Computers & Industrial Engineering 83 (2015) 111–122

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows

Chao Wang [a], Dong Mu [a,*], Fu Zhao [b,c], John W. Sutherland [b]

[a] School of Economics and Management, Beijing Jiaotong University, Room 718, Si Yuan Dong Lou, No. 3 Shang Yuan Cun, Hai Dian District, Beijing 100044, China
[b] Environmental and Ecological Engineering, Purdue University, Potter Engineering Center, Room 364, 500 Central Drive, West Lafayette, IN 47907, United States
[c] School of Mechanical Engineering, Purdue University, ME 2194, 585 Purdue Mall, West Lafayette, IN 47907, United States

A R T I C L E I N F O

Article history:
Received 20 March 2014
Received in revised form 6 February 2015
Accepted 12 February 2015
Available online 21 February 2015

Keywords:
Parallel simulated annealing
Simultaneous delivery and pickup
Vehicle routing
Time windows

A B S T R A C T

This paper addresses a variant of the vehicle routing problem in which customers require simultaneous pickup and delivery of goods during specific individual time windows (VRPSPDTW). A general mixed integer programming model is employed to minimize the routing cost due to: the cost of vehicles and the travel cost of vehicles. A parallel Simulated Annealing (p-SA) algorithm that includes a Residual Capacity and Radial Surcharge (RCRS) insertion-based heuristic is developed and applied to solve this NP-hard optimization problem. Computational results are reported for 65 test problems from Wang and Chen's benchmark and compared with the results from a Genetic Algorithm (GA) that minimizes the number of vehicles (NV) as the primary objective. Experimental results demonstrate the effectiveness of the p-SA algorithm, which is able to achieve the same objective response as 100% of the Wang and Chen small-scale benchmarks (number of customers from 10 to 50). For the Wang and Chen medium-scale benchmarks (number of 100 customers), the p-SA algorithm obtains better NV solutions for 12 instances and the same NV solutions for the remaining 44 instances. For the 44 instances with the same NV solutions, a secondary objective, travel distance (TD), the p-SA provides better solutions than the GA for 16 instances, and equal solutions for 7 instances. In addition, solutions are found for 30 large-scale instances, with customers of 200, 400, 600, 800 and 1000, which may serve as a new benchmark for the VRPSPDTW problem.

1. Introduction

Owing to fossil fuel consumption and airborne emissions, freight transportation has a significant effect on the environment. In the U.S, truck freight transportation is the dominant domestic freight mode and optimization of truck routing and scheduling will reduce the environmental impact of this sector. Many enterprises have initiated efforts to incorporate reverse logistics into their regular delivery and distribution systems to reduce costs associated with energy consumption, and satisfy applicable regulations and laws (Dethloff, 2001; Wang & Hsu, 2010). Such an integrated system has found applicability in a wide variety of fields such as library books distribution (Min, 1989), grocery distribution system (Zachariadis, Tarantilis, & Kiranoudis, 2009), parcel delivery (Berbeglia, Cordeau, & Laporte, 2010), and home health care service (Liu, Xie, Augusto, & Rodriguez, 2013). In the literature, this problem has attracted research interest because it models a wide variety of business operations involving bi-directional flow of goods, and has been referred to as the pickup and delivery problem (PDP). Toth and Vigo (2002), and Berbeglia, Cordeau, Gribkovskaia, and Laporte (2007) have provided excellent surveys of models for the PDP. The PDP can be divided into four situations: (i) delivery-first, pickup-second for the vehicle routing problem with back-hauls (VRPB), e.g. (Goetschalckx & Jacobs-Blecha, 1989; Salhi & Nagy, 1999), (ii) simultaneous pickups and deliveries for the vehicle routing problem with simultaneous pickup and delivery (VRPSPD), e.g. (Min, 1989), (iii) mixed deliveries and pickups for the vehicle routing problem with mixed pickup and delivery (VRPMPD), e.g. (Nagy & Salhi, 2005; Wassan, Wassan, & Nagy, 2008), and (iv) inter-related pickups and deliveries for the dial-a-ride problem, e.g. (Diana & Dessouky, 2004; Lu & Dessouky, 2004).

This paper investigates an extended situation of the VRPSPD problem, in which enterprises allow customers to request their goods to be delivered and picked up within pre-defined time

* Corresponding author at: Room 718, Si Yuan Dong Lou, Beijing Jiaotong University, No.3 Shang Yuan Cun, Hai Dian District, Beijing 100044, China. Tel.: +86 138 0129 2938.

E-mail addresses: chaowang.hn@gmail.com (C. Wang), mudong.bjtu@gmail.com (D. Mu), fzhao@purdue.edu (F. Zhao), jwsuther@purdue.edu (J.W. Sutherland).

http://dx.doi.org/10.1016/j.cie.2015.02.005
0360-8352/© 2015 Elsevier Ltd. All rights reserved.

windows in order to provide satisfactory service (Wang & Chen, 2013). In the literature this problem is usually termed the vehicle routing problem with simultaneous pickup and delivery and time windows (VRPSPDTW). Time windows define the earliest and latest time that a vehicle may arrive at a customer and start service. A soft time window is nonbinding and there is no penalty on violation. On the other hand, a hard time window cannot be violated, i.e., if a vehicle arrives before the time window opens, it must wait until the time window opens; and it is not allowed to arrive after the time window has closed. In this paper hard time windows are considered. The VRPSPDTW problem involves a fleet of homogenous or non-homogenous vehicles stationed at a depot to serve different geographically scattered customers. The vehicles are not only required to deliver goods from the depot to customers but also simultaneously to pick up goods at the customer locations for return to the depot, without violating vehicle capacity constraints and the defined time windows specified by the customers.

The VRPSPDTW is an NP-hard combinatorial optimization problem (Wang & Chen, 2012) since it can be reduced to a VRPSPD (which is NP-hard) problem when all the customers' earliest service times are equal to the depot's open time, and the latest service times are equal to the depot's closing time. Since VRPSPDTW is NP-hard, exact algorithms can only be used to find solutions for small-and-medium scale instances. The commercial linear programming software CPLEX has been reported to solve the VRPSPDTW problem executed on an Intel Core2 Quad 2.4G computer with 1G memory, and was only able to solve a 10 customer instance, and a portion of the 25 and 50 customer scenarios (Wang & Chen, 2012). For the solved 50 customers instance, the computation time was 327,404 s. Therefore, large-scale instances of VRPSPDTW cannot be solved by exact solution methodologies within an acceptable computational time. Due to the computational challenge associated with applied problems that involve large numbers of customers, researchers and practitioners are usually interested in developing heuristic or meta-heuristic approaches to produce high-quality solutions (but not necessarily optimal solutions) with reasonable computational times. Among the solution methods for VRP and its variants, parallel heuristic and metaheuristic methods have emerged in recent years (Jin, Crainic, & Løkketangen, 2012). Unlike sequential algorithms that run a search process on a single processor, parallel algorithms guide the search by a cooperative multi-processor (or thread) mechanism that offers versatile, robust and powerful tools to address large and complex VRPs (Crainic, 2008). For a survey of parallel solution methods for the VRP, the reader is referred to the books of Crainic (2008).

Simulated Annealing (SA) is a strong, robust, and versatile metaheuristic, which is easy to develop and implement. Many combinatorial optimization problems can be solved efficiently by SA. It has been used to solve the VRP and its variants, e.g., VRP (Osman, 1993), VRPTW (Chiang & Russell, 1996), time dependent VRP (Kuo, 2010), and competitive VRPTW (Tavakkoli-Moghaddam, Gazanfari, Alinaghian, Salamatbakhsh, & Norouzi, 2011). However, an application that combines parallel processing and SA to solve a VRP is rare. Czech and Czarnas (2002) proposed a p-SA (parallel SA) to solve the VRPTW problem, and Baños, Ortega, Gil, Fernández, and de Toro (2013) proposed a p-SA to solve the multi-objective VRPTW problem. Hence, the application of p-SA to solve VRP and its variants is still an open research field.

The purpose of this study is to develop an efficient meta-heuristic to solve the VRPSPDTW problem using a p-SA algorithm. The proposed p-SA is simple, versatile, and robust. The remainder of this paper is organized as follows. Section 2 presents a literature review on the VRPSPDTW, VRPSPD, and VRPTW. Section 3 formally defines the VRPSPDTW to be considered, and develops a mathematical model. Section 4 gives a detailed description of how the proposed p-SA algorithm is implemented to address the VRPSPDTW problem. Section 5 compares the performance of the p-SA with solutions reported in the literature for a variety of benchmarking instances and proposes a new benchmark for large VRPSPDTW problems. Finally, conclusions are drawn in Section 6.

2. Literature review

As stated before, three main bodies of vehicle routing literature are relevant to our problem. The first is the vehicle routing problem with simultaneous pickup–delivery and time windows. The second and third literature areas are special cases of this basic situation, i.e., the vehicle routing problem with simultaneous pickup and delivery and vehicle routing problem with time windows. The technical literature on the VRPSPDTW problem is relatively sparse compared to the body of literature accumulated for the VRPSPD and VRPTW problems.

Angelelli and Mansini (2003) were the first and only researchers to solve the VRPSPDTW problem with an exact algorithm. They implemented a branch-and-price approach based on a set covering formulation for the master problem. A relaxation of the elementary shortest path problem with time windows and capacity constraints was used for the pricing problem. A branch-and-bound approach was applied to obtain integer solutions. They modified Solomon's 100 customers instances and assumed the pickup amount P_i corresponding to the delivery amount D_i is computed by $P_i = (1-\alpha)D_i$ if i is even and $P_i = (1+\alpha)D_i$ if i is odd, where $0 \leqslant \alpha \leqslant 1$. Twenty-nine 20 customers instances were generated. The authors reported an average CPU time for these 29 instances of 26.58 s and 10.51 s for parameter α values of 0.2 and 0.8 respectively.

Several heuristics algorithms has been proposed to solve the VRPSPDTW problem. Lai and Cao (2010) proposed an Improved Differential Evolution (IDE) algorithm for solving this problem and did numerical experiments with their own instances. Boubahri, Addouche, and El Mhamedi (2011) constructed a multi-agent colonies algorithm for the problem, but did not test the algorithm. Wang and Chen (2012) proposed a co-evolution genetic algorithm with variants of the cheapest insertion method for VRPSPDTW. They also developed 65 instances adapted from the well-known Solomon benchmark (Solomon, 1987) for VRPTW. Liu et al. (2013) proposed a GA and a Tabu Search (TS) method to solve a practical vehicle scheduling problem encountered in home health care logistics.

The VRPSPD problem is better studied than the more specific class of problems noted above. VRPSPD was proposed by Min (1989) for a real life application which involved 22 customers and two vehicles to solve the book distribution problem facing a public library system. The solution was obtained by clustering customers into two disjoint groups and solving the traveling salesperson problem (TSP) for each group. Then, there was a gap of more than 10 years with limited work on this problem (Montane & Galvao, 2006). Recently, a number of researchers have revisited the problem owing to the growing interest in reverse logistics and the increasing focus on environmental protection.

For the vehicle routing problem, having only 15 customers can result in more than 10^{12} feasible solutions (Phannikul & Sindhuchao, 2010), due to the fact that the number of feasible solutions grows factorially with the number of customers. Therefore, very few exact approaches have been explored for this problem, such as: branch-and-price technique with one-commodity flow formulation (Dell'Amico, Righini, & Salani, 2006), branch-and-cut approach (Rieck & Zimmermann, 2009), branch-and-cut over one-commodity and two-commodity flow formulations (Subramanian, Uchoa, & Ochi, 2010), branch-and-cut with lazy separation (Subramanian, Uchoa, Pessoa, & Ochi, 2011) and branch-cut-and-price algorithm (Subramanian, Uchoa, Pessoa, &

Ochi, 2013). Heuristics and meta-heuristics have dominated approaches for VRPSPD problems. Some classic heuristic approaches have been proposed to solve the VRPSPD problem. Salhi and Nagy (1999) developed a cluster insertion heuristic. Dethloff (2001) studied the problem from the view of reverse logistics, and proposed a Residual Capacity and Radical Surcharge (RCRS) algorithm. Nagy and Salhi (2005) applied VRP heuristic routines which are modified to tackle the VRPSPD problem.

Given the fact that the quality of solutions produced by meta-heuristics are much higher than that obtained by classical heuristics (Toth & Vigo, 2002), a considerable amount of research has been done to develop meta-heuristics for the problem in the past decade. Early research into solution techniques for the VRPSPD problem focused on the Tabu Search (TS), which has been proved a very effective algorithm to provide a good compromise between near-optimal solutions and computational time (Montane & Galvao, 2006). Various techniques, such as variable neighborhood descent (VND) (Crispim & Brandão, 2005), record-to-record travel approximation (Chen & Wu, 2006), variable neighborhood search (VNS) (Bianchessi & Righini, 2007), reactive mechanism (Wassan et al., 2008), guided local search (GLS) (Zachariadis et al., 2009), and static move descriptor (SMD) entities (Zachariadis & Kiranoudis, 2011) have also been proposed to enhance the performance of TS to solve the VRPSPD problem. In recent years, several authors have applied other meta-heuristics methodologies for VRPSPD problem that can be categorized into two main types: single solution based meta-heuristics and population based meta-heuristics (Lin, Choy, Ho, Chung, & Lam, 2014). The first one starts with an initial solution that is perturbed to explore the search space of the problem addressed. Greedy Randomized Adaptive Search Procedure (GRASP) (Subramanian, Cabral, & Carvalho, 2007), Iterated Local Search (ILS) algorithm (Souza, Mine, Silva, Ochi, & Subramanian, 2011; Subramanian & Cabral, 2008; Subramanian, Cabral, & Ochi, 2008; Subramanian, Drummond, Bentes, Ochi, & Farias, 2010; Subramanian, Uchoa, & Ochi, 2013) are the main methods that have been explored. Population search based metaheuristics maintain and improve multiple candidate solutions, often using population characteristics to guide the search. Typical algorithms for the VRPSPD problem are Ant Colony Optimization (ACO) (Gajpal & Abad, 2009; Çatay, 2010), Genetic Algorithm (GA) (Tasan & Gen, 2012; Vidal, Crainic, Gendreau, & Prins, 2014) and Particle Swarm Optimization (PSO) (Ai & Kachitvichyanukul, 2009; Gan, Wang, Li, & Niu, 2012; Wei, Zhang, & Tang, 2011).

Numerous studies in the literature have considered VRPTW, and some comprehensive reviews have been conducted, e.g., Cordeau, Desaulniers, Desrosiers, Solomon, and Soumis (2002), Bräysy and Gendreau (2005a), Bräysy and Gendreau (2005b), Kallehauge, Larsen, Madsen, and Solomon (2005), Kallehauge, 2008, Gendreau and Tarantilis (2010), and Baldacci, Mingozzi, and Roberti (2012). Some exact methods have been developed for the VRPTW, which can be classified into three categories: Lagrange relaxation-based methods, column generation, and dynamic programming (El-Sherbeny, 2010). However, high computational cost and poor performance of exact methods in solving large problem instances has prompted researchers to concentrate on heuristic and meta-heuristic algorithms. Some examples are: Simulated Annealing (Baños et al., 2013), Genetic Algorithms (Vidal, Crainic, Gendreau, & Prins, 2013), Tabu Search (Moccia, Cordeau, & Laporte, 2011), Memetic Algorithms (Nagata, Bräysy, & Dullaert, 2010), and Particle Swarm Optimization (Marinakis, Marinaki, & Dounias, 2010); results from these methods have produced good solutions in a relatively reasonable time.

Although significant literature has been published on the VRPSPD and VRPTW problems, few studies have attempted to solve the VRPSPDTW and there is a lack of highly efficient methods to solve it. Moreover, to the best of our knowledge, no effort has been undertaken to solve the VRPSPDTW by a simulated annealing approach. Meanwhile, three key observations may be made for the all solution approaches in the literature: (1) a majority of various VRPs were solved by heuristics and meta-heuristics, (2) the best known meta-heuristics developed for the VRPSPD and VRPTW typically identify better local optima than earlier heuristics, but they also tend to be more time intensive, and (3) while developing meta-heuristics, a heuristic approach is often used to obtain a good initial solution or population of initial solutions.

3. Problem formulation

This section provides a formal description of the VRPSPDTW problem and then presents a mixed integer programming (MIP) formulation of the problem. VRPSPDTW is described as choosing routes for a limited number of vehicles to serve a group of customers with specific time windows. The problem constraints consist of a set of homogenous vehicles, a central depot node, a set of geographic customer nodes, and a network connecting the depot and customers. The description can be defined on a complete graph $G = (V_0, E)$ with $V_0 = \{1, 2, \ldots, n+1\}$ as the nodes set and E as the arc set defined between each pair of nodes, $E = \{\langle i,j\rangle | i,j \in V_0, i \neq j\}$.

Customers: The customers are noted as 2, 3, …, $n+1$. For notational convenience, we set the central depot as a special customer corresponding to node 1. The depot acts as the distribution center and collection center. The set V $(V = \{2, 3, \ldots, n+1\})$ represents the set of customers. The set V_0 $(V_0 = V \cup \{1\})$ represents the set of customers plus depot.

Demand: Every customer $i \in V$ has a fixed delivery demand D_i, and a pickup demand P_i. D_i represents the amount of goods to deliver from the depot to customer i. P_i represents the amount of goods to pick up from customer i, that must be delivered to the depot. The different kinds of goods are assumed to be compatible and can be loaded onto the same vehicle. The depot does not have delivery and pickup requirements.

Vehicles: A fleet of identical vehicles $(K = \{1, 2, \ldots, m\})$, each with a capacity of Q and dispatching cost c_d, is initially located in the depot which is available to service the customers. There is no restriction on the number of vehicles. Each vehicle starts at the depot and ends at the depot.

Time windows: The depot and customers have time windows. The depot has a time window $[a_1, b_1]$. Vehicles may not leave the depot before a_1 and cannot return after b_1. The ith customer has a pre-specified service time interval of $[a_i, b_i]$. The lower bounds a_i and upper bounds b_i of the time window define the earliest and latest service time of customer i. The time windows in this paper are treated as hard constraints; therefore, arrival of a vehicle at the ith customer before the time window lower limit, a_i, results in a wait before service can begin. On the other hand, arrival at the customer after the time window upper limit, b_i, is infeasible. The ith customer also has a specified service time, t_i, which is the time spent by the vehicle to load and unload the goods.

Distance: The distance between customer i and j is based on the Euclidean distance, d_{ij}; the distances also satisfy the triangular inequality $d_{ij} + d_{jk} \geqslant d_{ik}$.

Speed: The vehicle (travel) speed between customers is 1, which means the travel time t_{ij} between node i and node j equals the travel distance d_{ij} between node i and node j $(i,j \in V_0, i \neq j)$. If the speed varies, the VRPSPDTW problem changes to a time-dependent VRPSPDTW problem, which is beyond the research scope of the present research.

Routes: A route starts at the depot, visits a number of customers (at most once), and then returns to the depot. In other words, a route is a sequence $\{1, V_1, V_2, \ldots, 1\}$, where all V_i are different.

Each arc $\{i,j\} \in E$, has an associated distance, d_{ij}, and the travel/transportation cost per unit distance is c_t.

The objective of the VRPSPDTW problem is to find routes for vehicles that serve all the customers at a minimal cost (in terms of TD, NV, waiting time (WT), etc.) and satisfy the following assumptions: (1) each customer is visited once and only once by exactly one vehicle, (2) each vehicle serves only one route, (3) at each customer site, pickup goods are loaded after delivery goods are unloaded, (4) no vehicle is allowed to transport good in excess of its capacity, and (5) no service time window may be violated.

A MIP mathematical formulation of the VRPSPDTW problem is considered in this paper. The MIP is derived from Dethloff (2001) and Kallehauge et al. (2005). Four types of decision variables are used, which will help in the description of the method used for solving the VRPSPDTW.

x_{ijk}	Traveling variable of a vehicle $k \in K,\ x_{ijk} \in \{0,1\}$. $x_{ijk} = 1$ if it goes directly from node $i \in V_0$ to node $j \in V_0$; otherwise $x_{ijk} = 0$
L_{1k}	Load of vehicle $k \in K$ when leaving the depot
L_j	Load of vehicle after having serviced customer $j \in V$
s_{ik}	Time of beginning of service at customer $i \in V$ by, it is 0 if vehicle k does not serve customer i

Based on these variable definitions, the MIP problem may be stated as:

$$Minimize Z = \sigma \sum_{j\in V_0}\sum_{k\in K} c_d x_{1jk} + (1-\sigma)\sum_{i\in V_0}\sum_{j\in V_0}\sum_{k\in K} c_t d_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \sum_{i\in V_0}\sum_{k\in K} x_{ijk} = 1 \quad \forall j \in V \tag{2}$$

$$\sum_{i\in V_0} x_{ihk} = \sum_{j\in V_0} x_{hjk} \quad \forall h \in V;\ \forall k \in K \tag{3}$$

$$\sum_{j\in V_0} x_{ojk} = 1 \quad \forall k \in K \tag{4}$$

$$\sum_{j\in V} x_{1jk} = \sum_{i\in V} x_{i1k} \quad \forall k \in K \tag{5}$$

$$\sum_{i\in V_0} x_{i,0,k} = 1 \quad \forall k \in K \tag{6}$$

$$L_{1k} = \sum_{i\in V_0}\sum_{j\in V} D_j x_{ijk} \quad \forall k \in K \tag{7}$$

$$L_j \geqslant L_{1k} - D_j + P_j - M(1 - x_{1jk}) \quad \forall j \in V;\ \forall k \in K \tag{8}$$

$$L_j \geqslant L_i - D_j + P_j - M\left(1 - \sum_{k\in K} x_{ijk}\right) \quad \forall i \in V; \quad \forall j \in V;\ i \neq j \tag{9}$$

$$L_{1k} \leqslant Q \quad \forall k \in K \tag{10}$$

$$L_j \leqslant Q + M\left(1 - \sum_{i\in V_0} x_{ijk}\right) \quad \forall j \in V;\ \forall k \in K \tag{11}$$

$$s_{ik} + t_i + t_{ij} - M\left(1 - \sum_{\forall k\in K} x_{ijk}\right) \leqslant s_{jk} \quad \forall i \in V_0;\ \forall j \in V_0 \tag{12}$$

$$a_i \leqslant s_{ik} \leqslant b_i \quad \forall i \in V_0;\ \forall j \in V_0;\ \forall k \in K \tag{13}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in V_0;\ \forall j \in V_0;\ \forall k \in K \tag{14}$$

Most exact methods and some meta-heuristics for the VRPTW or VRPSPD minimize NV, and some authors have minimized TD (Dethloff, 2001; Montane & Galvao, 2006) or a hierarchical objective of minimizing NV (primary objective) and TD (secondary objective) (Wang & Chen, 2012). The objective function of VRPSPDTW in this paper also seeks to minimize the hierarchical objective used in Wang and Chen's (2012) algorithm, where σ is a parameter that trades-off dispatching cost and travel cost. Constraints (2) ensure that each customer is served by one and only one vehicle. Constraints (3) make sure the same vehicle k enters and leaves each customer, and provides for route continuity. Constraints (4)–(6) guarantee that each vehicle starts from the depot, after a vehicle arrives at a customer it has to leave for another destination, and finally terminates at the depot. Constraints (7)–(9) are vehicle load constraints associated with initial vehicle loads, vehicle loads after the first customer and vehicle loads 'en route', respectively. M is an arbitrary large constant. Constraints (10) and (11) are vehicle capacity constraints, which require that a vehicle's specified capacity is never exceeded by the quantity of goods. Constraints (12) and (13) are time window constraints, which ensure feasibility of the time schedule. Finally, constraints (14) specify the variables used in the formulation.

In the constraints of (8), (9), (11) and (12), the so-called "big M" method is used. For example, constraints (8) can be written as:

$$L_j \geqslant (L_{1k} - D_j + P_j)x_{1jk} \quad \forall j \in V; \forall k \in K \tag{15}$$

If constraints (15) are replaced by (8), this is an example of a Mixed Integer Linear Program (MILP), i.e., a linear program where some of the variables are constrained to be integers. It is easy to express this formulation in a MILP solver, commercial or otherwise (Hasle & Kloster, 2007). In the constraints (8), if a vehicle k ($\forall k \in K$) goes directly from the depot 1 to node j ($j \in V_0$), then $x_{1jk} = 1$ and vehicle loads after the first customer L_j equals to $L_{1k} - D_j + P_j$; otherwise, $x_{1jk} = 0$ and big M ensures that the right part of inequality (6) is negative infinity, so L_j always larger than $-\infty$.

4. Parallel simulated annealing applied to VRPSPDTW

Parallel simulated annealing algorithm for the VRPSPDTW problem is a parallel construction procedure that uses an initial solution as a starting basis for seeking improved solutions by searching different neighborhoods with a SA route improvement approach. In this section, we first give brief information about the Residual Capacity and Radial Surcharge (RCRS) insertion-based heuristic to generate an initial solution, the definition of neighborhoods, classic sequential simulated annealing, and then describe the parallelism of SA using a master–slave paradigm.

4.1. An initial solution

Most metaheuristic search strategies involve finding an initial feasible solution and then improving on that solution using local or global optimization techniques. Here, routes are constructed by solving the VRPSPDTW problem using the Residual Capacity and Radial Surcharge (RCRS) insertion-based heuristic, first introduced by Dethloff (2001) as a method to create an initial route configuration. RCRS is an efficient method to insert customers into new routes.

RCRS is an extension of the cheapest insertion procedure that takes into account travel distance, residual capacity, and radial surcharge. A detailed description of the algorithm can be obtained from Dethloff (2001). In short, RCRS can be summarized as follows. RCRS is an improved cheapest insertion approach. Usually a simple cheapest insertion approach only computes the extra distance when inserting a new customer, which can be viewed as being "short-sighted". RCRS modifies the travel distance insertion criterion at two points. First, the remaining vehicle capacity after a potential insertion is taken into consideration (residual capacity insertion criterion), which seeks to measure the degrees of freedom for future insertions. Second, the distance of customers to the depot is taken into consideration (radical surcharge insertion criterion), which seeks to avoid remotely located customers being "left over" to a late stage of the insertion procedure and resulting in unfavorable extra travel distances.

4.2. Local search procedure

A local search procedure is utilized to reduce the total dispatching cost and total travel cost of a candidate solution by iteratively moving to a neighboring solution. In this paper, the neighboring solutions are obtained by applying four traditional types of improvement methods: Or-opt move, 2-opt* move, λ-interchange, and swap/shift. These moves are commonly embedded in an SA algorithm, and other meta-heuristics. A brief description of these improvement methods are given here for completeness. Please refer to (Bräysy & Gendreau, 2005a) for a comprehensive overview.

Or-opt move (Or, 1976): It removes a strings of three, two, or one consecutive customers from a route, and reinserts the strings elsewhere on the same or on different route, which is a special case of the cross-exchange move (Taillard, Badeau, Gendreau, Guertin, & Potvin, 1997).

2-opt* move (Potvin & Rousseau, 1995): It removes two arcs from two different routes to divide each route into two parts and swaps the end parts of the two routes. It is a variant of the 2-opt move (Lin, 1965) to preserve the orientation of the routes and also a special case of the cross-exchange move (Taillard et al., 1997).

λ-interchange move (Osman, 1993): It selects two subsets of customers, with cardinality less than or equal to λ, from two different routes and exchanges them. In this paper, we only consider the case of $\lambda = 2$, which means a maximum of two customer nodes may be interchanged between routes and leads to a cross-exchange move.

Swap/shift move (Chen & Wu, 2006): It combines the relocation move and exchange move, which either shifts a customer from one route to a different position in the same or a different route, or exchanges a customer on one route with a customer on another route.

Given a solution, s_k, repeatedly evaluate other solutions, s'_k, until no better solution is found in its neighborhood, $N(s_k)$; improved solutions reachable from s_k are sought using one of the above moves. When an improvement has not been made for a certain number of iterations for a given move/operator, another improvement operator is explored.

4.3. Sequential simulated annealing

Simulated annealing is an approximate local search meta-heuristic, described by Kirkpatrick and Vecchi (1983), which adapts the Metropolis–Hastings algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953). It has been applied successfully to a wide variety of highly complicated combinatorial optimization problems as well as various real-world problems (Suman & Kumar, 2005). The key feature of SA compared with other local search algorithms is that SA does not search for the best solution in the neighborhood of the current solution. Instead, a random solution is drawn from the neighborhood and it is either accepted or rejected depending on its relative cost. An improved or unchanged solution is always accepted while a fraction of inferior solutions are also accepted in the hope of escaping from being trapped at a local optimum in pursuit of a global optimum. The probability of accepting inferior solutions depends on the current temperature. SA will converge to the global optimum with a probability of 1 if the initial temperature is sufficiently large and the cooling schedule is very slow provided that the algorithm is convergent; otherwise, the algorithm will converge to a local optimum, which may or may not be the global optimum (Nikolaev & Jacobson, 2010). As pointed out by Dowsland (1993), the "temperature" is used to imitate the cooling process in physical annealing. It is merely a control parameter that controls the probability of accepting the total NV increase in the VRPSPDTW problem. A high temperature translates into a high probability of accepting a solution, s'_k, as the new solution. The framework of this algorithm is shown in Fig. 1.

The pseudo-code shown in Fig. 2 describes the steps in the sequential SA algorithm as applied to solve the VRPSPDTW problem.

It is to be noted that $\text{cost}(s_k) = \sigma c_d(c \times n + e_{\min}) + (1-\sigma)c_t d$, $T_k = \gamma \times \text{cost}(s_k)$, and $T_{k+1} = \beta \times T_k$, where s_k is the feasible solution, c is the number of routes in solution s_k (equal to the number of vehicles needed), n is the number of customers, $e_{\min}$ is the number of customers in the shortest route, d is the total travel distance of the routes, $\gamma < 1$ and is a constant, β is the cooling ratio, k is the temperature counter, ω is a counter that describes the number of iterations (and temperature reductions) where the current best solution has not been improved, and l is the iteration counter.

4.4. Parallelization using the master–slave paradigm

In sequential SA, high-quality solutions are guaranteed by the cooling schedule, particularly when a sufficiently high initial temperature, slow annealing speed, a large number of iterations, and a holding time at the same temperature are utilized. This is usually at the price of high computation resources and time. Therefore, parallelization of SA offers the possibility to speed up the search, and improve the robustness and the quality of the solutions obtained relative to non-parallelized sequential algorithms.

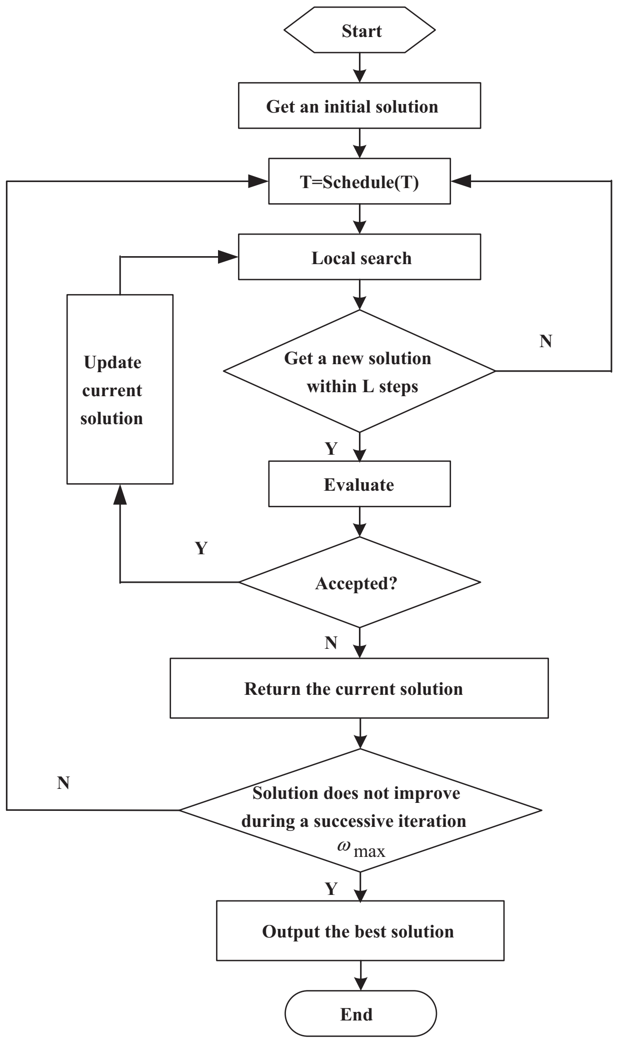

Fig. 1. Framework of the simulated annealing algorithm.

```
1.  $s := s_0, T_0 := \gamma \times \text{cost}(s_0)$
2.  $k := 0,\ \omega := 0$
3.  WHILE ( $\omega < \omega_{\max}$ ) DO
4.      $l := 0$ , updateOptimalSolution := true
5.      WHILE ( $l < L$ ) DO
6.          BEGIN
7.              $s'_k$=local search($s_k$) $s'_k \in N(s_k)$
8.              IF  $\text{cost}(s'_k) < \text{cost}(s_k)$ , THEN
9.                  $s_k := s'_k$ , $l = 0$ , updateOptimalSolution = true , ELSE
10.                 $s_k := s'_k$ with probability  $e^{\text{cost}(s_k)-\text{cost}(s'_k)/T_k}$ , $l := l+1$
11.             END IF
12.     END WHILE
13.     $k := k+1,\ T_{k+1} := \beta \times T(k)$
14.     IF  !updateOptimalSolution
15.     $\omega := \omega + 1$
16. END WHILE
17. Return  $s_k$
```

Fig. 2. Pseudo-code of sequential SA for VRPSPDTW.

Some strategies to parallelize the simulated annealing algorithm include move acceleration, parallel moves, multiple Markov chains (MMC), and speculative computation (Chandy, Kim, Ramkumar, Parkes, & Banerjee, 1997). The MMC method is the most common strategy used for parallelization. The simulated annealing algorithm can be modeled as a process that performs a slight modification from the current configuration to generate a neighbor. This process can be modeled as a Markov chain. Asynchronous MMC strategy splits a Markov chain in sequential simulated annealing into several Markov chains, which can independently explore the entire search space by performing one random move, evaluating the solution, and modifying the configuration if the move is accepted. This process gets scalable parallelization and breaks the bottleneck of intrinsic sequentiality (Ferreiro, García, López-Salas, & Vázquez, 2012) in the classic simulated annealing algorithm. However, speedup is not achieved since each thread (or processor) performs the same amount of work as the sequential algorithm individually and asynchronously. In order to speed up the optimization process, memorizing a Markov chain's current best solution (estimate of the optimum) and exchanging the current best solution among the different Markov chains periodically is necessary.

Synchronous MMC (Lee & Lee, 1992) compares solutions at periodic intervals. With this method, each thread represents a Markov chain. Each Markov chain of constant length N runs independently beginning with an initial solution, s_0, until the next target temperature is reached. Once reaching a target temperature, all threads communicate their current cost value, cost(s^p), $p = 0, 1, \ldots, W-1$. The cost(s^p) of W threads are compared; the minimum across the threads p^Ω, is then identified, and the current local best solution (LBS), s^{p^Ω}, is stored as the current global best solution (GBS). Then, s^{p^Ω} is used as the starting solution for the next temperature level. In the case of two or more LBS s^{p^Ω}, s^{p^Υ}, …, the algorithm selects one of them and this choice does not affect the final result. If each independent Markov chain is regarded as a different individual in a genetic algorithm, the reduce operator can be understood as a crossover operation of a genetic algorithm to select the evolution of these species (Ferreiro et al., 2012).

Czech, Mikanik, and Skinderowicz (2010) employed an integrated asynchronous and synchronous MMC approach (see Fig. 3); they suggested that the probability of moving from one solution to another depends not only on the costs of these solutions and the current temperature (exploration process) but also on the best

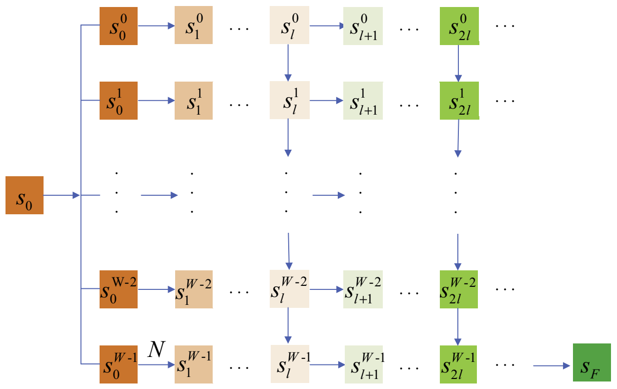

Fig. 3. Integrated MMC approach.

solution s^{p^Ω} computed by the left neighbor in a line of a cooperating thread (exploitation process). The combination of exploration and exploitation is utilized in many parallel heuristics and a good ratio between exploration and exploitation is needed to make the search effective (Črepinšek, Liu, & Mernik, 2013). In their approach, after cooling to a target temperature, T, the co-thread p updated its current LBS $\pi_\ell^{(p)}$ to a better solution in step ℓ, based on the comparison between the LBS $\Lambda_T(V_{\ell-1}^{p-1})$ found by its left neighbor in step $\ell - 1$ and its own realization at this step $\Lambda_T(V_{\ell-1}^{p})$.

$$\pi_\ell^{(p)} = \begin{cases} \Lambda_T(V_{\ell-1}^{(p)}) & \textit{if } \cos t(\Lambda_T(V_{\ell-1}^{(p)})) \leqslant \cos t(\overline{\pi}_\ell^{(p-1)}) \\ \overline{\pi}_\ell^{(p-1)} & \textit{otherwise} \end{cases} \tag{16}$$

where $\pi_\ell^{(p)}(T)$ is Markov chain of co-thread $p\,(p = 1, 2, \ldots, W-1)$ at the annealing temperature T. $\Lambda_T(V)$ is a realization of one step of the chain at temperature T with a starting point of V.

The approach employed here builds upon the idea of Czech et al. (2010), and incorporates a master–slave structure into a modified parallel algorithm. Our integrated framework is shown in Fig. 4. The master-thread P_0 generates a high-quality solution as an initial solution from the RCRS algorithm (Dethloff, 2001), (Step 1), and then does not perform any computation in the p-SA procedure. Instead, it conducts a surveillance on the search process, and serves as a location for collecting the current LBS from thread $W-1$, (Step 7), and distributing the current GBS, (Step 2), to trigger the next round of parallel chains. The slave threads receive the current GBS, run local search independently and simultaneously (Step 3), and exchange information periodically about the best solutions found to date (Steps 4–6) in order to improve the current GBS (Step 7). The slave-thread P_1 receives an initial solution from P_0, (Step 2), and runs SA independently, (Step 3). The slave-thread P_2 updates the current LBS at steps $u \times \ell$ $(u = 1, 2, \ldots)$ based on the best solution s^1 received from P_1, (Step 4), and the best solution s_i^2 computed in the current last step by thread P_2, (Step 3). Similarly, slave-threads $P_3, P_4, \ldots, P_{W-2}$ compare solutions from the previous thread, (Step 5), with their own SA computation solutions, (Step 3), and updates their individual current LBS. The slave-thread P_{W-1} computes the current LBS, (Steps 3 and 6), and sends it to the master-thread, (Step 7). The master-thread refines the current GBS and uses it as a new initial solution for the next iteration process. These steps are then repeated until the stopping criterion is met, (Step 8), at which point the final solution is output, (Step 9).

The proposed p-SA for VRPSPDTW problem was built on the parallel simulated annealing algorithm proposed by Czech and Czarnas (2002) for the VRPTW problem. So, the required computation efforts for the two algorithms are the same. The worse case time complexity of the p-SA algorithm for the VRPSPDTW is $\Gamma_{W-1}(n) \leqslant O(n^3 + (W-1)n^2)$ (Czech & Czarnas, 2002). The p-SA

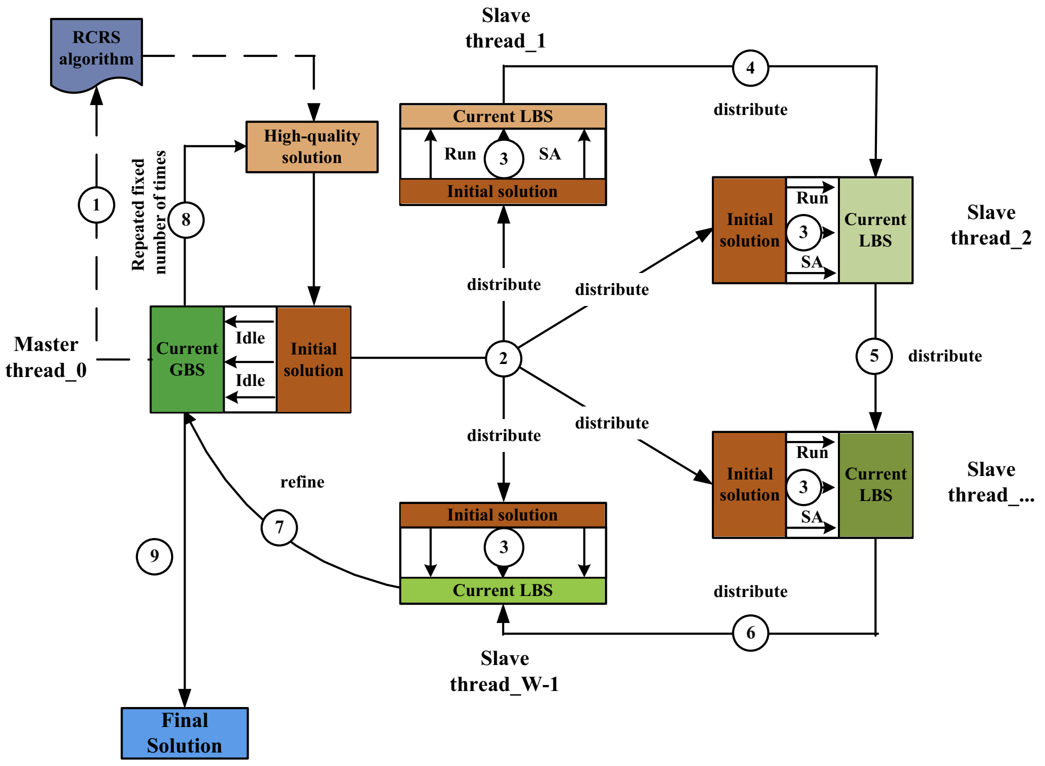

Fig. 4. Master–slave paradigm in parallel simulated annealing.

algorithm explores the solution space more extensively and speeds up the optimization process for solving the VRPSPDTW problem. It not only ensures that the algorithm has a large-scale parallel granularity, but also distributes the search results from the current slave-thread to the next slave-thread periodically, which achieves a fusion of the search information.

5. Evaluation and discussion

In this section, we demonstrate the effectiveness of our p-SA meta-heuristic by testing it using well known benchmark data sets and a new benchmark data set. Wang and Chen (2012) generated benchmark data sets for the VRPSPDTW problem, which are the only benchmark data sets in the literature. Wang and Chen's instances[1] include nine small-scale instances (three 10-customer instances, three 25-customer instances, and three 50-customer instances) and fifty-six medium-scale instances (fifty-six 100-customer instances). We have generated thirty large-scale instances (six 200-customer instances, six 400-customer instances, six 600-customer instances, six 800-customer instances, and six 1000-customer instances) derived from Gehring and Homberger's data sets (1999) for the VRPTW problem. So in total, ninety-five instances were tested using the proposed p-SA. The p-SA algorithm was developed in a JDK 7 environment and all experiments were executed on a 2.0 GHz desktop Intel Xeon CPU E5-2650 (2 processors) with 16 GB of RAM.

5.1. Configuration of parameters

Prior to conducting the experiments, several trial runs were undertaken to tune the algorithm parameters to ensure convergence and speed, and to obtain high-quality solutions in reasonable computation time. These parameters were W, σ, c_d, c_t, γ, β, Δ, L, $\omega_{\max}$, *lambda* and *alpha*. Parameter W denotes the number of threads, and is set based on the empirical formula $W = cores \times 2 + 2$ (Richter & Nasarre, 2011). Wang and Chen (2012) minimized a hierarchical objective function where the primary objective is to minimize NV and the secondary objective is to minimize the TD, so the same objective function was employed in this paper. Therefore, parameter σ is set to 1, and c_d is set to 1. Since in Eq. (1) c_t is ultimately multiplied by $1 - \sigma$, its value need not be specified. Parameter γ relates the cost and temperature, and as suggested by Czech and Czarnas (2002), γ is set to 1. The parameter β is the cooling ratio. The temperature should be decreased in such a way to avoid excessively long Markov chains, since we obtain one Markov chain node for each temperature value. The parameter Δ represents the probability of accepting poorer solutions. The function call, random(), returns a uniform random variable in the range $[0, 1]$. If the acceptance probability function $P(T_k, T, D) > random()$; then the solution is accepted, otherwise it is not accepted. Parameter L is the maximum number of iterations for temperature T. The parameter $\omega_{\max}$ is the maximum number of iterations for which a non-improved solution is obtained; this should be large enough to achieve a good solution. But, this must be balanced against the fact that a larger $\omega_{\max}$ value will increase the computation time, and the probability of getting a better solution is rather small at low temperatures. The parameters *lambda* and *alpha*, which are parameters in the RCRS insertion criterion proposed by Dethloff (2001), are not critical because the initial solution obtained by RCRS algorithm is improved by the parallel SA meta-heuristic.

Our experiments verify the expected behavior. Higher cooling ratio values correspond to slower cooling schedules and therefore more reduction steps are required for the algorithm to converge. The above parameter values were obtained through fine tuning and are listed in Table 1. In general, the performance of the p-SA algorithms is very sensitive to the cooling schedule.

[1] Dr. Chen provided the Wang and Chen benchmark instances: http://oz.nthu.edu.tw/~d933810/test.htm.

Table 1
Parameters setting summary.

p-SA parameter	W	σ	c_d	γ	β	L	$\omega_{\max}$	Δ	*lambda*	*alpha*
Value	66	1	1	1	0.96	n^2	40	12	0.2	1.0

5.2. Computational results

This subsection presents the computational results of the proposed p-SA for the VRPSPDTW problem. The p-SA is compared with the GA proposed by Wang and Chen (2012). Two criteria are adopted: (1) the number of vehicles (the primary goal) and (2) the travel distance (the secondary goal). Results are shown for both the Wang and Chen (2012) and our algorithms.

5.2.1. Results for the small-scale instances

Wang and Chen (2012) employed the commercial linear programming software CPLEX to find the optimal solutions for the small-scale instances of the Wang and Chen benchmark, which can be used to evaluate the accuracy of the proposed p-SA. Table 2 compares the best results obtained by CPLEX (Wang & Chen, 2012), the GA meta-heuristic (Wang & Chen, 2012), and the proposed p-SA. It is to be noted that the instances provided by Wang and Chen (2012) provide geographical positions for every customer and the number of products to be picked up and delivered for each customer. The last column of this table shows the gaps between GA and the proposed p-SA. The gap for NV is the differences between the NVs of the proposed p-SA and the GA (a positive value for the gap indicates that the proposed p-SA algorithm produces a larger NV value), and TD% gap is one hundred times the difference between the TDs of the proposed p-SA and GA divided by the TD for the GA. These results are also summarized in the Table 2. In examining the table, the following notation (adopted from Wang and Chen (2012)) is employed to denote an instance. "RCdp" refers to the fact that the customer locations are a mix of uniformly random (R) and clustered (C) positions. The four digit numbers that follows "RCdp" is the data set number, and the number that follows the slash denotes the number of customers.

Table 2 demonstrates that CPLEX is only able to solve five of the instances to optimality; for the remaining instances, CPLEX reports an "out of memory" condition. For the 5 solved instances, the proposed p-SA finds or gets very close to optimal solution in a much shorter time (1–16 s). For the other four instances, the solutions from CPLEX are poor, and the proposed p-SA's solutions are much better.

When the GA meta-heuristic and the p-SA are compared, the NV obtained by p-SA is the same as the NV obtained by the GA meta-heuristic. However, the TD obtained by p-SA is slightly poorer than the TD obtained by the GA meta-heuristic. The average and maximum percentage gaps are 0.04% and 0.21% respectively.

Table 3
Description of instances adopted from Wang and Chen.

	Description
RdpX0Y	Customer locations are randomly (uniformly) located
CdpX0Y	Customer locations are clustered
RCdpX0Y	Customer locations are a mix of random and clustered positions
Mdp10Y	Instances with narrow time windows and small vehicle capacity
Mdp20Y	Instances with large time windows and large vehicle capacity
MdpX01, MdpX02, MdpX03...	Instances have the same customer locations, but the demand amounts, pickup amounts, and/or time windows may vary from the same customer

5.2.2. Results for the medium-scale instances

For the medium-scale instances of the Wang and Chen benchmark, each instance is coded as MdpX0Y. The notation used for each instance is adopted from Wang and Chen (2012) and is defined in Table 3.

Table 4 compares the best results obtained by Wang and Chen (2012) (GA meta-heuristic) with the results obtained by the proposed p-SA for medium-scale instances of the Wang and Chen benchmark. The last column of this table shows the gaps between the proposed p-SA and the GA solutions.

Based on the results presented in Table 4, the differences between the two algorithms may be summarized as follows:

- For all 56 medium-scale instances, the p-SA algorithm obtained the same or fewer NVs (which is the primary objective) as compared with the GA. More precisely, in 12 instances (21.4%), the p-SA algorithm obtained solutions that had one vehicle less than the GA. For the remaining 44 instances (78.6%) the NV was the same.
- For the 44 instances with same NV, there were 16 instances where the TD (the secondary objective) obtained by the p-SA was better than the TD obtained by the GA; the average percentage improvement for these 16 instances was 1.31%. There were 7 instances where the TD obtained by the p-SA matched the TD obtained by the GA. For the balance of the instances (21 of the 44 instances with the same NV), the p-SA provided poorer solutions than the GA. This may suggest that the GA algorithm manages a secondary objective better than the proposed p-SA method.

Table 2
Comparison between CPLEX, GA and the proposed p-SA of small-scale instances to the VRPSPDTW.

Problem/instance	CPLEX			GA			Proposed p-SA			Gap	
	NV	TD	T	NV	TD	T^*	NV	TD	T^{*a}	NV	TD%
RCdp1001/10	3	348.98	1	3	348.98	1	3	348.98	1	0	0.00
RCdp1004/10	2	216.69	1503	2	216.69	1	2	216.69	1	0	0.00
RCdp1007/10	2	310.81	25	2	310.81	1	2	310.81	1	0	0.00
RCdp2501/25	5	551.05	16	5	551.05	3	5	552.21	2	0	0.21
RCdp2504/25	7[a]	738.32[a]	485,660[a]	4	473.46	2	4	473.46	1	0	0.00
RCdp2507/25	7[a]	634.20[a]	439,321[a]	5	540.87	3	5	540.87	2	0	0.00
RCdp5001/50	9	994.18	327,404	9	994.18	18	9	994.70	14	0	0.05
RCdp5004/50	14[a]	1961.53[a]	839,320[a]	6	725.59	23	6	725.90	16	0	0.04
RCdp5007/50	13[a]	1814.33[a]	1,546,429[a]	7	809.72	22	7	810.04	15	0	0.04
								Minimum			0.00
								Mean			0.04
								Maximum			0.21

T^*: **CPU times in seconds** executed on an Intel Core 2 Quad 2.4 GHz with 1G memory.
[a] The "out of memory" values.

Table 4
Comparison between GA and the proposed p-SA of medium-scale instances to the VRPSPDTW.

Problem	GA		Proposed p-SA			Gap		Problem	GA		Proposed p-SA			Gap	
	NV	TD	NV	TD	T	NV	TD%		NV	TD	NV	TD	T	NV	TD%
Rdp101	19	1653.53	19	1660.98	43	0	0.45	Rdp201	4	1280.44	4	1286.55	84	0	0.48
Rdp102	17	1488.04	17	1491.75	29	0	0.25	Rdp202	4	1100.92	4	1150.31	123	0	4.49
Rdp103	14	1216.16	14	1226.77	41	0	0.87	Rdp203	3	950.79	3	997.84	102	0	4.95
Rdp104	10	1015.41	10	1000.65	45	0	**−1.45**	Rdp204	3	775.23	2	848.01	120	**−1**	9.39
Rdp105	15	1375.31	14	1399.81	45	**−1**	1.78	Rdp205	3	1064.43	3	1046.06	116	0	**−1.73**
Rdp106	13	1255.48	12	1275.69	37	**−1**	1.61	Rdp206	3	961.32	3	959.94	134	0	**−0.14**
Rdp107	11	1087.95	11	1082.92	35	0	**−0.46**	Rdp207	3	835.01	2	899.82	85	**−1**	7.76
Rdp108	10	967.49	10	962.48	41	0	**−0.52**	Rdp208	3	718.51	2	739.06	127	**−1**	2.86
Rdp109	12	1160.00	12	1181.92	46	0	1.89	Rdp209	3	930.26	3	947.80	111	0	1.89
Rdp110	12	1116.99	11	1106.52	45	**−1**	**−0.94**	Rdp210	3	983.75	3	1005.11	164	0	2.17
Rdp111	11	1065.27	11	1073.62	41	0	0.78	Rdp211	3	839.61	3	812.44	98	0	**−3.24**
Rdp112	10	974.03	10	966.06	51	0	**−0.82**								
Cdp101	11	1001.97	11	992.88	36	0	**−0.91**	Cdp201	3	591.56	3	591.56	86	0	**0.00**
Cdp102	10	961.38	10	955.31	38	0	**−0.63**	Cdp202	3	591.56	3	591.56	91	0	**0.00**
Cdp103	10	897.65	10	958.66	34	0	6.80	Cdp203	3	591.17	3	591.17	88	0	**0.00**
Cdp104	10	878.93	10	944.73	35	0	7.49	Cdp204	3	590.60	3	594.07	90	0	0.59
Cdp105	11	983.10	11	989.86	37	0	0.69	Cdp205	3	588.88	3	588.88	90	0	**0.00**
Cdp106	11	878.29	11	878.29	37	0	**0.00**	Cdp206	3	588.49	3	588.49	88	0	**0.00**
Cdp107	11	913.81	11	911.90	41	0	**−0.21**	Cdp207	3	588.29	3	588.29	85	0	**0.00**
Cdp108	10	951.24	10	1063.73	39	0	11.83	Cdp208	3	588.32	3	599.32	83	0	1.87
Cdp109	10	940.49	10	947.90	21	0	0.79								
RCdp101	15	1652.90	15	1659.59	47	0	0.40	RCdp201	4	1587.92	4	1513.72	64	0	**−4.67**
RCdp102	14	1497.05	13	1522.76	41	**−1**	1.72	RCdp202	4	1211.12	4	1273.26	72	0	5.13
RCdp103	12	1338.76	11	1344.62	45	**−1**	0.44	RCdp203	4	964.65	3	1123.58	78	**−1**	16.48
RCdp104	11	1188.49	10	1268.43	47	**−1**	6.73	RCdp204	3	822.02	3	897.14	80	0	9.14
RCdp105	14	1581.26	14	1581.54	46	0	0.02	RCdp205	4	1410.18	4	1371.08	62	0	**−2.77**
RCdp106	13	1422.87	13	1418.16	41	0	**−0.33**	RCdp206	3	1176.85	3	1166.88	66	0	**−0.85**
RCdp107	12	1282.10	11	1360.17	35	**−1**	6.09	RCdp207	4	1036.59	3	1089.85	75	**−1**	5.14
RCdp108	11	1175.04	11	1169.57	38	0	**−0.47**	RCdp208	3	878.57	3	862.89	73	0	**−1.78**

Boldface indicates that those solutions of the proposed p-SA have less NV or lower TD than GA.

Table 5
Categories for large instances (adopted from Gehring and Homberger (1999)).

Customers	Type C1	Type C2	Type R1	Type R2	Type RC1	Type RC2
200	S-C1-200	S-C2-200	S-R1-200	S-R2-200	S-RC1-200	S-RC2-200
400	S-C1-400	S-C2-400	S-R1-400	S-R2-400	S-RC1-400	S-RC2-400
600	S-C1-600	S-C2-600	S-R1-600	S-R2600	S-RC1-600	S-RC2-600
800	S-C1-800	S-C2-800	S-R1-800	S-R2-800	S-RC1-800	S-RC2-800
1000	S-C1-1000	S-C2-1000	S-R1-1000	S-R2-1000	S-RC1-1000	S-RC2-1000

- Data from Wang and Chen (2012) was not available to compare with the proposed p-SA method in terms of the time required to obtain a solution for each instance.

5.2.3. Results for the large-scale instances

Gehring and Homberger (1999) extended Solomon's VRPTW benchmark, and generated instances for 200, 400, 600, 800, and 1000 customers. As with the other benchmarks, a labeling scheme is needed to refer to specific instances; Table 5 details the labeling scheme. In the table, instances are divided into three categories: C-type (clustered customers), R-type (uniformly distributed customers), and RC-type (a mix of R and C types). Type 1 means instances with narrow time windows and small vehicle capacity

Table 6
Results obtained by the proposed p-SA of large-scale instances to the VRPSPDTW.

Customer	Problem	Proposed p-SA			Problem	Proposed p-SA		
		NV	TD	T		NV	TD	T
200	C1_2_1	3169.52	21	62	C2_2_1	1972.97	6	112
	R1_2_1	5083.39	22	89	R2_2_1	4372.17	5	295
	RC1_2_1	3865.18	20	81	RC2_2_1	2662.75	4	216
400	C1_4_1	8135.35	42	147	C2_4_1	5085.08	14	279
	R1_4_1	12202.62	42	237	R2_4_1	14119.64	9	735
	RC1_4_1	10036.82	40	193	RC2_4_1	7229.22	13	791
600	C1_6_1	19720.65	69	257	C2_6_1	9509.15	20	926
	R1_6_1	25729.28	62	581	R2_6_1	27294.11	13	2439
	RC1_6_1	20535.26	60	733	RC2_6_1	22837.36	20	2860
800	C1_8_1	32801.92	88	1054	C2_8_1	14573.93	27	3636
	R1_8_1	51949.49	93	1869	R2_8_1	48611.6	19	7663
	RC1_8_1	32801.92	88	1620	RC2_8_1	39375.78	22	6026
1000	C1_10_1	52328.78	110	2418	C2_10_1	23981.11	33	6529
	R1_10_1	77993.35	115	4539	R2_10_1	67441.51	22	21,379
	RC1_10_1	66883.49	102	3483	RC2_10_1	61473.68	29	13,793

and type 2 means instances with large time windows and large vehicle capacity.

We selected the first instance from the 200, 400, 600, 800, and 1000 customers instances as our large test problem which are totally 30 instances. We added pickup amount P_j to each customer. The following procedure is employed: The pickup amount P_j corresponding to the delivery amount D_j is computed by using a random number r_j that is uniformly distributed over the interval $[0, 1]$ such that $P_j = (r_j + 0.5)D_j$.

Table 6 presents the best results obtained by the proposed p-SA for large-scale instances. It was demonstrated that type 2 is more complicated to solve by the p-SA algorithm than type 1. Type 2 requires on average needs 3.4 times more computational time than type 1 to obtain the best solution. When the number of customer is under 400, the computational time of the p-SA algorithm increases nearly linearly. As the number of customers increases up to 1000, the computational time grows factorially. For the large scale instances, no other results are available to compare with the p-SA algorithm.

6. Summary and conclusions

The vehicle routing problem with simultaneous pickup–delivery and time windows is a complex, constrained NP-hard problem with widespread business applications; the problem has both economic and environmental implications. The complexity of real-life medium and large-scale instances prohibits exact solutions to the problem, making it necessary to implement heuristic or meta-heuristic approaches to provide approximate solutions within reasonable computational time. This paper has proposed a p-SA meta-heuristic to solve the VRPSPDTW problem. To the best of our knowledge, this is the first implementation of p-SA approach to solve this problem.

In order to demonstrate the effectiveness of the p-SA algorithm, an experimental study has been carried out using 65 test problems from Wang and Chen's benchmark and compared to a GA meta-heuristic. Experimental results show the effectiveness of the proposed algorithm for solving the VRPSPDTW problem, which obtains the same number of vehicles, NV (the primary objective) as all 9 (100%) of the Wang and Chen small-scale benchmarks. For the 56 medium-scale instances, the p-SA algorithm obtains the same (44 instances, 78.6%) or better (12 instances, 21.4%) NV solution as compared with the GA method. For the 44 instances with the same NV, there are 16 instances where the TD obtained by the p-SA is better than the TD obtained by the GA, and there are 7 instances where the TD obtained by the p-SA equals the TD obtained by the GA. A set of 30 new instances with 200, 400, 600, 800 and 1000 customers were generated and utilized as a new data set for the large-scale VRPSPDTW. The p-SA algorithm, with parameters selected through fine tuning, has been shown to be very robust, returning either better solutions (or equal to the best solutions) or solutions very close in quality to all Wang and Chen benchmarks. The results demonstrate the benefits of using the p-SA algorithm to solve the VRPSPDTW problem.

The proposed model points to a number of avenues for future work. First, although the proposed p-SA algorithm obtains the same or fewer NVs as compared with the GA, the average TD (the secondary objective) is poorer in some cases, although no comparisons are available for large-scale instances. Further research may examine the integration of local search techniques into the algorithm to decrease the TD objective. Second, this paper has assumed a constant travel speed or no travel time variability. It is well known that congestion is a very common phenomenon in urban areas and congestion creates a substantial variation in travel speeds during peak morning and evening hours (Figliozzi, 2012). This model could be extended to consider traffic congestion, which is a time-dependent VRPSPDTW problem. Third, the cost of carbon emissions may be incorporated into the objective function and a simulated annealing-based parallel multi-objective approach may be utilized to simultaneously minimize the transportation cost and environmental impacts of carbon emissions.

Acknowledgements

This work has been supported by the State Key Program of National Natural Science of China (No. 71132008), The General Program of National Natural Science of China (No. 71473013), China Scholarship Council (201207090034), and the Fundamental Research Funds for the Central Universities of China (2012YJS034). A portion of the computational resources for this work were provided by the Engineering Computer Network (ECN) at Purdue, West Lafayette, Indiana. The authors wish to express our thanks for Dr. Shengchuan Zhou from Qingdao Geotechnical Investigation and Surveying Research Institute, and Mr. Weiren Kong from College of Science, Northwest Polytechnical University, who have provided the programming help.

References

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

Angelelli, E., & Mansini, R. (2003). The vehicle routing problem with time windows and simultaneous pick-up and delivery. In A. Klose, M. G. Speranza, & L. N. V. Wassenhove (Eds.), *Quantitative approaches to distribution logistics and supply chain management* (pp. 249–267). Berlin: Springer.

Baldacci, R., Mingozzi, A., & Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research, 218*(1), 1–6.

Baños, R., Ortega, J., Gil, C., Fernández, A., & de Toro, F. (2013). A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications, 40*(5), 1696–1707.

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

Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research, 202*(1), 8–15.

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

Boubahri, L., Addouche, S.A. & El Mhamedi, A. (2011). Multi-ant colonies algorithms for the VRPSPDTW. In *CCCA: 2011 international conference on communications, computing and control applications* (pp. 1–6), March 3–5, 2011, Hammamet.

Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science, 39*(1), 104–118.

Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science, 39*(1), 119–139.

Çatay, 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.

Chandy, J. A., Kim, S., Ramkumar, B., Parkes, S., & Banerjee, P. (1997). An evaluation of parallel simulated annealing strategies with application to standard cell placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 16*(4), 398–410.

Chen & Wu (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society, 57*(5), 579–587.

Chiang, W. C., & Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research, 63*(1), 3–27.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., & Soumis, F. (2002). VRP with time windows. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem SIAM monographs on discrete mathematics and applications* (pp. 157–193). Philadelphia: Society for Industrial and Applied Mathematics.

Crainic, T. G. (2008). Parallel solution methods for vehicle routing problems. In B. Golden, S. Raghavan, & W. Wasil (Eds.), *The vehicle routing problem: Latest advances and new challenges* (pp. 171–198). Berlin: Springer.

Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR), 45*(3). Article 35.

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*(11), 1296–1302.

Czech, Z. J., & Czarnas, P. (2002). Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings of 10th Euromicro workshop on parallel, distributed and network-based processing* (pp. 376–383), Canary Islands, Spain.

Czech, Z., Mikanik, W., & Skinderowicz, R. (2010). Implementing a parallel simulated annealing algorithm. In R. Wyrzykowski, J. Dongarra, K. Karczewski, & J. Wasniewski (Eds.), *PPAM: International conference on parallel processing and applied mathematics, September 13–16, 2009* (pp. 146–155). Berlin: Springer.

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

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

Diana, M., & Dessouky, M. M. (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological, 38*(6), 539–557.

Dowsland, K. A. (1993). Simulated annealing. In C. R. Reeves (Ed.), *Modern heuristic techniques for combinatorial problems* (pp. 20–69). Oxford: Blackwell Scientific Publications.

El-Sherbeny, N. A. (2010). Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University – Science, 22*(3), 123–131.

Ferreiro, A., García, J., López-Salas, J., & Vázquez, C. (2012). An efficient implementation of parallel simulated annealing algorithm in GPUs. *Journal of Global Optimization, 57*(3), 863–890.

Figliozzi, M. A. (2012). The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review, 48*(3), 616–636.

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

Gan, X., Wang, Y., Li, S., & Niu, B. (2012). Vehicle routing problem with time windows and simultaneous delivery and pick-up service based on MCPSO. *Mathematical Problems in Engineering, 2012*, 11.

Gehring, H., & Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In K. Miettinen, M. Makela, & J. Toivanen (Eds.). *Proceedings of EUROGEN99* (Vol. 2, pp. 57–64). Berlin: Springer.

Gendreau, M., & Tarantilis, C.D. (2010). *Solving large-scale vehicle routing problems with time windows: The state-of-the-art*. Technical Report 2010-04, CIRRELR, University of Montreal.

Goetschalckx, M., & Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research, 42*(1), 39–51.

Hasle, G., & Kloster, O. (2007). Industrial vehicle routing. In G. Hasle, K.-A. Lie, & E. Quak (Eds.), *Geometric modelling, numerical simulation, and optimization* (pp. 397–435). Berlin: Springer.

Jin, J., Crainic, T. G., & Løkketangen, A. (2012). A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research, 222*(3), 441–451.

Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research, 35*(7), 2307–2330.

Kallehauge, B., Larsen, J., Madsen, O. B., & Solomon, M. M. (2005). Vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), *Column generation* (pp. 67–98). New York: Springer.

Kirkpatrick, S., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

Kuo, Y. (2010). Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering, 59*(1), 157–165.

Lai, M. Y., & Cao, E. B. (2010). An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence, 23*(2), 188–195.

Lee, K.-G., & Lee, S.-Y. (1992). Efficient parallelization of simulated annealing using multiple Markov chains: An application to graph partitioning. In T. N. Mudge (Ed.), *Proceedings of the international conference on parallel processing, III: Algorithms and applications* (pp. 177–180). CRC press.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal, 44*(10), 2245–2269.

Lin, C., Choy, K. L., Ho, G. T. S., Chung, S. H., & Lam, H. Y. (2014). Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications Part, 41*(4), 1118–1138.

Liu, R., Xie, X., Augusto, V., & Rodriguez, C. (2013). Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research, 230*(3), 475–486.

Lu, Q., & Dessouky, M. (2004). An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science, 38*(4), 503–514.

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.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*, 1087–1092.

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

Moccia, L., Cordeau, J.-F., & Laporte, G. (2011). An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society, 63*(2), 232–244.

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

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 37*(4), 724–737.

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.

Nikolaev, A. G., & Jacobson, S. H. (2010). Simulated annealing. In M. Gendreau & J. Y. Potvin (Eds.), *Handbook of metaheuristics*. *International series in operations research & management science* (Vol. 146, pp. 1–39). New York: Springer.

Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Ph.D. thesis, USA: Northwestern University.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research, 41*(4), 421–451.

Phannikul, T., & Sindhuchao, S. (2010). A customized tabu search for vehicle routing problem with simultaneous pickup and delivery. *International Journal of Science and Technology, 15*(2), 70–82.

Potvin, J.-Y., & Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society, 46*(12), 1433–1446.

Richter, J., & Nasarre, C. (2011). *Windows via C/C++* (5th ed.). Washington: Microsoft Press.

Rieck, J., & Zimmermann, J. (2009). A Branch-and-cut approach to the vehicle routing problem with simultaneous delivery and pick-up. In B. Fleischmann, K.-H. Borgwardt, R. Klein, & A. Tuma (Eds.), *Operations research proceedings 2008* (pp. 301–306). Berlin: Springer.

Salhi, S., & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society, 50*(10), 1034–1042.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*(2), 254–265.

Souza, M. J. F., Mine, M. T., Silva, M. d. S. A., Ochi, L. S., & Subramanian, A. (2011). A hybrid heuristic, based on iterated local search and GENIUS, for the vehicle routing problem with simultaneous pickup and delivery. *International Journal of Logistics Systems and Management, 10*(2), 142–157.

Subramanian, A., & Cabral, L. D. A. F. (2008). An ILS based heuristic for the vehicle routing problem with simultaneous pickup and delivery and time limit. In J. V. Hemert & C. Cotta (Eds.), *Evolutionary computation in combinatorial optimization* (pp. 135–146). Berlin: Springer.

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

Subramanian, A., Cabral, L.D.A.F. & Carvalho, G.R. (2007). A hybrid metaheuristic for the vehicle routing problem with simultaneous pick-up and delivery. In *ICIEOM: XIII international conference on industrial engineering and operations management, energy that moves production: A dialogue among integration, project and sustainability*. October 09–11, 2007, Iguassu Falls, PR, Brazil.

Subramanian, A., Cabral, L. & Ochi, L. (2008). *An efficient ILS heuristic for the vehicle routing problem with simultaneous pickup and delivery*. Technical Report, Universidade Federal Fluminense. < http://www2.ic.uff.br/PosGraduacao/RelTecnicos/401.pdf >.

Subramanian, A., Uchoa, E., & Ochi, L. S. (2013a). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research, 40*(10), 2519–2531.

Subramanian, A., Uchoa, E., & Ochi, L. S. (2010b). New lower bounds for the vehicle routing problem with simultaneous pickup and delivery. In P. Festa (Ed.), *Experimental algorithms* (pp. 276–287). Berlin: Springer.

Subramanian, A., Uchoa, E., Pessoa, A. A., & Ochi, L. S. (2011). Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. *Operations Research Letters, 39*(5), 338–341.

Subramanian, A., Uchoa, E., Pessoa, A., & Ochi, L. (2013b). Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters, 7*(7), 1569–1581.

Suman, B., & Kumar, P. (2005). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society, 57*(10), 1143–1160.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science, 31*(2), 170–186.

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*(3), 755–761.

Tavakkoli-Moghaddam, R., Gazanfari, M., Alinaghian, M., Salamatbakhsh, A., & Norouzi, N. (2011). A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing. *Journal of Manufacturing Systems, 30*(2), 83–92.

Toth, P., & Vigo, D. (2002). VRP with backhauls. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem, SIAM monographs on discrete mathematics and applications* (pp. 195–221). Philadelphia: Society for Industrial and Applied Mathematics.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research, 40*(1), 475–489.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research, 234*(3), 658–673.

Wang, H.-F., & Chen, Y.-Y. (2012). A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering, 62*(1), 84–95.

Wang, H.-F., & Chen, Y.-Y. (2013). A coevolutionary algorithm for the flexible delivery and pickup problem with time windows. *International Journal of Production Economics, 141*(1), 4–13.

Wang, H.-F., & Hsu, H.-W. (2010). A closed-loop logistic model with a spanning-tree based genetic algorithm. *Computers & Operations Research, 37*(2), 376–389.

Wassan, N. A., Wassan, A. H., & Nagy, G. (2008). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization, 15*(4), 368–386.

Wei, R., Zhang, T., & Tang, H. (2011). An improved particle swarm optimization algorithm for vehicle routing problem with simultaneous pickup and delivery. In R. B. Zhu, Y. C. Zhang, B. X. Liu, & C. F. Liu (Eds.), *Information Computing and Applications* (pp. 430–436). Berlin: Springer.

Zachariadis, E. E., & Kiranoudis, C. T. (2011). A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Systems with Applications, 38*(3), 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), 1070–1081.