CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Tabu Search Heuristic for the Split Delivery Vehicle Routing Problem with Production and Demand Calendars

Marie-Claude Bolduc
Gilbert Laporte
Jacques Renaud
Fayez F. Boctor

May 2008

CIRRELT-2008-13

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Tabu Search Heuristic for the Split Delivery Vehicle Routing Problem with Production and Demand Calendars

Marie-Claude Bolduc[1,2], Gilbert Laporte[1,3], Jacques Renaud[1,2,*], Fayez F. Boctor[1,2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Département des opérations et systèmes de décision, Pavillon Palasis-Prince, Université Laval, Québec, Canada G1V 0A6

[3] Canada Research Chair in Distribution Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract. The purpose of this article is to propose a tabu search heuristic for the split delivery Vehicle Routing Problem with Production and Demand Calendars (VRPPDC). This new problem consists of determining which customers will be served by a common carrier, as well as the delivery routes for those served by the private fleet, in order to minimize the overall transportation and inventory costs. We first model this problem and then propose a simple decomposition procedure that can be used to provide a starting solution. Next, we introduce a new tabu search heuristic and we describe two new neighbor reduction strategies. Finally, we present the results of our extensive computational tests. According to these tests, our reduction strategies are efficient not only at reducing computing time but also at improving the overall solution quality**.**

Keywords. Vehicle routing, distribution calendar, production calendar, inventory, internal fleet, common carrier, split delivery, tabu.

Acknowledgements. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants OPG0036509, 0039682-05 and OPG0172633. This support is gratefully acknowledged**.**

---

* Corresponding author: jacques.renaud@fsa.ulaval.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2008

1. Introduction

This paper proposes a tabu search heuristic for the *split delivery Vehicle Routing Problem with Production and Demand Calendars* (VRPPDC), defined as follows. Let $G=(V,A)$ be a graph, where $V=\{0,...,n\}$ is the vertex set and $A=\{(i,j):i,j\in V,i\neq j\}$ is the arc set. Vertex 0 is a distribution centre (DC), and the remaining vertices represent customers. The DC must deal with an *a priori* production calendar provided by the factory. This calendar fixes the availability of each product over a time horizon of T periods. The production calendar (g_{pt}) defines the number of units of product $p=\{1,...,P\}$ made available at the beginning of period $t=\{1,...,T\}$; g_{pt} is not cumulative and may vary from period to period. Finished products p can be stored at the DC, incurring a periodic unit inventory holding cost h_p. Each customer i has a demand calendar d_{pit}, which specifies the number of new units of product p that must be received in period t (i.e., by the delivery date).

A private fleet of m identical vehicles is based at the DC. At each period t, each customer i may be served by the private fleet vehicles $(k=1,...,m)$, by a common carrier ($k=0$), or by any combination of the two. Each vehicle in the private fleet can perform at most one route per period. The common carrier is assumed to have an unlimited capacity, while the private fleet vehicles have a capacity Q expressed in terms of volume, and each product p has a volume b_p.

For the private fleet, c_{ij} and t_{ij} represent the cost and the travel time associated with arc (i,j). The length of each vehicle route is limited by L time units per period. In practice, the common carrier charges a fixed cost f and a variable cost which is a step function of the distance l_i between the depot and customer i, and of the quantity q_{it} transported to i in period t. The lengths of the discretization intervals for distance and quantity are $\overline{l}$ and

$\bar{q}$, and the variable costs are c_l and c_q. The total cost charged by the common carrier for customer i is therefore:

$$e_{it} = f + c_l \left\lceil l_i / \bar{l} \right\rceil + c_q \left\lceil q_{it} / \bar{q} \right\rceil .$$

The VRPPDC consists of determining the routes of the private fleet, as well as the customers that must potentially be served by a common carrier, in order to minimize the overall transportation and inventory costs over a given time horizon. The resulting delivery calendar must satisfy the following constraints: *i*) for each period, each private fleet vehicle performs only one route, starting and ending at the DC; *ii*) the capacity Q of each private fleet vehicle must be respected; *iii*) the maximal length L of each route must be respected; *iv*) product availability for shipment or inventory is fixed according to the production calendar; and *v*) customers must receive their products no later than their contracted delivery dates.

The literature on the VRPPDC is rather limited. To our knowledge, most of the articles that focus on problems involving production calendars have dealt with third-party logistics for transportation. There are, however, some exceptions. Fumero & Vercellis (1999) have developed a mathematical VRPPDC model in which only a private fleet is used. Their objective was to determine both distribution and production calendars, with the latter being adapted to feed the former. This study used "on time" deliveries only. Boudia, Louly & Prins (2006) have proposed a GRASP with a reactive mechanism for solving the single product problem, while Archetti, Speranza & Hertz (2006) have developed a tabu search heuristic for solving the single-period split delivery problem with a private fleet and no production constraints. The usefulness of such split deliveries has recently been demonstrated by Archetti, Savelsbergh & Speranza (2008).

This paper makes the following contributions. We first introduce a new complex VRP variant, which, to our knowledge, has never been studied before. We then model this problem as a mixed integer program, and we solve the model using a tabu search heuristic with split deliveries. Last, we present two new neighbor reduction strategies which, when tested, proved efficient not only at reducing computing time but also at improving overall solution quality.

The remainder of this article is organized as follows. We provide our formulation (i.e., model) of the problem in Section 2. In Section 3, we present our tabu search heuristic and the new neighbor reduction strategies. Section 4 describes our computational experiments and our results, and Section 5 offers our conclusions.

2. Problem formulation

In addition to the notation already introduced, define the following variables:

$x_{ijkt} = \begin{cases} 1 \text{ if vehicle } k \text{ visits a vertex } j \text{ immediatly after vertex } i \text{ at period } t \\ 0 \text{ otherwise;} \end{cases}$

$y_{kt} = \begin{cases} 1 \text{ if vehicle } k \text{ is used in period } t \\ 0 \text{ otherwise;} \end{cases}$

u_{ikt} : upper bound for the load of vehicle k upon leaving customer i in period t;

s_{pt} : inventory of product p at the DC at the end of period t;

q_{pikt} : quantity of product p leaving the DC to be delivered to customer i with vehicle k in period t. Note that $q_{it} = \sum_{p=1}^{P} q_{pi0t},\ t \in \{1,...,T\}$.

The model is expressed as follows:

Minimize $$\sum_{i=0}^{n}\sum_{\substack{j=0 \\ j\neq i}}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T}\left(c_{ij}x_{ijkt} + e_{it}\right) + \sum_{p=1}^{P}\sum_{t=1}^{T} h_p s_{pt} \tag{1}$$

subject to $$s_{pt} = s_{p,t-1} + g_{pt} - \sum_{i=1}^{n}\sum_{k=0}^{m} q_{pikt} \qquad \left(p \in \{1,...,P\}; t \in \{1,...,T\}\right) \tag{2}$$

$$\sum_{k=0}^{m}\sum_{h=1}^{t} q_{pikh} \geq \sum_{h=1}^{t} d_{pih} \qquad \left(p \in \{1,...,P\}; i \in \{1,...,n\}; t \in \{1,...,T\}\right) \tag{3}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{0jkt} \leq m \qquad \left(t \in \{1,...,T\}\right) \tag{4}$$

$$\sum_{\substack{j=0 \\ j\neq h}}^{n} x_{hjkt} = \sum_{\substack{i=0 \\ i\neq h}}^{n} x_{ihkt} \qquad \left(h \in \{0,...,n\}; k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{5}$$

$$\sum_{i=0}^{n}\sum_{\substack{j=0 \\ j\neq i}}^{n} t_{ij}x_{ijkt} \leq L \qquad \left(k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{6}$$

$$\sum_{p=1}^{P} q_{pikt} \leq Q\sum_{\substack{j=0 \\ j\neq i}}^{n} x_{jikt} \qquad \left(i \in \{1,...,n\}; k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{7}$$

$$\sum_{i=0}^{n}\sum_{\substack{j=0\\ j\neq i}}^{n} x_{ijkt} \leq (n+1)\, y_{kt} \qquad \left(k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{8}$$

$$\sum_{p=1}^{P}\sum_{i=1}^{n} b_p q_{pikt} \leq Q y_{kt} \qquad \left(k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{9}$$

$$u_{ikt} - u_{jkt} + (n-1)\, x_{ijkt} \leq n-2 \qquad \left(i, j \in \{1,...,n\}, i \neq j; k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{10}$$

$$x_{ijkt} \in \{0,1\} \qquad \left(i, j \in \{0,...,n\}, i \neq j; k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{11}$$

$$y_{kt} \in \{0,1\} \qquad \left(k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{12}$$

$$u_{ikt} \geq 0 \qquad \left(i \in \{1,...,n\}; k \in \{1,...,m\}; t \in \{1,...,T\}\right) \tag{13}$$

$$s_{pt} \geq 0 \qquad \left(p \in \{1,...,P\}; t \in \{0,...,T\}\right). \tag{14}$$

The objective function minimizes the routing costs of the private fleet, the non-linear costs of the common carrier, and the DC's inventory costs. Constraints (2) maintain the product balance by adjusting the DC's inventory level according to incoming production and outgoing shipments. Constraints (3) ensure that the contracted customer demand calendars are satisfied. Constraints (4) specify that at most m private fleet vehicles can be used each day, while constraints (5) indicate that the same vehicle k must arrive at and leave from customer h. Constraints (6) require that each private fleet route lasts at most L time units. Constraints (7) and (8) apply only to the private fleet, respectively specifying that deliveries can only be made if a tour exists already and that a tour exists only if a private fleet vehicle is used. Constraints (9) ensure that the vehicle capacity is never exceeded. Constraints (10) were introduced by Miller, Tucker & Zemlin (1960) eliminate subtours.

3. Tabu search heuristic

The model presented in Section 2 cannot be solved optimally for any realistic size instance because, in addition to having a non-linear objective, it involves too many binary variables. For this reason, we have developed a tabu search heuristic that allows split deliveries. It includes a common carrier and thus must repeatedly evaluate a stepwise cost function, which implies an additional computational effort for each neighborhood evaluation. It also considers intermediate infeasible solutions (Gendreau, Hertz &

Laporte, 1994; Cordeau, Gendreau & Laporte, 1997). More specifically, vehicle capacity and route length may be violated during the search but they must be respected in the final solution. However, the production calendar (i.e., inventory availability) and the demand calendar (i.e., inventory quantities and the subsequent delivery dates) are always enforced during the search.

These hard and soft constraints lead to an important difference with respect to the other tabu searches proposed in the literature: the double exploration of each neighbor, respectively called "complete switch" and "partial switch". The complete switch evaluates the possibility of transferring all the product quantities for one customer from the original route (corresponding to a vehicle-period combination) to another one, while still respecting the hard constraints. Such a move may or may not generate feasible solutions. The partial switch is a split delivery strategy, consisting of moving only the quantities that respect the target vehicle's availability in terms of space and time. Such a partial transfer will always lead to a feasible route.

The following subsections present, in turn, the general procedure of our heuristic, its step-by-step description and our new neighbor reduction procedures.

3.1. General Procedure of the VRPPDC Heuristic

This section describes the general procedure which is composed of the initial feasible solution, the tabu search phase and the improvement phase.

3.1.1. Initial feasible solution

If the quantities to be delivered are determined for each product and each period, the routing problem for each period corresponds to a multi-product version of the *Vehicle Routing Problem with a Private fleet and a Common carrier* (VRPPC) as defined by Bolduc *et al.* (2006). The VRPPC consists of minimizing the total cost of serving once a set of customers using either the limited private fleet vehicles or the common carrier and, for the private fleet, to perform the routes.

Two heuristics have recently been developed for the VRPPC: the Selection, Routing and Improvement heuristic (SRI – Bolduc, Renaud & Boctor, 2006) and the Randomized construction, Improvement, Perturbation metaheuristic (RIP – Bolduc *et al.*, 2006). SRI is

a three phase heuristic: 1) selection of customers to be served by the external carrier, 2) construction of an initial solution, and 3) improvement procedure. This heuristic is quite fast and generates better results than a previous algorithm. RIP is a five step metaheuristic: 1) randomized savings construction phase, 2) a 4-opt* route improvement procedure, 3) 2*-interchange inter-route improvement procedure, 4) 2-add-drop improvement procedure, and 5) switch procedure. In the VRPPC computational results, RIP generated better results than SRI, but required much longer computational processing times.

The heuristics developed for the VRPPC can be adapted to generate solutions to the single-period subproblems. Indeed, adaptations are necessary since these VRPPC heuristics were designed to handle only one product.

The rest of this subsection presents a two-phase method designed to allow SRI and RIP to be used for the solution of single-period multi-product subproblems. The two phases solve the problem for each period independently, and the best result of each phase is retained for each period.

Two-phase method using SRI and RIP

Phase 1: Based on the demand calendars, generate a list of customers with a demand due in the current period. For each customer, generate a new temporary customer for each product p ordered (each customer may be split into, at most, P customers) and evaluate the common carrier cost for this scenario. This cost is needed to solve the VRPPC. Solve this subproblem for each period, using either the SRI or the RIP heuristic. This strategy may produce large subproblems since each customer may be split a number of times.

Phase 2: Generate a list of customers with a demand due in the current period. For each customer, create a single unified demand, corresponding to the sum of all product demands, and evaluate the common carrier cost for this scenario. Solve this subproblem for each period, using either the SRI or the RIP heuristic.

Repeating these strategies for each period leads to one single feasible initial solution, composed of all the routes obtained for each period. This solution can be enhanced through a tabu exploration phase and an improvement phase.

3.1.2. Tabu search phase

The structure of our tabu search heuristic is similar to that of Cordeau, Gendreau & Laporte (1997). The notation used is provided in Table 1. The heuristic works with the penalized cost function $f(s) = c(s) + h(s) + \alpha q(s) + \beta d(s)$, where

$$c(s) = \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} \sum_{k=1}^{m} \sum_{t=1}^{T} \left(c_{ij} x_{ijkt} + e_{it} \right), \qquad h(s) = \sum_{p=1}^{P} \sum_{t=1}^{T} h_p s_{pt},$$

correspond to the objective function (1). Violations of the vehicle capacity or route duration constraints are penalized according to the following functions:

$$q(s) = \sum_{k=1}^{m} \sum_{t=1}^{T} \left[\sum_{p=1}^{P} \sum_{i=1}^{n} b_p q_{pikt} - Q \right]^+, \qquad d(s) = \sum_{k=1}^{m} \sum_{t=1}^{T} \left[\sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} t_{ij} x_{ijkt} - L \right]^+,$$

and $[x]^+ = \max\{0, x\}$. When s is feasible, $q(s)$ and $d(s)$ both equal to zero. The values α and β are positive and are updated at each tabu iteration by $(1+\delta)$, where δ is a user-controlled parameter. The value α is divided by $(1+\delta)$ f the current solution respects vehicle capacities, and is multiplied by $(1+\delta)$ otherwise. The same updates are applied to β with respect to tour duration feasibilities.

Each solution s has an associated attribute set $B(s) = \{(i,k,t)\}$, in which customer i is served by vehicle k at period t. Moving from a solution s to a neighbor solution $s' \in N(s)$ modifies $B(s)$. The neighborhood $N(s)$ of a solution s is composed of all solutions obtained by moving a quantity $v_{piktk't'}$ associated to customer i from route (k,t) to (k',t'), where $k' \neq k$ or $t' \neq t$. For a customer i and the initial route (k,t) under consideration, the movable quantities $v_{piktk't'}$ depend on the type of switch. In the complete switch, when the target period t' is earlier than the original period t, $v_{piktk't'}$ is

the minimum movable quantity possible between the delivered quantity and the available inventory from period t' to period t. However, when the period t' is later than the original period t, $v_{piktk't'}$ is equal to the delivered quantities. In the partial switch, $v_{piktk't'}$ also takes the available vehicle capacity into account. To summarize, $v_{piktk't'}$ may be formulated as follows:

complete switch:
$$v_{piktk't'} = \begin{cases} \min\left\{q_{pikt}, \min\limits_{h=t',\ldots,t}\left\{s_{pih}\right\}\right\}, \text{if } t' < t \\ q_{pikt}, \text{ otherwise,} \end{cases}$$

partial switch:
$$v_{piktk't'} = \begin{cases} \min\left\{q_{pikt}, \min\limits_{h=t',\ldots,t}\left\{s_{pih}\right\}, Q - \sum\limits_{h=1}^{n} q_{phk't'}\right\}, \text{if } t' < t \\ \min\left\{q_{pikt}, Q - \sum\limits_{h=1}^{n} q_{phk't'}\right\}, \text{ otherwise.} \end{cases}$$

Table 1 – Notation used in the tabu search algorithm

$B(s)$:	attribute set for solution s $\left(B(s)=\{(i,k,t)\},\quad i=\{1,...,n\},\quad k=\{1,...,m\},\quad t=\{1,...,T\}\right)$;
$N(s)$:	neighborhood for solution s;
$c(s)$:	transportation cost for solution s;
$h(s)$:	holding inventory costs for solution s;
$q(s)$:	total excess quantity for solution s;
$d(s)$:	total excess duration for solution s;
$f(s)$:	total cost for solution s, $f(s)=c(s)+h(s)+\alpha q(s)+\beta d(s)$;
$g(s)$:	total penalized cost for solution s (i.e., $f(s)$, to which is added a penalty term for the most frequently visited customers in order to diversify the search);
$s,\overline{s}$:	solutions;
s' :	best solution identified in the current neighborhood $N(s)$;
s^* :	best feasible solution identified;
α :	penalty factor for capacity violation;
β :	penalty factor for tour duration violation;
ρ_i :	number of times a customer i has been added to the solution;
σ_{ikt} :	aspiration criterion of attribute (i,k,t);
η :	total number of iterations to be performed;
θ :	tabu duration;
δ :	parameter used to update α and β;
λ :	iteration counter;
v_p :	movable quantities of product p;
τ_{ikt} :	last iteration for which attribute (i,k,t) is declared tabu;
ξ :	number of iterations without improving s^*;
μ :	average number of iterations needed to pass from one feasible solution s' to the next;
κ :	number of iterations a solution s' remains infeasible.

The aspiration criterion σ_{ikt} is initially set equal to the initial cost solution for the attributes included in the initial solution and is set equal to ∞ for the other attributes. As the number of tabu iterations increases, this criterion is updated using the attributes of the most recent best solution.

To evaluate a solution $\bar{s}$, a *tabu exploration phase* is completed. For given attributes (i,k,t) and (i,k',t'), this phase includes the following steps:

1. Set the current solution $\bar{s}$ equal to the initial solution in the neighborhood ($\bar{s} = s$). (In the following steps, $(k',t') \in \bar{s}$ and $(k,t) \in s$).
2. If customer i is not already in route (k',t'), insert this customer into the route and apply the 3-opt procedure (Lin, 1965).
3. Move $v_{piktk't'}$ for all p from route (k,t) to (k',t'). If $v_{piktk't'} < q_{pikt}$ (i.e., only some of the original delivery quantity is moved), prioritize the product that has the highest inventory cost.
4. If $v_{piktk't'} = q_{pikt}$, remove the customer from this route and reoptimize it using 3-opt.
5. For periods t to t', update the inventories s_{pt} and reevaluate the global inventory cost.
6. Reevaluate the routing cost for both routes.
7. If the solution $\bar{s}$ is non-tabu $(\tau_{ikt} < \lambda)$ or if a best feasible solution is obtained ($f(\bar{s}) < \sigma_{ik't'}$ and $\bar{s}$ is feasible), then apply the following operations:

   (a) If the current solution $\bar{s}$ is worse than the initial solution s in the neighborhood $(f(\bar{s}) > f(s))$, $f(\bar{s})$ is penalized for diversification $g(\bar{s}) = f(\bar{s}) + 0.02\sqrt{mnT}\,(c(\bar{s}) + h(\bar{s}))\,\rho_i/\lambda$; otherwise, $g(\bar{s}) = f(\bar{s})$.

   (b) If the current solution $\bar{s}$ is the new best neighbor $(g(\bar{s}) < f(s'))$, set $s' = \bar{s}$.

When a solution s' is retained as the best solution found in $N(s)$, customer i cannot be reinserted into route (k,t) for the next θ iterations. During these θ iterations, a tabu move is allowed only if its global cost is lower than the aspiration criterion σ_{ikt}. This criterion is updated whenever a new best feasible solution is obtained.

To diversify the search, the attributes repeatedly inserted into the solution are penalized by a term proportional to ρ_i (see Step 7(a)). Because of the split quantities, penalizing the attribute (i,k,t) is not appropriate since using a penalty ρ_{ikt} may lead to moving the

same customer to several different routes. To diversify the solution, different customers are instead moved at each iteration and their moves to other routes are penalized using ρ_i. The other variables used in the determination of $g(s)$ are the same as those mentioned by Cordeau, Gendreau & Laporte (1997), except for the constant factor, which is set to 0.02 instead of 0.015. After testing many values, we found that 0.02 generated the best results.

3.1.3. Improvement phase

When the tabu iterations are completed, the following improvement heuristic is applied:

1. advance the delivery of complete blocks of product demand;
2. advance the delivery of partial blocks of product demand;
3. switch an internal customer (i.e., one served by a private vehicle) with an external customer (i.e., one served by a common carrier).

For each route (k',t'), the first two steps verify whether it is possible to deliver, at t', a delivery planned for a later time t. Step 1 evaluates whether, in terms of inventory availability and the remaining vehicle capacity, a given customer's deliveries, planned for a later period (i.e., in period t, where $t > t'$) can be completely integrated into route (k',t') at an earlier period. After all current customers have been evaluated, this procedure is repeated for the partial blocks (Step 2). For a product that has been planned for delivery in quantity q_{pikt}, the delivery of any quantity between 1 and q_{pikt} may be moved to an earlier period. These two steps allow the global inventory cost to be reduced by filling the vehicles scheduled as much as possible.

Step 3 is similar to the Osman 1-1 exchange (1993). The difference between Step 3 and the original method is that the switches between customers served by a private fleet vehicle and those served by common carrier are the only ones evaluated. When a customer switch is performed, the entire product quantities for these two customers are moved. In other words, there are no partial transfers. A switch can be conducted for any vehicle-period-customer combination, as long as the problem constraints are respected and the global cost (i.e., transportation and inventory costs) is reduced. Then, 3-opt is applied to each modified route.

3.2. Step-by-step description of the VRPPDC heuristic

A step-by-step description of the VRPPDC heuristic is given below.

1. Obtain the initial feasible solution s^* using the *two-phase method using SRI and RIP* outlined in Section 3.1.1.
2. Set $\rho_i = 0$, for all i.
3. For each attribute (i,k,t) included in the initial best feasible solution s^*, set $\sigma_{ikt} = f(s^*)$; otherwise, set $\sigma_{ikt} = \infty$.
4. For each tabu search iteration $(\lambda = 1,...,\eta)$, perform the following operations:

   (a) Set the best neighborhood solution value $f(s') = \infty$.

   (b) Set the initial solution in the neighborhood $s = s^*$.

   (c) For each attributes (i,k,t) and (i,k',t'), where $k \neq k'$ or $t \neq t'$, do the following. If $t' > t$ and the current solution $\bar{s}$ do respect the delivery dates, go to Step 4(c1); otherwise, go to Step 4(c2).

      (c1) COMPLETE SWITCH.

         1. For each product p, calculate $v_{piktk't'}$. If $\sum_{p=1}^{P} v_{piktk't'} = 0$, restart Step 4(c) using the next neighbor.
         2. Apply the *tabu exploration phase* described in Section 3.1.2.

      (c2) PARTIAL SWITCH.

         1. For each product p, calculate $v_{piktk't'}$. If $\sum_{p=1}^{P} v_{piktk't'} = 0$, restart Step 4(c) using the next neighbor.
         2. Apply the *tabu exploration phase* described in Section 3.1.2 for all p, where $v_{piktk't'} \neq 0$.

   (d) Set the best neighbor s' tabu $(\tau_{ikt} = \lambda + \theta)$ and adjust the diversification criterion $(\rho_i = \rho_i + 1)$.

   (e) If the best neighbor s' is a new best feasible solution (i.e., $f(s') < f(s^*)$ and s' is feasible), perform the following operations:

      (e1) Set $s^* = s'$.

      (e2) For each attribute (i,k,t) included in the solution s', update the aspiration criteria $(\sigma_{ikt} = \min\{\sigma_{ikt}, f(s')\})$.

(f) If $q(s')=0$, then $\alpha=\alpha/(1+\delta)$. Otherwise, $\alpha=(1+\delta)\alpha$.

(g) If $d(s')=0$, then $\beta=\beta/(1+\delta)$. Otherwise, $\beta=(1+\delta)\beta$.

5. Apply the improvement phase described in Section 3.1.3 to the best solution s^*.

3.3. Neighbor reduction strategy

The tabu search can be very time consuming due to the large size to the neighborhood $N(s)$ and also to the non-linear cost functions that must be constantly reevaluated. Thus, we propose two neighbor reduction strategies designed to reduce the computing time.

The first strategy, called *Random*, involves randomly selecting some neighbors from $N(s)$ for evaluation, thus allowing only a predefined proportion of the total neighborhood to be considered. This strategy is quite simple and may be used on any kind of problem. By forcing a decrease in the number of neighbors, the strategy is able to decrease the time needed to find a solution. However, because the strategy does not necessarily select the best neighbor, it could lead to a different, and maybe worse, solution.

The second strategy, called *Distance*, takes the problem configuration into account. The idea is simple: a given neighbor is evaluated only if the customer under consideration is within a fixed distance from the route into which the customer will be inserted. The fixed distance is predefined as being equal to a fraction of the longest distance on the geographical map. This strategy makes it possible not to evaluate the insertion of a customer into a distant route.

4. Computational results

We first describe our test instances, then present the tabu search parameters, and finally report our results.

4.1. Test instances

For each of the 100 instances, 50 customer coordinates were randomly generated on a plane measuring 240 km by 200 km. The number of products was set equal to 3; the

inventory costs per product unit were \$0.15, \$0.25 and \$0.35, respectively; and the planning horizon was equal to 10 periods.

To determine the demand calendar for each customer, the following rules were applied. First, each customer demand level was classified accordingly to the following probabilities: 25% had a low demand per period, 50% had an average demand per period (i.e., twice the low demand), and 25% had a high demand per period (i.e., four times the low demand). The upper bounds of low demand products were set equal to 2, 4 and 3 units for product 1, 2 and 3, respectively. For each period, demands were randomly generated according to a continuous uniform distribution between 0 and an upper bound based on the customer category. For each customer, the ordering interval was randomly selected between one and four periods; the day of the first order was then selected. The quantity of each order was the sum of the customer's daily demand over the ordering interval.

The production calendar was designed according to the customer demand calendars. For each product, the initial inventory at the DC was equal to the average demand over five periods, and the production lot size was the average demand divided by the production frequency, which was set randomly between one and five periods. Since the factory was assumed capable of producing the demand for five periods of any product in any one period, the production of each period was assigned entirely to the most urgent product remaining to be delivered, chosen with respect to the minimum ratio between the current stock level and the demand to be filled.

The parameters of the common carrier cost function were set as: $f = 60$, $c_l = 50$, $\bar{l} = 25$, $c_q = 15$, $\bar{q} = 0.25Q$.

When determining the number and capacity of the private fleet vehicles, we have assumed that, on average, about 75% of the demand would be filled by this fleet, while the other 25% would be filled by a common carrier. The number of vehicles was initially set equal to 2, and this number was increased until the demand could be covered by the number of vehicles selected, each with a capacity between 100 and 150 units. The exact vehicle capacity was determined with respect to the demands. For example, a demand of

500 units per period would require a total fleet capacity of at least 375 (75% of 500). Since each vehicle is assumed to have a capacity between 100 and 150 units, two vehicles would have a total capacity within the interval [200, 300], while three vehicles would have a capacity of [300, 450]. Thus, for a demand of 375, three vehicles would be necessary, each with a capacity of 125 (i.e., 375 divided by 3). The volume of each product was set to 1, and the vehicle speed was set to 80 km/h, the maximum route length to 13 time units, and the variable distance cost to $2/km. Table 2 presents the number of vehicles, their capacity and the total demand for each instance.

Table 2 – Some instance characteristics

Instance	*m*	*Q*	*Total demand*	Instance	*m*	*Q*	*Total demand*	Instance	*m*	*Q*	*Total demand*
1	3	110	4230	35	3	110	4216	68	3	110	4422
2	3	130	5089	36	3	110	4190	69	3	120	4475
3	2	140	3692	37	3	100	3998	70	3	110	4287
4	3	120	4550	38	3	100	4064	71	3	120	4528
5	3	100	4015	39	2	140	3643	72	3	110	4300
6	3	110	4242	40	2	150	3815	73	3	110	4146
7	3	120	4789	41	3	110	4226	74	3	110	4343
8	3	110	4146	42	3	140	5291	75	3	120	4690
9	3	110	4090	43	3	110	4191	76	3	120	4787
10	3	110	4188	44	3	120	4850	77	3	100	3991
11	3	110	4292	45	2	130	3324	78	3	110	4178
12	3	110	4373	46	2	140	3570	79	3	110	4266
13	3	120	4601	47	3	100	3887	80	3	110	4220
14	2	150	3774	48	3	140	5483	81	3	120	4541
15	3	120	4601	49	3	130	4896	82	3	130	5015
16	3	120	4683	50	3	120	4502	83	3	110	4420
17	3	110	4404	51	3	120	4452	84	2	130	3283
18	3	120	4459	52	3	110	4211	85	3	110	4099
19	3	100	3989	53	3	110	4249	86	3	110	4257
20	3	100	3977	54	3	110	4254	87	3	110	4251
21	3	100	4063	55	3	120	4515	88	3	120	4490
22	3	130	5121	56	2	140	3716	89	3	110	4284
23	3	120	4606	57	3	110	4290	90	3	120	4534
24	3	120	4656	58	3	120	4590	91	3	100	4004
25	3	120	4704	59	2	150	3867	92	3	120	4517
26	3	100	3924	60	3	110	4103	93	3	120	4615
27	3	110	4439	61	3	110	4153	94	3	100	3922
28	3	110	4161	62	3	120	4575	95	3	120	4608
29	3	120	4457	63	3	110	4212	96	3	110	4253
30	3	110	4391	64	3	120	4581	97	2	150	3827
31	3	100	3974	65	3	120	4483	98	3	100	4027
32	3	100	3888	66	3	100	3900	99	3	100	4070
33	3	110	4253	67	3	110	4392	100	3	110	4214
34	3	130	4982								

4.2. Tabu search parameters

The number of iterations of the tabu search algorithm in the VRPPDC heuristic was set $\eta = 100,000$ iterations. Two solution procedures (A and B), with different intensification and diversification strategies, were used.

Solution Procedure A

Procedure A used the VRPPDC heuristic as described in Section 3.2, with the following parameter values: $\alpha = 1$, $\beta = 1$, $\theta = \lfloor 7.5 \log_{10} n \rfloor$ and $\delta = 1.5$.

Solution Procedure B

Procedure B was designed to adapt the parameter values as the solution evolved. First, the penalty factors were set at $\alpha = 0.5$ and $\beta = 0.5$. Because the VRPPDC heuristic starts from an initial feasible solution, lower penalty factors can be used at first. In addition, the problem considered in this paper can lead to many tabu iterations with infeasible solutions. Since many infeasible solutions may be generated in a neighborhood before a new feasible solution is obtained, we have added a step in our algorithm before Step 4(f) updating the penalty factors α and β:

(f') If s' is feasible, or if $\kappa \leq \mu$, then $\delta = 1.5$; otherwise, $\delta = 1$ until s' becomes feasible.

To prevent the tabu search algorithm from being stuck in a local optimum with only a few neighbors, we also adjusted the criterion in Step 4(d) (Section 3.2) to read: $\theta = \lfloor 7.5 \log_{10} n \rfloor + \lceil \xi / 100 \rceil$. This variable tabu duration allows the search to visit other neighbors.

4.3. Results

The heuristic was coded in Microsoft VB.Net 2005 and run under Windows XP on a personal computer with a Xeon 3.6 GHz processor and 1.00 Go of RAM. All reported times are expressed in seconds, and all statistics represent the average results over a set of 100 test instances. The following results are compared with the best known solutions obtained for all instances (see Table 3). This section presents our computational

experiments with our tabu search heuristic. We first present the results of the initial feasible solution and improvement phase. The neighbor reduction parameters are then analyzed. Finally, complete results for the VRPPDC heuristic are presented.

Table 3 – Best known solutions

Instance	Best	Instance	Best	Instance	Best	Instance	Best
1	23,113.57	26	24,745.46	51	27,856.98	76	26,828.62
2	25,391.41	27	27,181.70	52	24,289.10	77	27,672.02
3	18,268.49	28	25,466.19	53	20,583.28	78	22,771.41
4	23,585.96	29	23,228.56	54	24,700.93	79	26,817.43
5	25,173.42	30	29,169.52	55	27,918.29	80	24,737.76
6	27,788.49	31	24,635.79	56	19,058.70	81	26,776.09
7	27,581.27	32	25,759.79	57	25,576.13	82	23,887.33
8	24,156.73	33	22,809.43	58	27,075.20	83	28,674.86
9	21,783.57	34	23,604.40	59	20,262.76	84	19,024.32
10	25,171.74	35	26,281.08	60	23,327.91	85	24,307.21
11	28,200.50	36	24,837.63	61	24,512.32	86	23,319.70
12	28,937.65	37	26,247.15	62	24,346.14	87	25,379.44
13	26,087.67	38	25,163.93	63	25,515.31	88	25,831.23
14	18,151.88	39	19,983.13	64	27,200.36	89	23,950.72
15	25,250.77	40	20,232.92	65	25,266.38	90	25,659.14
16	27,917.18	41	24,333.88	66	26,227.86	91	25,024.39
17	27,729.59	42	24,578.88	67	25,801.10	92	22,336.13
18	22,731.64	43	24,127.50	68	28,676.45	93	25,274.56
19	22,629.94	44	24,602.39	69	25,742.31	94	25,586.53
20	25,401.72	45	20,382.55	70	24,900.09	95	25,879.00
21	26,403.15	46	20,443.80	71	25,947.90	96	23,210.87
22	26,359.05	47	22,683.97	72	25,470.19	97	19,520.18
23	27,477.55	48	26,214.30	73	23,799.70	98	22,771.17
24	24,901.55	49	24,993.88	74	23,840.44	99	26,181.31
25	26,618.87	50	24,243.11	75	26,594.19	100	24,832.96

4.3.1. Evaluation of the initial feasible solution and improvement phase

Our two-phase method (3.1.1) used to obtain the initial feasible solution is clearly inefficient, since, as shown in Table 4, the average deviation above the best known solution is 57.23% for the RIP-based heuristic and 64.01% for SRI-based heuristic. The improvement phase yields slightly better results: 49.15% for RIP and 51.28% for SRI. However, RIP solution times are much longer than those of SRI, with an average solution time of 301 seconds compared to only two seconds for SRI. The results reported in this section for the VRPPDC heuristic were initiated from the SRI-based solution, called SRITabu, which was much faster.

Table 4 – Results for the initial feasible solution and the improvement phase heuristics

Initial feasible solution	RIP		SRI	
Improvement phase	Without	With	Without	With
Average deviation from the best-known solution	57.23%	49.15%	64.01%	51.28%
Average time (sec)	301	301	2	2
Number of best solutions	0	0	0	0

4.3.2. Neighbor reduction parameters

To evaluate the proposed neighbor reduction strategies, all 100 instances were solved with SRITabu and 500 iterations, using solution procedure A and the two neighbor reduction strategies. The goal of the evaluation was to demonstrate that, using our neighbor reduction strategies, solutions of the same quality could be obtained in less time. Table 5 present the results of our tests.

Table 5 – Comparison of the neighbor reduction strategies for SRITabu with 500 iterations

	Original SRITabu	Neighbor reduction strategy					
		Random			Distance		
% of reduction	0	25	50	75	15	25	35
Average deviation from the best-known solution	10.88%	11.68%	11.57%	12.16%	18.48%	10.36%	10.57%
Average time (sec)	29	23	16	9	18	21	23
Number of best solutions	0	0	0	0	0	0	0

The first column in the table presents the average deviation for the original SRITabu after 500 iterations. Columns 2, 3 and 4 contain the results obtained with the *Random* reduction strategy that eliminates 25, 50 and 75% of the neighbors. The best quality result for this strategy was obtained with a random 50% reduction, producing an average deviation of 11.53%. This result is close to that of the original SRITabu, which produced an average deviation of 10.83%, but it is much faster, with an average of 16 seconds compared to 29.

Columns 5, 6 and 7 contain the results using the *Distance* reduction strategy. The first test discarded all neighbors that were not closer than 15% of the longest distance on the map (column 5), the second, those that were not closer than 25% of the longest distance (column 6), and the last, those that were not closer than 35% (column 7). Overly, restricting the distance produced a significant deterioration in solution quality: the deviation obtained with distances closer than 15% is 18.48%, while 10.36% was obtained with distances closer than 25%. Both the 25% and 35% distance reduction strategies produced better results than the SRITabu by itself and were also faster, although the distance reduction strategy is, on average, slower than the random strategy.

The next subsection presents the results of our tests using the best parameter combination for each neighbor reduction strategy: "closer than 25%" for the distance strategy and "50%" for the random reduction strategy.

4.3.3. VRPPDC heuristic: SRITabu results

Tables 6 and 7 show the average results obtained by the SRITabu heuristic over the 100 instances, using 500 to 100,000 iterations. The tables are separated horizontally into three main sections: the results obtained with the SRITabu heuristic only, the results obtained with the "closer than 25%" distance reduction strategy, and then those obtained with the "50%" random reduction strategy. Each horizontal section shows the worst, best and average deviation from the best-known solution, the average time in seconds, the number of best solutions found, and the number of solutions found that deviated less than 0.5% from the best solutions. Table 6 uses the parameters of solution procedure A, while Table 7 employs the parameters of solution procedure B.

The best results obtained by SRITabu heuristic with procedure A are shown in the "100,000 iterations" column of Table 6. For the SRITabu only (i.e., no neighbor reduction strategy), the average deviation is 2.50%, and the solution was obtained in an average of 5,725 seconds. This "no strategy" method produced six best solutions and 24 results with less than 0.5% of deviation. The distance strategy, on the other hand, generated an average deviation of 2.62% and used half the computing time (i.e., an average of 2,719 seconds). This strategy produced four best solutions and 11 results with less than 0.5% of deviation. However, the random strategy generated the best overall

results. With only 10,000 iterations, this strategy yielded a 2.51% deviation, a result similar to those obtained by the two others methods with 100,000 iterations. Moreover, this strategy produced these results much faster, within only 300 seconds on average. With 100,000 iterations, the random strategy obtained the lowest average deviation (1.12%) almost as fast as the distance strategy obtained a much higher average deviation (i.e., an average of 2,879 seconds) and obtained four best solutions and 29 solutions with less than 0.5% of deviation. As these results show, the random reduction strategy halved both the average deviation and the computing time with respect to the original SRITabu application, using procedure A. From an empirical point of view, these results are significant.

Table 6 – Results of the SRITabu heuristic with solution procedure A

		Iterations	500	1,000	2,000	5,000	10,000	15,000	20,000	50,000	100,000
no strategy	Deviation	Worst	18.11%	17.03%	17.03%	17.03%	17.03%	17.03%	17.03%	17.03%	17.03%
		Best	4.65%	2.13%	0.33%	0.14%	0.00%	0.00%	0.00%	0.00%	0.00%
		Average	10.88%	7.36%	5.24%	3.73%	3.18%	2.96%	2.83%	2.56%	2.50%
	Time	Average (sec)	29	57	115	290	579	871	1,163	2,892	5,725
	Number of solutions	Best	0	0	0	0	1	1	1	5	6
		Under 0.5%	0	0	1	2	5	7	11	21	24
distance 25%	Deviation	Worst	18.69%	15.19%	14.10%	14.10%	14.10%	14.10%	14.10%	14.10%	14.10%
		Best	4.36%	3.53%	1.86%	0.80%	0.05%	0.00%	0.00%	0.00%	0.00%
		Average	10.36%	7.29%	5.28%	3.95%	3.42%	3.22%	3.08%	2.75%	2.62%
	Time	Average (sec)	21	39	72	162	304	444	582	1,385	2,719
	Number of solutions	Best	0	0	0	0	0	1	1	4	4
		Under 0.5%	0	0	0	0	1	2	3	10	11
random 50%	Deviation	Worst	20.34%	16.09%	10.37%	8.45%	8.11%	7.33%	7.33%	7.12%	7.12%
		Best	5.94%	2.87%	1.29%	0.13%	0.13%	0.13%	0.13%	0.00%	0.00%
		Average	11.57%	7.60%	5.24%	3.25%	2.51%	2.14%	1.88%	1.40%	1.12%
	Time	Average (sec)	16	32	65	153	300	443	589	1,475	2,879
	Number of solutions	Best	0	0	0	0	0	0	0	2	4
		Under 0.5%	0	0	0	2	3	4	4	11	29

Table 7 presents the results for solution procedure B. Using this strategy, almost no improvement was obtained over 50,000 iterations. For solution procedure B, the best quality results were obtained by the original SRITabu with an average deviation of 1.77%. This strategy generated six best solutions and 14 results with less than 0.5% of deviation. However, this strategy was also the most time consuming of the three, taking 2,702 seconds on average. In fact, both the random and distance reduction strategies obtained results faster, with an average of 1,433 and 1,420 seconds respectively. Unfortunately, these solutions have larger deviation of 2.42% and 2.59%, respectively.

With solution procedure B, only the original SRITabu performed better than the various scenarios using procedure A.

Table 7 – Results of the SRITabu heuristic with solution procedure B

		Iterations	500	1,000	2,000	5,000	10,000	15,000	20,000	50,000	100,000
no strategy	Deviation	Worst	36.81%	22.80%	22.80%	14.39%	9.15%	8.32%	8.32%	8.32%	8.32%
		Best	4.15%	2.92%	0.97%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		Average	11.90%	8.09%	5.50%	3.15%	2.33%	2.04%	1.88%	1.77%	1.77%
	Time	Average (sec)	30	59	117	282	550	817	1,095	2,702	5,165
	Number of solutions	Best	0	0	0	1	2	3	4	6	6
		Under 0.5%	0	0	0	3	6	9	12	14	14
distance 25%	Deviation	Worst	30.13%	16.80%	10.86%	7.70%	7.70%	7.70%	7.70%	7.70%	7.70%
		Best	4.84%	2.96%	1.89%	0.39%	0.53%	0.53%	0.53%	0.28%	0.28%
		Average	11.16%	7.82%	5.32%	3.49%	2.84%	2.70%	2.63%	2.59%	2.59%
	Time	Average (sec)	18	34	64	145	274	409	550	1,420	2,874
	Number of solutions	Best	0	0	0	0	0	0	0	0	0
		Under 0.5%	0	0	0	1	0	0	0	1	1
random 50%	Deviation	Worst	21.93%	16.52%	15.17%	10.07%	9.57%	9.57%	9.57%	9.57%	9.57%
		Best	5.42%	2.87%	1.37%	0.48%	0.48%	0.48%	0.48%	0.48%	0.48%
		Average	11.92%	8.03%	5.69%	3.58%	2.70%	2.55%	2.53%	2.42%	2.42%
	Time	Average (sec)	16	31	60	144	285	427	572	1,433	2,800
	Number of solutions	Best	0	0	0	0	0	0	0	0	0
		Under 0.5%	0	0	0	1	1	1	1	1	1

Examining both tables, strategy by strategy allows procedure A to be compared to B. In terms of quality, the "no strategy" results obtained with A at 100,000 iterations were obtained with B at 10,000 iterations and with only one tenth of the computing time. However, the same improvement is not true for the random and the distance strategies.

In addition, it appears that the average deviation always decreases progressively as the number of iterations increases. With solution procedure B, the average deviation decreased up to 50,000 iterations, and then no further improvement was obtained.

Overall, the best combination appears to be the random neighbor reduction strategy applied with solution procedure A. This is the only combination that produces average results under 2% within less than 600 seconds, and it leads to the best overall performance, with an average deviation of 1.12%.

5. Conclusion

We have introduced the split delivery Vehicle Routing Problem with Production and Demand Calendars, a rich variant of the VRP which, to our knowledge, has never previously been studied. The problem was modeled as a mixed integer program and

solved by means of a tabu search heuristic with split deliveries. Our results demonstrate that it is possible to obtain an excellent solution within a reasonable amount of time. We have also introduced several neighbor reduction strategies which were proved efficient not only at reducing computing time but also at improving the overall solution quality.

Acknowledgment

This work was partially supported by the Canadian Natural Sciences and Engineering Research Council under grants OPG0036509, 0039682-05 and OPG0172633. This support is gratefully acknowledged.

References

1. Archetti C., Savelsbergh M.W.P. & Speranza M.G. "To split or not to split: That is the question." *Transportation Research Part E,* 44, 2008, 114–123.
2. Archetti C., Speranza M.G. & Hertz A. "A tabu search algorithm for the split delivery vehicle routing problem." *Transportation Science*, 40, 2006, 64–73.
3. Bolduc M.-C., Renaud J. & Boctor F.F. "A heuristic for the routing and carrier selection problem." *European Journal of Operational Research*, 183, 2007, 926–932.
4. Bolduc M.-C., Renaud J., Boctor F.F. & Laporte G. "A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers." *Journal of the Operational Research Society*, 2006, to appear.
5. Boudia M., Louly M.A.O. & Prins C. "A reactive GRASP and path relinking for a combined production–distribution problem." *Computers & Operations Research*, 34, 2007, 3402–3419.
6. Cordeau J.-F., Gendreau M. & Laporte G. "A tabu search heuristic for periodic and multi-depot vehicle routing problem." *Networks*, 30, 1997, 105–119.
7. Fumero F. & Vercellis C. "Synchronized development of production inventory, and distribution schedules." *Transportation Science*, 33, 1999, 330–340.
8. Gendreau M., Hertz A. & Laporte G. "A tabu search heuristic for vehicle routing problem." *Management Science*, 40, 1994, 1276–1290.
9. Lin S. "Computer solutions of the traveling salesman problem." *The Bell System Technical Journal*, 1965, 2245–2269.
10. Miller C.E., Tucker A.W. & Zemlin R.A. "Integer programming formulation of traveling salesman problems." *Journal of the Association for Computing Machinery*, 7, 4, 1960, 326–329.
11. Osman I.H. "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem." *Annals of Operations Research*, 41, 1993, 421–451.