X	4364.82	2964.82	14	549	4395.38	2995.38	14	502
GJ11X	4374.48	2974.48	14	507	4463.5	2963.50	15	551
Average	2484.77	1548.41	9.36	229.36	2499.29	1553.83	9.45	217.55
GJ1Y	1109.13	509.13	6	13	1111.24	511.24	6	12
GJ2Y	747.93	447.93	3	17	747.93	447.93	3	18
GJ3Y	1203.03	603.03	6	17	1203.79	603.79	6	20
GJ4Y	1730.85	830.85	9	55	1735.84	835.84	9	47
GJ5Y	1186.80	686.80	5	63	1186.80	686.80	5	60
GJ6Y	1670.72	770.72	9	39	1671.89	771.89	9	47
GJ7Y	1684.39	784.39	9	45	1687.11	787.11	9	48
GJ8Y	4778.05	3378.05	14	594	4782.76	3382.76	14	586
GJ9Y	4481.90	3081.90	14	524	4532.31	3132.31	14	473
GJ10Y	4372.14	2972.14	14	608	4396.72	2996.72	14	508
GJ11Y	4374.30	2974.30	14	491	4379.44	2979.44	14	471
Average	2485.39	1549.02	9.36	224.18	2494.17	1557.80	9.36	208.18

Table 5
Comparison of the average results for the MDVRPSDP.

No. of instance	Salhi and Nagy (1999)			Nagy and Salhi (2005)			ILS_ANS					
	Routing cost	No. of vehicles	CPU Time	Routing cost	No. of vehicles	CPU Time	Initial Solution			Final Solution		
							Routing cost	No. of vehicles	CPU time	Routing cost	No. of vehicles	CPU time
GJ1X	674	14	0.2	_	_	_	560.79	9	0.002	509.13	6	14
GJ2X	569	6	2.3	_	_	_	491.33	5	0.003	448.31	3	18
GJ3X	734	13	1.5	_	_	_	643.71	10	0.004	604.33	6	20
GJ4X	1193	18	1.6	_	_	_	978.78	13	0.048	831.64	9	55
GJ5X	909	10	26.5	_	_	_	748.94	6	0.045	686.80	5	70
GJ6X	954	19	0.7	_	_	_	881.78	12	0.023	770.21	9	63
GJ7X	973	16	1.5	_	_	_	839.66	13	0.013	786.01	9	71
GJ8X	5326	30	52.2	_	_	_	3934.05	19	0.82	3376.97	14	604
GJ9X	4426	26	150	_	_	_	3605.01	20	0.59	3079.82	14	552
GJ10X	4446	31	157	_	_	_	3539.08	19	0.184	2964.82	14	549
GJ11X	4323	31	40.5	_	_	_	3512.93	21	0.116	2974.48	14	507
Average	2230	19.5	39.4	1993		13.9	1794.19 (10.0%)[a]	13.4	0.17	1548.41 (13.7%)[b]	9.4	229.36
GJ1Y	614	12	0.2	_	_	_	560.79	9	0.002	509.13	6	13
GJ2Y	519	5	0.3	_	_	_	491.33	5	0.002	447.93	3	17
GJ3Y	737	14	1.4	_	_	_	643.71	10	0.004	603.03	6	17
GJ4Y	1162	18	1.7	_	_	_	978.78	13	0.048	830.85	9	55
GJ5Y	912	8	26.5	_	_	_	748.94	6	0.205	686.80	5	63
GJ6Y	1003	16	3.1	_	_	_	881.78	12	0.023	770.72	9	39
GJ7Y	973	16	1.5	_	_	_	839.66	13	0.012	784.39	9	45
GJ8Y	4804	27	24.7	_	_	_	3934.05	19	0.818	3378.05	14	594
GJ9Y	4501	31	27.8	_	_	_	3605.01	20	0.348	3081.90	14	524
GJ10Y	4183	29	35.9	_	_	_	3539.08	19	0.184	2972.14	14	608
GJ11Y	4357	31	40.5	_	_	_	3512.93	21	0.111	2974.30	14	491
Average	2160	18.8	14.9	1993		10.9	1794.19 (10.0%)[a]	13.4	0.16	1549.02 (13.7%)[b]	9.4	224.18

[a] The best routing cost obtained from Nagy and Salhi (2005) improved by our initial solution method.
[b] The routing cost obtained from our initial solution method improved by ILS_ANS.

5.4. Comparison of ILS_ANS with other heuristics for MDVRPSDP

Table 5 reports the results obtained by previous heuristics and ILS_ANS for MDVRPSDP. In the previous results, those of Nagy and Salhi (2005) are better. For the instances *X* and *Y*, Table 5 shows that our proposed initial solution approach improves the average value of Nagy and Salhi (2005) by 10%, and the results obtained by ILS_ANS further improve the initial solutions by 13.7%. The main reason that ILS_ANS outperforms other approaches is because they are heuristic approaches which have no perturbation. But the proposed ILS_ANS is a metaheuristic approach which allows perturbation when a current search is trapped in local optima. By

Table 6
Comparison of the best results for VRPSDP.

No. of instance	BKS	Ropke and Pisinger (2006b)			Ai and Kachitvichyanukul (2009)	Çatay (2010)	ILS_ANS			Gap% to BKS
		Std. removals	6R-no learning	6R-normal learning			Routing cost	No. of vehicles	CPU time(s)	
CMT1X	466.77	467	467	467	467	470.67	466.77	3	23	0.00
CMT2X	684.21	702	709	704	710	705.24	684.61	6	26	0.06
CMT3X	721.27	727	731	731	738	726.55	721.40	5	71	0.02
CMT4X	852.46	894	877	879	912	893.9	852.46	7	195	0.00
CMT5X	1029.25	1108	1138	1108	1167	1115.75	1035.22	10	315	0.58
CMT6X	555.43	559	559	559	557	558.68	555.43	6	37	0.00
CMT7X	900.12	905	903	901	919	901.22	900.54	11	64	0.05
CMT8X	865.50	866	866	866	896	865.51	865.50	9	81	0.00
CMT9X	1160.68	1221	1197	1205	1225	1173.44	1161.88	14	231	0.10
CMT10X	1373.40	1494	1490	1462	1520	1424.06	1383.76	18	475	0.75
CMT11X	833.92	875	875	837	895	887.36	846.86	4	149	1.55
CMT12X	662.22	688	683	685	691	681.02	663.54	5	98	0.20
CMT13X	1542.86	1595	1591	1578	1560	1551.25	1542.86	11	146	0.00
CMT14X	821.75	876	863	885	826	821.75	821.75	10	74	0.00
Average	**890.70**	**927**	**925**	**919**	**934.50**	**912.60**	**893.04**	8.50	141.79	**0.26**
CMT1Y	466.77	467	467	467	467	472.37	466.77	3	17	0.00
CMT2Y	684.21	685	691	685	710	704.16	684.75	6	23	0.08
CMT3Y	721.27	734	742	738	740	729.02	721.40	5	78	0.02
CMT4Y	852.46	854	856	876	913	895.25	854.38	7	154	0.23
CMT5Y	1029.25	1131	1132	1146	1142	1112.61	1032.89	10	302	0.35
CMT6Y	555.43	559	559	559	557	556.68	555.43	6	18	0.00
CMT7Y	900.12	969	979	952	934	901.22	900.54	11	25	0.05
CMT8Y	865.50	880	894	873	902	865.51	865.50	9	79	0.00
CMT9Y	1160.68	1267	1256	1271	1230	1171.95	1161.31	14	261	0.05
CMT10Y	1373.40	1567	1573	1552	1485	1429.46	1391.26	18	487	1.30
CMT11Y	833.92	938	956	920	900	874.13	846.57	4	110	1.52
CMT12Y	662.22	673	686	675	697	671.32	663.59	5	64	0.21
CMT13Y	1542.86	1726	1612	1602	1568	1547.75	1542.86	11	107	0.00
CMT14Y	821.75	–	–	–	823	822.35	821.75	10	79	0.00
Average	**890.70**	**957.69**	**954.08**	**947.38**	**933.43**	**910.98**	**893.50**	8.50	128.86	**0.31**
Overall average time(s)		694	679	757	141.18	–			135.33	

perturbation, the search can escape the local optima. Accordingly, the more computational time is needed.

5.5. ILS_ANS applied to VRPSDP

To further evaluate the effectiveness of our algorithm, the ILS_ANS is applied to VRPSDP. To compare the results, we test our approach on the data set generated by Salhi and Nagy (1999). The data set is divided into type *X* and type *Y*, and each type of them contains two classes of instances with max route duration and without max route duration. So far, although most of the papers focusing on VRPSDP have tested their algorithm on the data set, most of them only test VRPSDP instances without max route duration. All the best known solutions (BKS) are summarized in Table 6.

We compare our results with the ones obtained by large neighborhood search (Ropke & Pisinger, 2006b), particle swarm optimization (Ai & Kachitvichyanukul, 2009), ant colony optimization approach (Çatay, 2010). Table 6 shows that the average values obtained by the ILS_ANS are better than those obtained by the three approaches for type *X* and type *Y*. The average gaps% to the best known solutions are equal to 0.26% and 0.31%, respectively, for type *X* and type *Y*, where gap% = (the best solution obtained by ILS_ANS-BKS)/BKS * 100. The comparison shows that the performance of the ILS_ANS is competitive to the existing sophisticated methods for VRPSDP.

6. Conclusions

MDVRPSDP is a very important problem in practical applications. This work presents a metaheuristic to solve it. The proposed algorithm integrates an adaptive neighborhood selection mechanism into ILS, employs different structural neighborhoods in the improvement and perturbation steps, and uses the probability rule to accept a worse solution based on the number of its repetition. The new insertion methods are developed for perturbation neighborhood methods. The effectiveness of our proposed approach is demonstrated by the computational results for MDVRPSDP and VRPSDP. Moreover, this approach is flexible, because it is observed that it can be applied to the related problems by modifying the initial solution and the feasibility check of neighborhood methods. The approach is easy to be extended and adjusted by adding or replacing these neighborhood methods.

The drawback of the proposed approach is that the number of parameters is increased because of the adaptive neighborhood selection mechanism. To simplify the experiment, the values of the parameters are obtained by changing the value of one parameter while keeping the other parameters fixed, omitting their interaction effect among them.

As for future work, we intend to improve this approach from several following ways: (1) exploring more effective adaptive neighborhood selection mechanism by using multiple features instead of the sole objective function. Especially, different features are employed in improvement and perturbation steps to assess the efficiency of the neighborhood method. (2) Considering how to assessing a sequence of neighborhood methods instead of single neighborhood method to improve the adaptive neighborhood selection mechanism, because a local optima is often obtained by a sequence of neighborhood methods. (3) Improving the parameter experiment method to obtain the better combination of parameter values, such as automated parameter tuning, because the drawback of manual tuning is very time consuming and failure-prone. (4) Designing the more robust neighborhood methods to

strengthen the intensification, such as compound neighborhood which can execute several simple neighborhood moves simultaneously.

Acknowledgment

The authors gratefully acknowledge valuable comments of two anonymous referees, which considerably improved the quality of this paper. This work is supported by a Research Grant from National Natural Science Foundation of China (No. 71001053, 71231004) and partially supported by LATNA Laboratory, NRU HSE, RF government Grant (ag. 11.G34.31.0057).

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*, 1693–1702.

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

Campbell, A. M., & Savelsbergh, M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science, 38*, 369–378.

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

Chakhlevitch, K., & Cowling, P. (2008). Hyperheuristics: Recent developments. In C. Cotta et al. (Eds.), *Adaptive and multilevel metaheuristics SCI 136* (pp. 3–29). Heidelberg: Springer.

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

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, P. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial optimization* (pp. 315–338). Chichester: Wiley.

Clark, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operation Research, 12*, 568–581.

Cordeau, J.-F., & Maischberger, M. (2012). A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research, 39*, 2033–2050.

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

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 Spectrum, 23*, 79–96.

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

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

Gillett, B. E., & Johnson, J. G. (1976). Multi-terminal vehicle-dispatch algorithm. *Omega, 4*, 711–718.

Goksal, F. P., Karaoglan, I., & Altiparmak, F. (2013). A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering, 65*, 39–53.

Lourenço, H. R., Martin, O., & Stützle, T. (2001). A beginner's introduction to iterated local search. In *Proceedings of 4th metaheuristics international conference*.

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*, 595–619.

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

Or, I. (1976). Traveling salesman-type combinational problems and their relation to the logistics of blood banking (PhD thesis). USA: Northwestern University.

Potvin, J. Y., Kervahut, T., Garcia, B. L., & Rousseau, J. M. (1992). A tabu search heuristic for the vehicle routing problem with time windows. Technical Report CRT-855, Quebec, Canada: Universite de Montreal.

Ropke, S., & Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*, 455–472.

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

Salhi, S., & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *The Journal of the Operational Research Society, 50*, 1034–1042.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics, 159*, 139–171.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings CP-98 (fourth international conference on principles and practice of constraint program-ming)*.

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 & Operations Research, 37*, 1899–1911.

Subramanian, A., Uchoa, E., & Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research, 40*, 2519–2531.

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

Tasan, A. S., & Gen, M. (2012). A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering, 62*, 755–761.

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 pickup service. *Expert Systems with Applications, 36*, 1070–1081.

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