A Local Search Metaheuristic Algorithm for the Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries

Emmanouil E. Zachariadis, Chris T. Kiranoudis

Department of Process Analysis and Plant Design, National Technical University of Athens, Athens, Greece, {ezach@mail.ntua.gr, kyr@chemeng.ntua.gr}

Abstract

This article proposes a local search metaheuristic solution approach for the Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries (VRPSPD), which models numerous practical transportation operations in the context of reverse logistics. The proposed algorithm is capable of exploring wide solution neighborhoods by statically encoding moves into special data structures. To avoid cycling and induce diversification, the overall search is coordinated by the use of the promises concept which is applied to solution arcs. In terms of the challenging capacity constraints imposed by the VRPSPD model, we present a constant-time feasibility checking procedure for the employed local search operators. The presented metaheuristic development was tested on eighteen large-scale VRPSPD benchmark instances derived from the literature. It proved to be both robust and effective, improving most of the previously best-known solutions of the examined test problems.

keywords: *vehicle routing, simultaneous pick-ups and deliveries, metaheuristics, computational complexity*

---

1. Introduction

In the present paper, we examine a practical Vehicle Routing Problem (VRP) variant which considers customers to simultaneously require both delivery and pick-up services. The examined transportation problem is referred to as the Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries (VRPSPD), and has attracted research interest because it models a wide variety of business operations involving bi-directional flow of goods.

In graph theory terms, VRPSPD is defined on a complete graph $G = (V, A)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is the vertex set, and A is the arc set. Vertex v_0 corresponds to the depot which acts as the central station for a fleet of homogeneous vehicles. Each of these vehicles has a maximum carrying load equal to Q. Vertices of the set $V - \{v_0\} = \{v_1, v_2, \ldots, v_n\}$ represent the customer population. With each customer vertex are associated two fixed, product demands: the pick-up demand p_i, and the delivery demand d_i, whereas with each arc $(v_i, v_j) \in A$ is associated a non-negative cost c_{ij} reflecting the time, the distance or the actual monetary cost required for traveling from v_i to v_j. The VRPSPD aims at generating the minimum cost set of Hamiltonian circuits (*routes*) so that customer delivery and pick-up demand is totally satisfied. The produced set of routes must respect the following requirements: (a) each route originates from, and terminates to the central depot v_0, (b) each customer v_i ($i = 1,\ldots, n$) is visited once by exactly one route, (c) at no point of any route, the transported quantity of goods exceeds the maximum carrying load Q of the vehicles.

The VRPSPD model has been in the focus of research interest mainly because of its commercial importance, and theoretical challenge. In terms of the operational importance, VRPSPD finds applicability in numerous reverse logistics systems: retailers are able to negotiate the return excess unsold products back to the manufacturers with beneficial effects for both parts. Furthermore, extended responsibilities have been assigned to producers regarding the entire life-cycle of their products. Used products like industrial equipment, hardware devices etc. are also sent back to the manufacturer facilities to be disassembled into valuable components. Another example of reverse product flows is due to recent pro-environmental legislation which forces companies to collect various used products such as lubricants, batteries, tires, fluorescent lights, etc. in order to be appropriately processed. From the theoretical perspective, VRPSPD generalizes the standard Capacitated VRP (CVRP) version. In specific, any CVRP instance can be interpreted as a VRPSPD instance for which either all delivery, or all pick-up demands, respectively, are equal to zero. As a CVRP generalization, VRPSPD is a NP-hard combinatorial optimization problem. Thus, to efficiently deal with large-scale VRPSPD instances which arise in practical operations, one's interest has to be focused in

heuristic and metaheuristic approaches, which can produce high quality solutions within limited computational times.

The purpose of this paper is to propose a new metaheuristic methodology for the VRPSPD model. The proposed method is a local-search based approach which examines rich solution neighborhoods. To examine these rich neighborhood structures efficiently, we make use of the Static Move Descriptor entities which statically encode tentative local search moves (Zachariadis and Kiranoudis, 2009a). To effectively explore the solution space, the search is controlled by using the promise concept initially introduced for the VRP with Backhauls (Zachariadis and Kiranoudis, 2009b). In the present study, instead of complete routes, we consider arcs to be the solution attributes examined by the promises strategy. Moreover, we present an efficient scheme for examining the feasibility of the applied local search operators in terms of the challenging VRPSPD capacity constraints. The proposed metaheuristic was tested on well-known VRPSPD benchmark instances derived from the literature. It produced satisfactory results and managed to improve several best-known solutions.

The remainder of the present paper is organized as follows: Section 2 provides a literature review on the VRPSPD, followed by Section 3 which presents the proposed solution approach. In Section 4, the computational results are provided, followed by some concluding remarks in Section 5.

2. Literature Review

Reverse logistics are increasingly important, as numerous practical distribution operations require that goods are bi-directionally transported. In this context, several routing problem variants which consider customers to require both delivery and pick-up services have been examined. Three of these widely-studied routing models (Toth and Vigo 2002; Berbeglia et al. 2007) are the VRP with Backhauls (VRPB), the VRP with mixed pick-ups and deliveries (VRPMPD), and the VRP with simultaneous pick-ups and deliveries (VRPSPD).

The VRPB divides customers into two groups, namely the linehauls and backhauls. Vehicles are assumed to originate from the depot to satisfy the delivery demand of linehauls. After the last linehaul of each route is serviced, vehicles proceed to visit the

backhaul customers to collect their pick-up demand, before terminating their trips at the central depot. Under the VRPB model, the vehicle load monotonically decreases in the linehaul phase (until it is exhausted after the last linehaul is serviced), and then monotonically increases, as goods are collected from backhaul locations. This precedence constraint which forces linehauls before backhauls was imposed in the VRPB model because, as Goetschalckx and Jacobs-Blecha (1989) state, "*the vehicles are rear-loaded and rearrangement of the loads on the trucks at the delivery points is not deemed feasible*".

On the contrary, VRPMPD drops the aforementioned precedence constraint by letting linehauls and backhauls occur in any order during the vehicle trips. This causes the vehicle load to fluctuate along its trip, so that capacity constraints are harder to tackle.

The VRPSPD generalizes the VRPMPD model. It considers customers to simultaneously require both delivery and pick-up services, so that the customers are not divided into separate linehaul and backhaul groups. Every, VRPMPD instance can be seen as a VRPSPD one, for which either the delivery d_i or pick-up p_i demand of every customer v_i is equal to zero. In the following, we provide a detailed literature review on solution methodologies

The first work which dealt with a practical VRPSPD application is due to Min (1989). It involved 22 customers and 2 vehicles. The solution was obtained by clustering customers into disjoint sets, and then for each set, solving the Traveling Salesman Problem (TSP). The capacity constraints are satisfied by penalizing infeasible solution arcs. Dethloff (2001) has proposed and compared a series of construction heuristics which employ several insertion criteria. The algorithm of Nagy and Salhi (2005) first solves the corresponding VRP by handling both linehauls and backhauls in an integrated manner. Then, to eliminate capacity infeasibilities, the authors apply some VRP heuristic routines which are modified to tackle the VRPSPD. More recently, several Tabu Search (TS) frameworks have been proposed for the VRPSPD: Crispim and Brandão (2005) have implemented a hybridization of TS and Variable Neighborhood Descent (VND), while the article of Chen and Wu (2006) propose a hybrid scheme incorporating the TS and the record-to-record travel strategies. A pure TS implementation was proposed by Tang-Montanè and Galvão (2006). To induce diversification, the authors make use of an arc

frequency penalization scheme. Wassan et al. (2007) have designed a TS method which reacts to repetitions in order to guide the conducted search. Biancessi and Righini (2007) evaluate and compare the performance of several constructive heuristics, local search methods and TS implementations for the VRPSPD. Another TS-based algorithm has been proposed by Zachariadis et al. (2009a). In specific, the proposed approach explores the solution space by hybridizing the TS and Guided Local Search (GLS) strategies. The most recent VRPSPD metaheuristic methodologies have been published by Gajpal and Abad (2009), Ai and Kachitvichyanukul (2009), and Zachariadis et al. (2009b). The former article (Gajpal and Abad, 2009) presents an Ant Colony Optimization approach, whereas the second study (Ai and Kachitvichyanukul, 2009) proposes a Particle Swarm Optimization VRPSPD solution approach. Finally, the methodology of Zachariadis et al. (2009b) belongs to the Adaptive Memory Programming approaches. In specific, routes included in high-quality VRPSPD solutions are stored in the Adaptive Memory from which customer sequences are periodically extracted to form new initial solutions for guiding the search. The risk of an overall elitistic behavior is eliminated by the use of an additional memory component which drives the algorithm to exploit diverse routing information stored in the Adaptive Memory.

3. The Proposed Metaheuristic

The proposed VRPSPD metaheuristic is a local search algorithm which makes use of two algorithmic concepts, namely the Static Move Descriptor (SMD) strategy for efficiently exploring solution neighborhoods, and the promises concept for avoiding search cycling and inducing diversification. The selection of these algorithmic ingredients was motivated by their successful application to both the Open VRP (Zachariadis and Kiranoudis 2009a) and VRPB (Zachariadis and Kiranoudis 2009b) models. The SMD strategy, which reduces the computational complexity required for exploring solution neighborhoods, was used for both the OVRP and VRPB, whereas the promises concept was firstly introduced for the VRPB (Zachariadis and Kiranoudis, 2009b). As will be later explained, in this research we aimed at examining the behavior of promises for more basic solution attributes. In specific, instead of using complete routes as the solution attributes considered by the promises concept, we used simple solution arcs. The

aforementioned selection lead to a much effective VRPSPD solution approach, as it managed to produce high-quality solutions for several large-scale test problems.

In the present Section, we firstly present the neighborhood structures examined and their SMD representation, followed by the description of the proposed promises implementation. The overall metaheuristic structure is then provided, together with the way in which feasibility investigation for the employed local search operators is performed in constant time.

3.1. The Applied Local Search Operators

As in the case of our previous studies for the OVRP and the VRPB models, we examine two solution neighborhood structures. The first one is defined by every tentative move exchanging the positions of customer sequences (thereafter to be referred to as *bones*) which may contain up to μ customers, and is denoted by Variable Length Bone Exchange operator (VLBE). The second neighborhood considered is the well known 2-opt operator defined by every possible replacement of two solution arcs. The aforementioned neighborhood structures (VLBE and 2-opt) and their SMD representation have been described in detail for the OVRP (Zachariadis and Kiranoudis 2009a). For the completeness of the present paper, we provide a brief presentation of the applied local search operators.

3.1.1. The VLBE operator and its SMD representation

The VLBE operator exchanges the positions of any pair of bones each of them containing from 0 to μ customers. Obviously, the cardinality of the VLBE neighborhood is $O(\mu^2 n^2)$, as in total there are $O(n^2)$ 2-combinations of vertices, and for each such combination, there are $O(\mu^2)$ tentative moves, corresponding to the 2-combinations of the lengths of the exchanged bones.

To encode the VLBE operator using the SMD strategy, we use the VLBE SMD instances each of them corresponding to a particular VLBE tentative move. With each VLBE SMD instance are associated two node values n_1 and n_2, and two bone lengths *n1_len* and *n2_len*. The move encoded by a VLBE SMD instance with n_1 = A, n_2= B, *n1_len* = *a*, and *n2_len* = *b*, involves exchanging the positions of the bones originating after nodes A

and B, and containing a and b customer vertices, respectively, as seen in Figure 1. To exhaustively map the VLBE neighborhood for a problem of n customers and K vehicles, $((n+K)!/(2!(n+K-2)!))\cdot((\mu+1)^2-1)$ VLBE SMD instances are required in total. The first term corresponds to the 2-combinations without repetition of the n customers and K depot vertex occurrences, while the second term corresponds to the 2-combinations with repetition of the two bone lengths which vary from 0 to μ (no SMD instance is generated for both bone lengths equal to 0).

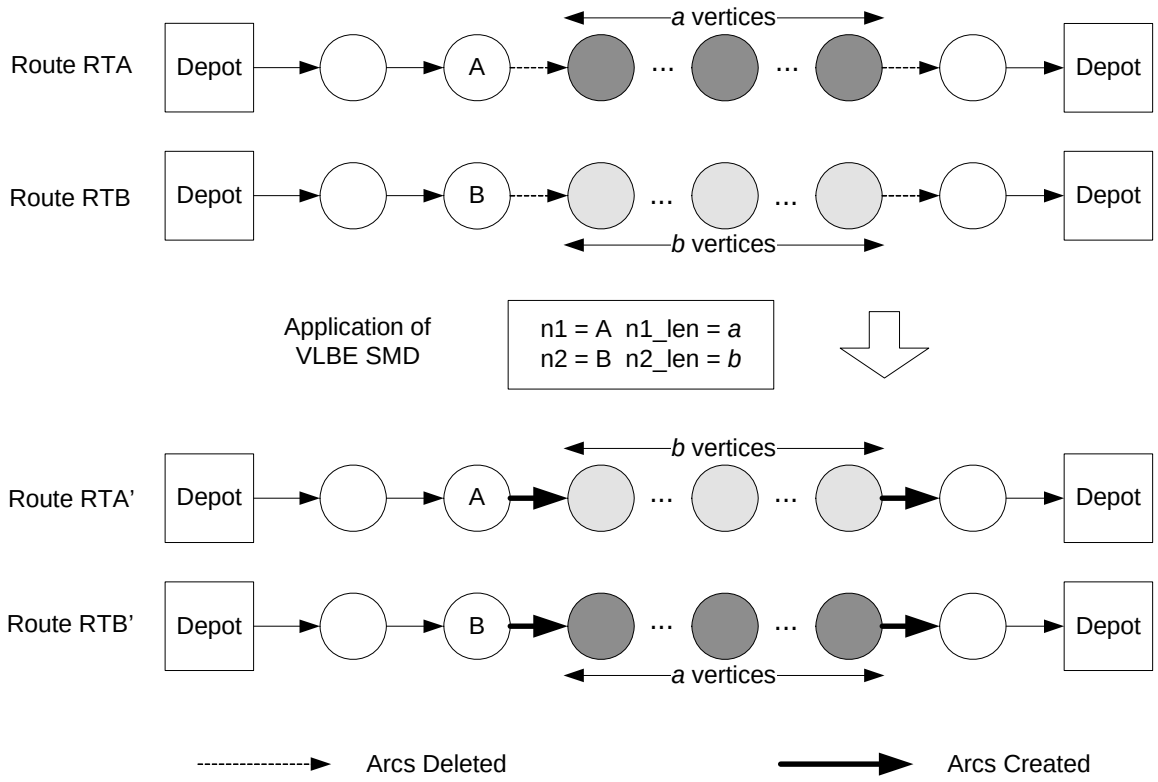


Fig 1. The VLBE local search operator

3.2.2. The 2-opt operator and its SMD representation

The 2-opt operator removes two solution arcs and replaces them with a new arc pair. When the deleted arc pair belongs to the same route, the 2-opt move implements the following solution modification: the deleted arc pair is replaced by a new one, and the path lying between these new arcs is reversed. For an inter-route 2-opt move, the routes involved are divided into their initial and terminating segments, respectively. The initial

segment of the first route is connected to the terminating segment of the second one. Analogously, the initial segment of the second route is connected to the terminating segment of the first one. The cardinality of the 2-opt neighborhood is $O(n^2)$, as one particular move is defined per vertex pair.

To encode the 2-opt moves instances into SMD instances, we generate one 2-opt SMD instance per vertex pair, so that in total $((n + K)!/(2!(n + K - 2)!))$ 2-opt SMD instances are required. Each of these instances contains two node values n_1 and n_2. The move represented by a 2-opt SMD instance with n_1 = A and n_2= B depends on whether A and B belong to the same route. If A and B belong to the same route (and without loss of generality A precedes B in the route vector), the path originating after A and terminating at B is reversed, and two new arcs are introduced to form the modified route (Fig. 2). If A and B belong to different routes, the route segment initiating from the depot and terminating at A is connected to the route segment originating after B and terminating at the depot. Similarly, the route path beginning at v_0 and terminating at B is linked to the route segment initiating after A and terminating at the depot, so that the modified route pair is generated (Fig. 3).

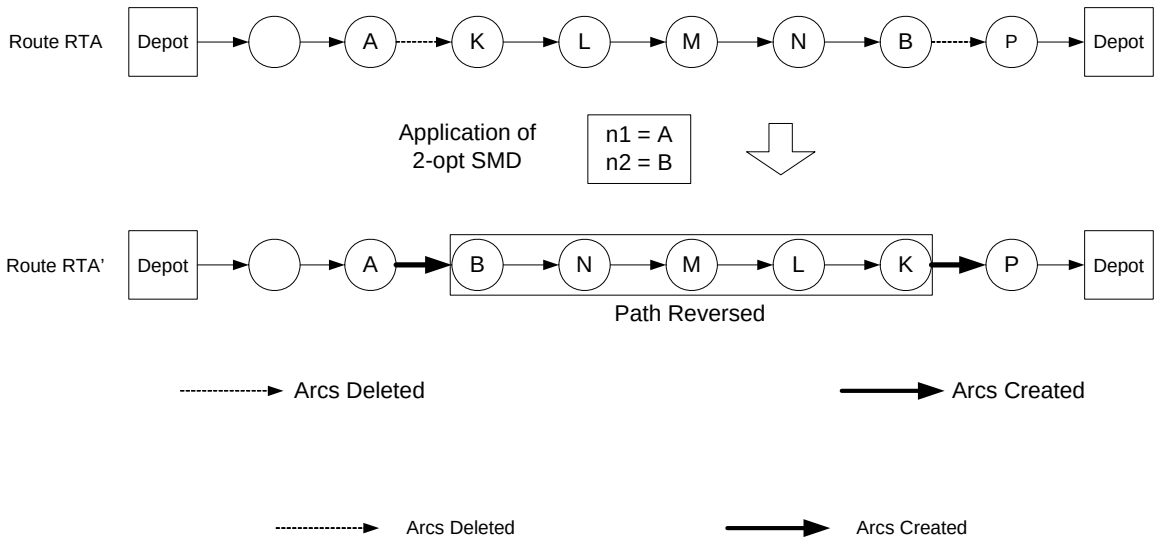


Fig 2. Intra-Route 2-opt local search operator

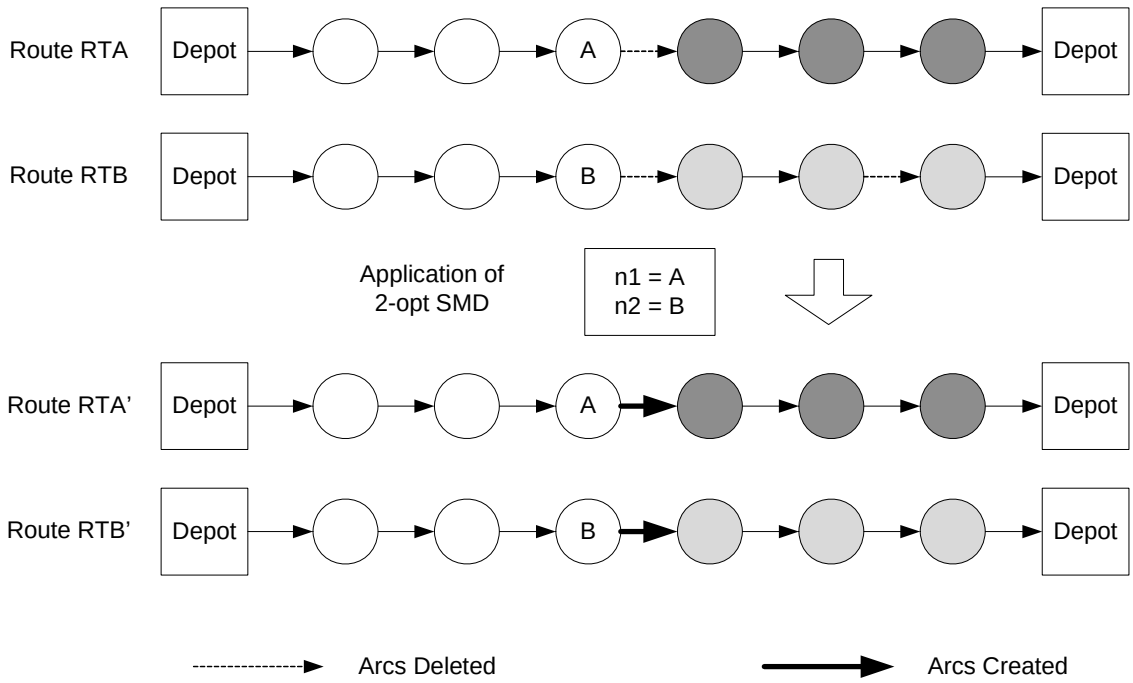


Fig 3. Inter-Route 2-opt local search operator

3.2. The cost of the local search moves

Apart from encoding a particular tentative move, both the VLBE and 2-opt SMD instances introduced in 3.1 contain a cost tag which is equal to the actual cost required for implementing that move to the candidate solution. When a move is applied to the solution, only a subset of the solution attributes is modified, so that the cost tags of only the SMD instances associated to this modified subset have to be updated according to the new solution structure. In this section, we briefly provide the SMD update cost rules for the application of VLBE and 2-opt moves, respectively. Note that these rules are reported in more detail in our previous work (Zachariadis and Kiranoudis 2009a) for the OVRP.

To improve clarity of exposition, for every VRPSPD solution, the following notation is introduced:

pred(v): denotes the bone which contains up to μ vertices and terminates before vertex v.

bone(v, a): represents the bone that contains a vertices and originates after vertex v.

succ(v, a): denotes the last vertex contained in *bone*(v, a).

part(v, y): denotes the bone originating from the successor of v and ending at vertex y.

init(v): denotes the set of predecessor vertices of v in its route.

fin(v): denotes the set of successor vertices of v in its route.

$z(v)$: denotes the total number of customers assigned to the route that contains vertex v.

$z(v, y)$: denotes the total number of vertices contained in the bone originating at the successor of v and ending at vertex y.

3.2.1. Applying a VLBE move

When a VLBE SMD instance with n_1 = A, n_2 = B, *n1_len* = a, and *n2_len* = b is applied to a candidate solution the cost tags of the following SMD instances have to be updated:

- VLBE SMD instances with either node value (n_1 or n_2) included in {{A},{B}, {*succ*(A, a)},{*succ*(B, b)}}, corresponding to $O(\mu^2 n)$ necessary updates.
- VLBE SMD instances with either node value in {*pred*(A), *pred*(B)} and relevant bone length pointing into the exchanged bones, corresponding to $O(\mu^3 n)$ updates.
- VLBE SMD instances with either node value in {*bone*(A, a-1), *bone* (B, b-1)} and relevant bone length reaching after the bones exchanged, corresponding to $O(\mu^3 n)$ updates.
- 2-opt SMD instances with either node value in {{A},{B},{*succ*(A, a)},{*succ*(B, b)}}, corresponding to $O(n)$ necessary updates.
- 2-opt SMD instances with their one node in {bone(A, a-1), bone (B, b-1)} and their other node contained in {*init*(A), *init*(B), *fin*(*succ*(A, a)), *fin*(*succ*(B, b))}, corresponding to $O(\mu (z(A) + z(B)))$ updates.

3.2.2. Applying a 2-opt move

When a 2-opt SMD instance with n_1 = A and n_2 = B is applied to a candidate solution, the necessary cost tag updates depend on whether the encoded move is applied within a single route, or it is an inter-route move.

If the 2-opt SMD encodes an intra-route move (and assuming that A precedes B in the route vector), the cost tags of the following SMD instances must be updated:

- VLBE SMD instances with either node value contained in {{A}, *part*(A,B)}, corresponding to $O(\mu^2 z(A, B))$ necessary updates.

- VLBE SMD instances with either node value in {*pred*(A)} and relevant bone length that refer into the reversed route segment, corresponding to $O(\mu^3 n)$ updates.
- 2-opt SMD instances with either node value in {{A}, *part*(A, B)}, corresponding to $O(n\, z(A, B))$ updates.

If the 2-opt SMD encodes an inter-route move the following cost tag re-evaluations must be performed:

- VLBE SMD instances with either node value contained in {{A},{B}}, corresponding to $O(\mu^2 n)$ necessary updates.
- VLBE SMD instances with either node value in {*pred*(A), *pred*(B)} and relevant bone lengths that refer to the route segments lying after nodes A and B, corresponding to $O(\mu^3 n)$ updates.
- 2-opt instances with either node value in {{A}, {B}}, corresponding to $O(n)$ updates.
- 2-opt instances with their one node in {*init*(A), *init*(B)} and their other node in {*fin*(A), *fin*(B)}, corresponding to $O(z(A)\, z(B))$ updates.

3.3 Examining the SMD instance feasibility

As earlier stated, the VRPSPD model considers that the load of vehicles fluctuates along their routes. This problem characteristic makes feasibility investigation of tentative moves much more challenging compared to the standard CVRP version, for which feasibility investigation can be straightforwardly performed in constant time.

To examine the feasibility status of the SMD instances according to the capacity requirements of the VRPSPD in O(1), we introduce several load metrics which were also used in the work of Zachariadis et al. (2009b) for examining feasibility of simple 1-0 and 1-1 exchanges and 2-opt moves.

Assume x to be the position of the vector of route RT which visits z_{RT} customer locations. Apparently, x varies from 0 to z_{RT}. The depot vertex is located at $x = 0$, whereas, at $x = z_{RT}$, lies the last customer visited by route RT. In addition, let $p_{RT}(x)$ and $d_{RT}(x)$ denote the p_i and d_i demand, respectively, of vertex v_i which is located at position x of the RT route vector. With every VRPSPD route RT, the following demand metrics are associated:

- $SPB_{RT}(x) = \Sigma_{(q = 0, 1, .., x\text{-}1)}\, p_{RT}(q)$, $x = 0, 1, \ldots, z_{RT}$

(sum of the *pick-up* demand of route RT vertices lying *before* position x).

• $SDA_{RT}(x) = \Sigma_{(q = x+1, x+2, .., ZRT)}\, d_{RT}(q)$, $x = 0, 1, \ldots, z_{RT}$

(sum of the *delivery* demand of route *RT* vertices lying *after* position *x*).

• $L_{RT}(x) = SPB_{RT}(x) + p_{RT}(x) + SDA_{RT}(x)$, $x = 0, 1, \ldots, z_{RT}$

(load of the vehicle, when travelling along the arc linking *RT* route positions *x* and *x* + 1).

• $MAX\_LB_{RT}(x) = \max_{(q = 0, 1, .., x)} \{L_{RT}(q)\}$, $x = 0, 1, \ldots, z_{RT}$

(peak load of the *RT* path which originates from the depot, and terminates at position *x* + 1 when $x < Z_{RT}$, or at the depot when $x = Z_{RT}$).

• $MAX\_LA_{RT}(x) = \max_{(q = x, x+1, .., ZRT)} \{L_{RT}(q)\}$, $x = 0, 1, \ldots, z_{RT}$

(peak load of the *RT* path which originates at position *x* and terminates at the depot).

• $MAXIM\_LA_{RT}(x, y) = \max_{(q = x, x+1, .., x+y)} \{L_{RT}(q)\}$, $x = 0, 1, \ldots, z_{RT}$, $y = 0, 1, \ldots, z_{RT} - x$

(peak load of the *RT* path which originates at position *x* and terminates at position *x* + *y* + 1 when $x + y < z_{RT}$, or at the depot when $x + y = z_{RT}$).

• $MINIM\_LA_{RT}(x, y) = \min_{(q = x, x+1, .., x+y)} \{L_{RT}(q)\}$, $x = 0, 1, \ldots, z_{RT}$, $y = 0, 1, \ldots, z_{RT} - x$

(lowest load of the *RT* path which originates at position *x* and terminates at point *x* + *y* + 1 when $x + y < z_{RT}$, or at the depot when $x + y = z_{RT}$).

The aforementioned demand metrics of any route *RT*, are updated whenever a local search move affects *RT*. The computational complexity required for their evaluation is $O(z_{RT}^2)$, where z_{RT} denotes the number of customers visited by *RT* after the local search move has been implemented.

In the following, let pst(*v*) denote the position of vertex *v* in its route vector. Obviously, if *v* is serviced by route RT, pst(*v*) varies from 0 to z_{RT}.

3.3.1. Feasibility Investigation for the VLBE SMD instances

To examine the feasibility status of a VLBE SMD instance with n_1 = A, n_2 = B, *n1_len* = *a*, and *n2_len* = b, the following checks are performed in constant time.

3.3.1.1. Intra-Route VLBE SMD instance feasibility

If both A and B belong to the same route *RT* the following preconditions must hold: pst(A) + $a \leq z_{RT}$, pst(B) + $b \leq z_{RT}$. To examine the feasibility, the following two cases must be considered.

Case A. Both bone lengths are greater than zero ($a \neq 0$ and $b \neq 0$):

Without loss of generality, assume that A precedes B in the RT route vector. To test the feasibility of the VLBE SMD instance, the following precondition must be satisfied: $pst(B) \geq pst(A) + a$. The VLBE SMD instance is infeasible if one of the following holds:

- $MAXIM\_LA_{RT}(pst(A)+a,\ pst(B)-pst(A)-a) + [SDA_{RT}(pst(A))-SDA_{RT}(pst(A)+a)] - [SPB_{RT}(pst(A)+a)+p_{RT}(pst(A)+a)-SPB_{RT}(pst(A)+1)] - [SDA_{RT}(pst(B))-SDA_{RT}(pst(B)+b)] + [SPB_{RT}(pst(B)+b)+p_{RT}(pst(B)+b)-SPB_{RT}(pst(B)+1)] > Q$.
- $MAXIM\_LA_{RT}(pst(A)+1,\ a-2) - [SDA_{RT}(pst(B))-SDA_{RT}(pst(B)+b)] - [SDA_{RT}(pst(A)+a)-SDA_{RT}(pst(B))] + [SPB_{RT}(pst(B)+b)+p_{RT}(pst(B)+b)-SPB_{RT}(pst(B)+1)] + [SPB_{RT}(pst(B)+1)-SPB_{RT}(pst(A)+a+1)] > Q$ (applicable when $a \geq 2$).
- $MAXIM\_LA_{RT}(pst(B)+1,\ b-2) - [SPB_{RT}(pst(A)+a)+p_{RT}(pst(A)+a)-SPB_{RT}(pst(A)+1)] - [SPB_{RT}(pst(B)+1)-SPB_{RT}(pst(A)+a+1)] + [SDA_{RT}(pst(A))-SDA_{RT}(pst(A)+a)] + [SDA_{RT}(pst(A)+a)-SDA_{RT}(pst(B))] > Q$ (applicable when $b \geq 2$).

Case B. Only one bone length is equal to zero (either $a = 0$ or $b \neq 0$):

If $a = 0$, set IP = A, N = B and $len = b$. Otherwise, if $b = 0$, set IP = B, N = A and $len = a$. Two subcases may arise depending on whether the relocated bone moves forward or backward in the route involved:

Subcase B1. The relocated bone moves forward ($pst(N) < pst(IP)$)

To test the feasibility of the VLBE SMD instance, the following precondition must be satisfied: $pst(IP) > pst(N) + len$. The VLBE SMD instance is infeasible if one of the following holds:

- $MAXIM\_LA_{RT}(pst(N)+len+1,\ pst(IP)-len-pst(N)-1) + [SDA_{RT}(pst(N))-SDA_{RT}(pst(N)+len)] - [SPB_{RT}(pst(N)+len)+p_{RT}(pst(N)+len)-SPB_{RT}(pst(N)+1)] > Q$.
- $MAXIM\_LA_{RT}(pst(N)+1,\ len-2) + [SPB_{RT}(pst(IP)) + p_{RT}(pst(IP)) - SPB_{RT}(pst(N)+len+1)] - [SDA_{RT}(pst(N)+len)-SDA_{RT}(pst(IP))] > Q$ (applicable when $len \geq 2$).

Subcase B2. The relocated bone moves backward ($pst(N) > pst(IP)$)

The VLBE SMD instance violates the capacity constraints if one of the following holds:

- $MAXIM\_LA_{RT}(pst(IP),\ pst(N)-1-pst(IP)) + [SPB_{RT}(pst(N)+len)+p_{RT}(pst(N)+len)-SPB_{RT}(pst(N)+1)] - [SDA_{RT}(pst(N))-SDA_{RT}(pst(N)+len)] > Q$.
- $MAXIM\_LA_{RT}(pst(N)+1,\ len-2) + [SDA_{RT}(pst(IP))-SDA_{RT}(pst(N))] - [SPB_{RT}(pst(N))+p_{RT}(pst(N))-SPB_{RT}(pst(IP)+1)] > Q$ (applicable when $len \geq 2$).

3.3.1.2. Inter-Route VLBE SMD instance feasibility

If vertices A and B belong to two different routes *RTA* and *RTB*, respectively, the following preconditions must hold: pst(A) + $a \leq z_{RTA}$, pst(B) + $b \leq z_{RTB}$. Regarding the capacity constraints, the following two cases must be considered.

Case A. Both bone lengths are greater than zero ($a \neq 0$ and $b \neq 0$):

The VLBE SMD instance violates the capacity constraints, if one of the following holds:

- $MAX\_LB_{RTA}(pst(A)) - [SDA_{RTA}(pst(A)) - SDA_{RTA}(pst(A)+a)] + [SDA_{RTB}(pst(B)) - SDA_{RTB}(pst(B)+b)] > Q$.
- $MAXIM\_LA_{RTB}(pst(B)+1, b-2) - SPB_{RTB}(pst(B)+1) + SPB_{RTA}(pst(A)+1) - SDA_{RTB}(pst(B)+b) + SDA_{RTA}(pst(A)+a) > Q$ (applicable when $b \geq 2$).
- $MAX\_LA_{RTA}(pst(A)+a) - [SPB_{RTA}(pst(A)+a)+p_{RTA}(pst(A)+a)-SPB_{RTA}(pst(A)+1)] + [SPB_{RTB}(pst(B)+b)+p_{RTB}(pst(B)+b)-SPB_{RTB}(pst(B)+1)] > Q$.
- $MAX\_LB_{RTB}(pst(B)) - [SDA_{RTB}(pst(B))-SDA_{RTB}(pst(B)+b)] + [SDA_{RTA}(pst(A))-SDA_{RTA}(pst(A)+a)] > Q$.
- $MAXIM\_LA_{RTA}(pst(A)+1,a-2) - SPB_{RTA}(pst(A)+1) + SPB_{RTB}(pst(B)+1) - SDA_{RTA}(pst(A)+a) + SDA_{RTB}(pst(B)+b) > Q$ (applicable when $a \geq 2$).
- $MAX\_LA_{RTB}(pst(B)+b) - [SPB_{RTB}(pst(B)+b)+p_{RTB}(pst(B)+b)-SPB_{RTB}(pst(B)+1)] + [SPB_{RTA}(pst(A)+a)+p_{RTA}(pst(A)+a)-SPB_{RTA}(pst(A)+1)] > Q$.

Case B. Only one bone length is equal to zero (either $a = 0$ or $b = 0$):

If $a = 0$, set IP = A, N = B, *len* = *b*, *FROM* = *RTB*, and *TO* = *RTA*. Otherwise, if $b = 0$, set IP = B, N = A, *len* = *a*, *FROM* = *RTA*, and *TO* = *RTB*.

The encoded move is infeasible, if one of the following holds:

- $MAX\_LB_{TO}(pst(IP)) + [SDA_{FROM}(pst(N)) - SDA_{FROM}(pst(N)+len)] > Q$.
- $MAXIM\_LA_{FROM}(pst(N)+1, len-2) - SPB_{FROM}(pst(N)+1) + SPB_{TO}(pst(IP)) + p_{TO}(pst(IP)) - SDA_{FROM}(pst(N)+len) + SDA_{TO}(pst(IP)) > Q$ (applicable when $len \geq 2$).
- $MAX\_LA_{TO}(pst(IP)) + SPB_{FROM}(pst(N)+len) + p_{FROM}(pst(N)+len) - SPB_{FROM}(pst(N)+1) > Q$.

3.3.2. Feasibility Investigation for the 2-opt SMD instances

The necessary checks for examining the feasibility status of a 2-opt SMD instance with n_1 = A, and n_2 = B, depend on the route pair servicing vertices A and B.

3.3.2.1. Intra-Route 2-opt SMD instance feasibility

If both A and B belong to the same route *RT*, and assuming that A precedes B in the *RT* route vector, the 2-opt SMD instance must satisfy the following preconditions: pst(A) ≤ z_{RT} - 2, and pst(B) ≥ pst(A) + 2. The capacity constraints are violated if:

- $L_{RT}(pst(A)) + L_{RT}(pst(B)) - MINIM\_LA_{RT}(pst(A), pst(B)-pst(A)-1) > Q$.

3.3.2.2. Inter-Route 2-opt SMD instance feasibility

If vertices A and B belong to two different routes *RTA* and *RTB*, respectively, the 2-opt SMD instance must satisfy the following preconditions: pst(A) ≤ z_{RTA}, and pst(B) ≤ z_{RTB}. The examined 2-opt SMD instance is infeasible, if one of the following holds:

- $MAX\_LB_{RTA}(pst(A)) - SDA_{RTA}(pst(A)) + SDA_{RTB}(pst(B)) > Q$.
- $MAX\_LA_{RTB}(pst(B)+1) - [SPB_{RTB}(pst(B)) + p_{RTB}(pst(B))] + [SPB_{RTA}(pst(A)) + p_{RTA}(pst(A))] > Q$ (applicable when pst(B) < z_{RTB}).
- $MAX\_LB_{RTB}(pst(B)) - SDA_{RTB}(pst(B)) + SDA_{RTA}(pst(A)) > Q$.
- $MAX\_LA_{RTA}(pst(A)+1) - [SPB_{RTA}(pst(A)) + p_{RTA}(pst(A))] + [SPB_{RTB}(pst(B)) + p_{RTB}(pst(B))] > Q$ (applicable when pst(A) < z_{RTA}).

3.4. The proposed implementation of the promises concept

The concept of promises, initially introduced for the VRPB, is an algorithmic mechanism for avoiding cycling and inducing diversification within any local-search procedure. Briefly, the main idea of the promise concept is the following: when a local search move is applied to a solution *S* of cost *cst*(*S*), some solution attributes are eliminated and replaced by some new solution attributes, so that solution *S′* is formed. The eliminated solution attributes are associated with a promise tag equal to the solution cost *cst*(*S*) (before the move application). Tentative local search moves are considered to be admissible (promise-keeping), only if they lead to the generation of solution attributes at a solution cost lower than the promise tags of these attributes.

For our previous promise study, complete routes were the solution attributes under consideration. On the contrary, the present work examines the behaviour of the promise mechanism for more basic solution attributes. In specific, the proposed method considers arcs to be the solution attributes exploited by the promises concept.

To check whether tentative moves are promise-keeping, with every VLBE and 2-opt SMD instance *smd* is associated a list of arcs LA_{smd}. The LA_{smd} list contains every arc (and its inverse) which is going to be introduced into the candidate solution by applying the local search move encoded by *smd*. Thus, for any SMD instance *smd* to be admissible, the cost tag of *smd* augmented by the current solution score must be lower than the promise tag of every arc contained in LA_{smd}.

Obviously, the above-presented arc lists are dynamic, as they depend on the structure of the current solution. To keep the arc list of every SMD instance updated according to the status of the candidate solution, when the cost update rules of Section 3.2 are executed, the arc lists of the affected SMD instances are appropriately modified.

3.5. The overall structure of the proposed methodology

The proposed solution approach starts off by applying a simple construction heuristic. In specific, to obtain an initial VRPSPD solution, we use the savings heuristic proposed by Paessens (1988) for the CVRP. The savings of customer insertions are evaluated as $s(i, j) = c_{i0} + c_{j0} - g\, c_{ij} + f\, |c_{i0} - c_{0j}|$, where f and g are uniformly distributed within [0, 1] and (0, 3], respectively. Note that insertion positions are considered only if they respect the capacity requirements of the examined problem. Furthermore, the examined VRPSPD model does not impose any restriction on the total number of routes of the solution. Thus, whenever a customer is assigned to an empty (of customers) route, a new empty route becomes available for subsequent customer insertions.

After the initial feasible VRPSPD solution has been produced, the improvement method, called Arc Promise Algorithm (APA), is initiated by preparing the SMD representation of the employed local search operators: the SMD instances for both the VLBE and 2-opt neighbourhoods are generated. To restrict the total SMD instance population, and thus reduce the computational effort of the overall method, we use a filtering approach aimed at excluding the generation of SMD instances which are highly unlikely to encode quality-improving local search moves. This approach is similar to the granularity strategy used for the VRPSPD in the work of Zachariadis et al. (2009a). In specific, we solve the examined instance by a VRPSPD-adapted Clarke and Wright heuristic (1964). Let $cst_{C\&W}$, and $K_{C\&W}$ denote the objective function and the total number of routes of the

solution produced. Then, a threshold value *thr* is evaluated as $thr = (\beta\ cst_{C\&W})/(n + K_{C\&W})$. An SMD instance (either a VLBE or a 2-opt one) with $n_1 = v_i$ and $n_2 = v_j$, is generated only if $c_{ij} \leq thr$, or $i = 0$, or $j = 0$. The cost tags of the generated SMD instances are calculated according to the status of the initial VRPSPD solution. Then, the SMD instances are inserted into a special priority queue structure called Fibonacci Heap (Fredman and Tarjan, 1987) which offers the following capabilities: constant time insertions, and minimum-retrievals, and logarithmic time deletions. The initialization step of the APA metaheuristic is completed by setting the promise tag of every arc of A equal to $+\infty$.

The iterative core of the APA improvement method is then applied. Every APA iteration involves the execution of three main steps: The first step involves the identification of the minimum-cost, feasible, and promise-keeping SMD instance to be applied to the current solution, denoted by *app*. To locate *app*, an inner loop must be executed: the minimum cost SMD is extracted from the Fibonacci Heap. Then, the feasibility and the promise-keeping checks are performed. The inner loop is executed in O(1), as all of its operations require constant time. Regarding the total number of iterations of the inner-loop, it depends both on the hardness of the constraints, and the status of the arc promises. Experimental runs indicated that the first APA step of determining *app* does not significantly contribute in the overall computational effort required by the proposed algorithm. After the SMD instance (and thus the local search that it encodes) has been identified, the second main algorithmic step is executed. This second step involves the implementation of the *app* SMD instance to the current solution. The VRPSPD solution is modified and the demand metrics presented in 3.3 are updated for the modified routes. In addition, the appropriate promise tag is assigned to the eliminated arcs. Finally, the third and most time-demanding APA step of updating the cost labels of the affected SMD instances is executed. An analytic discussion on the total number of the necessary SMD cost tag updates has been provided in 3.2. Note that each update is composed of three operations: deletion of the SMD from the Fibonacci Heap (O(log m), where m denotes the total number of SMD instances stored in the heap), calculation of the new SMD cost label according to the modified solution status (O(1)), and finally reinsertion of the SMD instance in the Fibonacci Heap (O(1)). To let the search intensify into promising solution

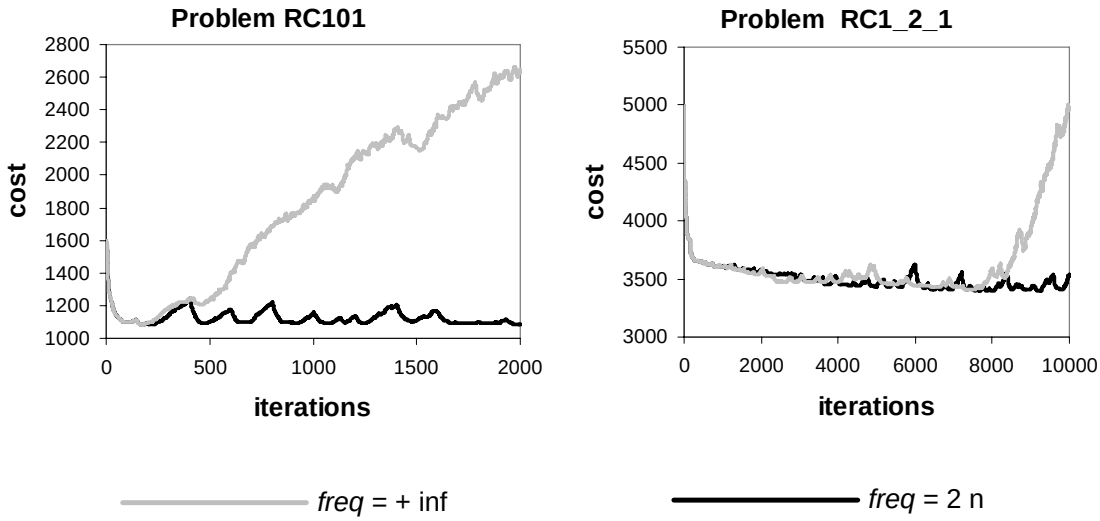
space regions, when a move improves the best solution found (so far), the promise tags of all arcs in A are reset to $+\infty$.The APA algorithm terminates when a certain time bound has been reached, by returning the overall best VRPSPD solution encountered through the search.

Preliminary executions of the above-presented APA scheme indicated the following problematic behaviour: In the initial stages of the search significant solution improvement takes place, and the diversification effect of the promises concept exhibits a satisfactory behavior. However, as the search evolves, the algorithm reaches to a point where the solution cost begins to monotonically increase, without being able to re-apply cost improving local search moves. This behaviour is due to the fact that the algorithm makes promises which are very hard to be fulfilled, causing the search to be excessively diversified: the cardinality of solution arcs is significantly lower than the cardinality of complete routes examined in our previous successful promise implementation (Zachariadis and Kiranoudis, 2009b). Thus, the algorithm gives away over-optimistic promises to the rather limited population of good-quality arcs. Subsequently, these arcs become inadmissible, and the method is forced to generate more expensive arcs which cause solution cost to increase. This cost increase makes the fulfilment of previously made promises even harder, so that the search cannot return to lower cost levels.

To eliminate this behaviour, the promise tags of arcs are periodically re-initialized. In specific, each time *freq* main APA iterations have been completed, the promise tags of all arcs contained in A are reset to $+\infty$. As will be presented in Section 4, this straightforward algorithmic addition lead to a satisfactory interplay between the intensification and diversification of the search, and subsequently helped the APA method to obtain high-quality solutions.

To visualize the effect of this promise tag re-initialization, we give Fig. 4, which illustrates the candidate solution cost curves through the search, with ($freq = 2\ n$, black line) and without ($freq = +\infty$, grey line) the periodic re-initialization of the promise tags, for two benchmark instances of 100 and 200 customers, respectively (see 4.1). As seen from the provided cost curves, in the initial phase of the runs both algorithmic configurations exhibit comparable performance. However, as the search evolves, we can see that the application of the promise tag periodic initialization leads to a balanced

algorithmic behavior, while on the contrary, when *freq* = $+\infty$, we observe an excessive diversification effect which drives the algorithm towards extremely poor-quality solution space regions.



4. Computational Results

The proposed metaheuristic algorithm has been coded in Visual C#, and executed on a single core of a T5500 processor (1.66GHz). All VRPSPD benchmark instances and the best solutions obtained can be found at http://users.ntua.gr/ezach/.

4.1 VRPSPD Benchmark Instances

To assess the performance of the APA metaheuristic, we solved the 18 large-scale VRPSPD instances introduced by Tang-Montanè and Galvão (2006). These problems involve from 100 up to 400 customers, and were originally proposed for the CVRP. Tang-Montanè and Galvão (2006) modified these 18 problems for the VRPSPD, by randomly generating the delivery and pick-up demands within the intervals used for the generation of the delivery demands of the original CVRP instances (Solomon, 1987; Gehring and Homberger, 1999). The arc costs are obtained by calculating the Euclidean distances between vertex positions, so that the generated cost matrix is both symmetric

and satisfies the triangular inequality. Table 1 summarizes the characteristics of the 18 Tang-Montanè and Galvão (2006) VRPSPD instances.

Table 1. Benchmark Instance Characteristics

Instance	***n***	***Q***	***D***	***P***
r101	100	200	1458	2339
r201	100	1000	1458	2262
c101	100	200	1810	3070
c201	100	700	1810	2910
rc101	100	200	1724	1912
rc201	100	1000	1724	2076
R1_2_1	200	200	3513	4406
R2_2_1	200	1000	3513	4358
C1_2_1	200	200	3530	5370
C2_2_1	200	700	3770	6010
RC1_2_1	200	200	3558	4473
RC2_2_1	200	1000	3558	4299
R1_4_1	400	200	7109	10433
R2_4_1	400	1000	7109	9571
C1_4_1	400	200	7190	12470
C2_4_1	400	700	7560	10050
RC1_4_1	400	200	7127	10065
RC2_4_1	400	1000	7127	10100

n: number of customers, ***Q***: Vehicle Capacity, ***D***: total delivery demand of all customers, ***P***: total pick-up demand of all customers.

4.2 Parameter Setting

To apply the APA method, we have to set the value of two algorithmic parameters: μ which determines the maximum length of the bones considered by the VLBE operator, and *freq* which, as explained in detail in 3.5, controls the APA-iteration period for re-initializing the promise tags of arcs.

In terms of the bone length μ, we experimentally executed the APA methodology taking the maximum number of customers in the bones exchanged from {4, 6, 8}. Although values 6 and 8 produced solutions of slightly higher quality, we fixed μ at 4, because we opted to keep the computational effort at minimum levels. As far as the *freq* parameter is concerned, our intention was to avoid complex tuning experiments, so that a simple tuning rule is derived. After performing preliminary experiments for solving the 18 VRPSPD benchmark instances, we observed that setting $freq = 2\ n$ resulted into a rather stable and effective performance. Following this, the APA method was executed for

solving all of the 18 VRPSPD test problems of 4.1 with the *freq* period fixed at 2 *n* iterations.

4.3 Computational Results on the Benchmark Instances

The APA methodology was applied to all 18 VRPSPD benchmark instances. The termination condition used was the completion of 60, 180, and 480 CPU seconds, for the 100-, 200- and 400-customer instances, respectively. The proposed improvement metaheuristic was applied 10 times for solving each test problem. Each of these 10 runs involved a different initial VRPSDPD solution, due to the stochastic setting of the *f* and *g* parameters used in the construction heuristic.

Table 2 summarizes the results obtained through the 10 runs. The APA methodology exhibited a rather stable performance, as the percent deviations between the best and the average solution scores over the 10 runs varied from 0.00% to 1.77%, averaging at 0.50%. In particular, for three 100-customer instances and two 200-customer instances, all 10 APA executions managed to obtain the same final solution value. Regarding the average CPU time required for obtaining the final solutions, it was limited to 18.5 seconds for problem c101, up to 421.5 seconds for the 400-customer problem R1_4_1.

Table 3 compares the scores of the best solutions obtained by the APA method, against the solution scores obtained by the tabu search of Tang-Montanè and Galvão (2006) (TG), the hybrid GLS - TS approach of Zachariadis et al. (2009a), and the evolutionary VLBR method of Zachariadis et al. (2009b). To our knowledge, the aforementioned approaches are the only previously published methodologies applied to the examined set of large-scale VRPSPD instances. From the comparative results, we can see that APA improved ten previous best-known solution scores, matched the best solution scores for seven test problems, while for problem C2_2_1, it produced a slightly worse solution value. The average best solution improvement is 0.16%, while the greatest solution improvement was observed for problem RC1_4_1 (1.30%) which involves 400 customers and approximately eight customers per route. In general, the APA methodology proved to be very effective for problems with low n / K ratios. In our opinion, this behavior is attributed to the fact that the VLBE operator is able to make drastic and effective changes in the arrangement of customers into routes. This effect is strengthened by the rather

powerful diversification role of the promises mechanism when applied to solution arcs. Regarding the size of the fleet required by the presented methodologies, we see that the APA solutions require one less vehicle for instances R1_4_1, and RC1_4_1, while for the other sixteen problems, it requires the same fleet size. In terms of the APA computational times for obtaining the best solutions, they appear to be slightly higher than the ones required by the VLBR methodology. However, we believe that the solution quality improvement compensates for the extra time required by the APA approach. Furthermore, some testing with problems of larger size ($n > 400$) indicated that APA was significantly faster than both the VLBR and GTS methods. This is because the per-iteration complexity of the two former methods is $O(n^2)$, whereas the per-iteration complexity of the APA method (which is mainly determined by the execution of the update rules of 3.2, and obviously depends on the move type performed) exhibits linearithmic ($O(n \log n)$) scalability with the instance size (for fixed μ values).

Table 2. Summary of the APA results on the benchmark instances

	AVG			**BST**			
Instance	***z***	***k***	***t***	***z***	***k***	***t***	***%gap***
r101	1009.95	12.0	28.7	1009.95	12	24.6	0.00
r201	666.20	3.0	31.4	666.20	3	22.9	0.00
c101	1222.43	16.0	18.5	1220.18	16	19.2	0.18
c201	662.07	5.0	23.5	662.07	5	18.7	0.00
rc101	1061.88	10.0	23.8	1059.32	10	25.2	0.24
rc201	673.08	3.0	21.2	672.92	3	19.0	0.02
R1_2_1	3378.52	23.0	84.6	3375.19	23	73.5	0.10
R2_2_1	1665.58	5.0	72.7	1665.58	5	67.4	0.00
C1_2_1	3652.20	28.0	57.0	3641.89	28	79.0	0.28
C2_2_1	1731.91	9.0	67.3	1726.73	9	74.8	0.30
RC1_2_1	3326.16	23.0	83.4	3316.94	23	75.7	0.28
RC2_2_1	1560.00	5.0	74.4	1560.00	5	66.9	0.00
R1_4_1	9813.42	53.4	421.5	9668.18	53	395.6	1.48
R2_4_1	3603.54	10.0	352.0	3560.73	10	306.0	1.19
C1_4_1	11325.78	63.0	384.6	11125.14	63	408.5	1.77
C2_4_1	3582.92	15.0	341.1	3549.20	15	367.7	0.94
RC1_4_1	9621.65	51.2	412.7	9520.06	51	377.0	1.06
RC2_4_1	3453.24	11.0	264.7	3414.90	11	289.5	1.11
average							*0.50*

The **AVG** group of columns provides average values over all 10 APA runs. The **BST** group of columns provides the values associated with the run which obtained the highest quality solution. ***z***: cost of the final VRPSPD solution. ***k***: number of vehicles. ***t***: time elapsed when the best solution was generated. ***%gap***: percent gap between the AVG and the BST solution scores ($= 100(z_{avg} - z_{bst}) / z_{avg}$).

Table 3. Comparison of the APA method against previous VRPSPD approaches

	TG		**GTS**		**VLBR**		**APA**		
Instance	***z***	***k***	***z***	***k***	***z***	***k***	***z***	***k***	***%gap***
r101	1042.62	12	1019.48	12	**1009.95**	12	**1009.95**	12	0.00
r201	671.03	3	**666.20**	3	**666.20**	3	**666.20**	3	0.00
c101	1259.79	17	**1220.99**	16	1220.99	16	***1220.18***	16	0.07
c201	666.01	5	**662.07**	5	**662.07**	5	**662.07**	5	0.00
rc101	1094.15	11	**1059.32**	10	**1059.32**	10	**1059.32**	10	0.00
rc201	674.46	3	**672.92**	3	**672.92**	3	**672.92**	3	0.00
R1_2_1	3447.20	23	3393.31	23	3376.30	23	***3375.19***	23	0.03
R2_2_1	1690.67	5	1673.65	5	**1665.58**	5	**1665.58**	5	0.00
C1_2_1	3792.62	29	3652.76	28	3643.82	28	***3641.89***	28	0.05
C2_2_1	1767.58	9	1735.68	9	**1726.59**	9	1726.73	9	-0.01
RC1_2_1	3427.19	24	3341.25	23	3323.56	23	***3316.94***	23	0.20
RC2_2_1	1645.94	5	1562.34	5	**1560.00**	5	**1560.00**	5	0.00
R1_4_1	10027.81	54	9758.77	54	9691.60	54	***9668.18***	53	0.24
R2_4_1	3695.26	10	3606.72	10	3572.38	10	***3560.73***	10	0.33
C1_4_1	11676.27	65	11207.37	63	11179.36	63	***11125.14***	63	0.48
C2_4_1	3732.00	15	3630.72	15	3549.27	15	***3549.20***	15	0.00
RC1_4_1	9883.31	52	9697.65	52	9645.27	52	***9520.06***	51	1.30
RC2_4_1	3603.53	11	3498.30	11	3423.62	11	***3414.90***	11	0.25
average									*0.16*
average CPU time	2.13		1.95		1.60		2.51		

T&G: the algorithm of Tang-Montanè and Galvão (2006) – (Athlon 2.0 GHz, Pascal), **GTS**: the guided tabu search of Zachariadis et al. (2009a) – (Pentium IV 2.4 GHz, Visual C#), **VLBR**: the evolutionary algorithm of Zachariadis et al. (2009b)– (T5500 1.66 GHz, Visual C#), **APA**: the proposed metaheuristic algorithm– (T5500 1.66 GHz, Visual C#), ***z***: the scores of the best solutions obtained, ***k***: number of vehicles, ***%gap***: the percent gap between the APA and the previous best solution scores (relatively to the previously best scores). Bold values represent higher quality solutions Bold and italic characters represent new best solutions.

5. Conclusions

In the present paper we have presented an effective metaheuristic algorithm for the Vehicle Routing Problem with Simultaneous Deliveries and Pick-ups (VRPSPD) which is an important routing problem variant with numerous applications in the context of reverse logistics. The proposed methodology is a local search approach which can efficiently examine rich solution neighborhoods by statically encoding tentative moves into special data structures. The solution space exploration is coordinated by the use of the promise concept which was originally introduced for the Vehicle Routing Problem with Backhauls (VRPB) and was designed to consider complete routes as the solution attributes under examination. Contrary to this aforementioned design, the promise mechanism, proposed in the present study, exploits more basic attributes, namely solution arcs. Our algorithmic design was tested on a set of 18 large-scale VRPSPD benchmark instances derived from the literature. It produced fine results, improving several previously best-known solutions.

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.

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

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

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

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

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.

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

Fredman, M., & Tarjan, R. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM, 34, 596-615

Gajpal, Y., & Abad, P. (2009). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. Computers & Operations Research, doi:10.1016/j.cor.2009.02.017.

Gehring, H., & Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In K. Miettinen, M. Mäkelä, & J. Toivanen (Eds.). Proceedings of EUROGEN99 (Vol. A2(S), pp. 57–64). Berlin: Springer.

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

Min, H. (1989). The multiple vehicle routing problem with simultaneousdelivery and pick-up points. Transportation Research Part A, 5, 377–386.

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.

Paessens, H. (1988). The savings algorithm for the vehicle routing problem. European Journal of Operational Research, 34(3), 336–344.

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

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

Toth, P., & Vigo, D. (2002). VRP with backhauls. In P. Toth & D. Vigo (Eds.), The vehicle routing problem (pp. 195–224). Philadelphia: Society for Industrial and Applied Mathematics.

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

Zachariadis, E.E., & Kiranoudis, C.T. (2009a). An Open Vehicle Routing Problem metaheuristic for examining wide solution neighborhoods. Computers & Operations Research, (to appear).

Zachariadis, E.E., & Kiranoudis, C.T. (2009b). An innovative metaheuristic solution approach for the Vehicle Routing Problem with Backhauls. Technical Report. National Technical University of Athens. (http://users.ntua.gr/ezach/).

Zachariadis, E.E., Tarantilis, C.D., & Kiranoudis, C.T. (2009a). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick- up service. Expert Systems with Applications 36 (2), 1070–1081.

Zachariadis, E.E., Tarantilis, C.D., & Kiranoudis, C.T. (2009b). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. European Journal of Operational Research, doi: 10.1016/j.ejor.2009.05.015.