A Column Generation based Tactical Planning Method for Inventory Routing

Sophie Michel, François Vanderbeck

A Column Generation based Tactical Planning Method for Inventory Routing

S. Michel (1) and F. Vanderbeck (2)
(1) Université du Havre (michels@univ-lehavre.fr),
(2) Université Bordeaux 1 (fv@math.u-bordeaux1.fr),
(1-2) INRIA Bordeaux Sud-Ouest, team RealOpt
(http://www.inria.fr/recherche/equipes/realopt.en.html).

November 15, 2008

Abstract

Inventory routing problems combine the optimization of product deliveries (or pickups) with inventory control at customer sites. Our application concerns the planning of single product pickups over time; each site accumulates stock at a deterministic rate; the stock is emptied on each visit. At the tactical planning stage considered here, our objective is to minimize a surrogate measure of routing cost while achieving some form of regional clustering by partitioning the sites between the vehicles. The fleet size is given but can potentially be reduced. Planning consists in assigning customers to vehicles in each time period, but the routing, i.e., the actual sequence in which vehicles visit customers, is considered as an "operational" decision. The planning is due to be repeated over the time horizon with constrained periodicity. We develop a truncated branch-and-price-and-cut algorithm combined with rounding and local search heuristics that yields both primal solutions and dual bounds. On a large scale test problem coming from industry, we obtain a solution within 6.25% deviation from the optimal. A rough comparison between an operational routing resulting from our tactical solution and the industrial practice shows a 10% decrease in number of vehicles as well as in travel distance. The key to the success of the approach is the use of a state-space relaxation technique in formulating the master program to avoid the symmetry in time.

Keywords: Inventory Routing, Branch-and-Price-and-Cut, Primal Heuristic, Symmetry.

Introduction

In the mid-1980s, research work began on integrating inventory control with vehicle routing in an effort to better manage an important segment of the supply chain. The Inventory Routing Problem (IRP) consists in designing routes for deliveries or pickups that incorporate issues of inventory management at customer sites. Three decisions have to be made: (i) when to serve a customer; (ii) how much to deliver to (resp. pick up from) a customer on each service; and (iii) which delivery (resp. pickup) tasks are assigned to each vehicle (potentially including their sequencing to define routes). Many variants are discussed in the literature. Federgruen and Simchi-Levi (1995) and Campbell et al. (1998) discuss the first studies on IRP, while recent surveys are provided in Cordeau et al. (2007) and Bertazzi et al. (2008).

The application that motivates our study is a special case in which the inventory management model is simple when considered at the tactical decision planning level. A single product must be picked-up from customers who accumulate it in their stock at a given rate that is assumed to be known and deterministic. The pickup quantity is necessarily equal to the stock level at the time of the visit: technical constraints impose what is known as an "order-up-to-level" policy in inventory management. The inventory management policy aims at avoiding stock-capacity over-flow (by having sufficiently frequent pickups) while minimizing pickup costs (that increase with the frequency of pickups). In the deterministic context of tactical planning, the objective of avoiding stock-capacity over-flow translates into hard constraints: the bounded stock capacity imposes a maximum amount of time (expressed in number of periods) between two visits. Hence, there are no inventory management costs, but only stock transportation costs.

While decisions (i) and (ii) above result from the inventory model, decisions (iii) result from the routing model that consists traditionally in solving a capacitated vehicle routing problem (VRP) for each period of the planning horizon. In this study we view routing decisions as operational. In practice, the specific route is often decided by an experienced driver given information updates about the route network. Instead of the pure minimization of travel distance/time, we attempt to group customers from a compact geographical area into clusters: a cluster is a set of customers that are assigned to a given vehicle in a given time period. Our industrial partner raised this issue as more important than travel distance/time in our application where a clear assignment of "sectors" to vehicles is needed. Building routes of minimum length often leads to routes crossing each other or expanding over several districts (as we shall illustrate in Figure 1). To replace the routing cost, we use a measure of the cost of a cluster that is a compromise between the measure of dispersion around a cluster center used in facility location problems and the routing insertion cost used in heuristic methods for the VRP. This measure favours grouping customer sites that are close to one another, and it also acts as a surrogate measure for transportation costs.

Our tactical planning model takes yet another issue into account. To ease the real-life implementation of the plan, the schedule of customer visits must repeat itself over time, and the periodicity is constrained to be reasonably small. Finally, our tactical model allows us to consider reducing the fleet size that resulted from a decision taken at the strategic level. Determining what is the minimum fleet size that is required to satisfy all the planning requirements is far from being trivial (as we shall illustrate in Example 2).

Our tactical planning solution must serve as a target in operational planning so as to make the latter less myopic. The operational model will try to incorporate urgent pickup requirements into the tactical planning solution. Note that the routing solution associated with a cluster defined at the tactical planning level can be computed in a post optimization phase by solving a traveling salesman problem over the selected set of customer sites. The expected stochastic variation of customer accumulation rates can be accounted for in the tactical planning model by over-estimating stock filling rates, reserving buffer space in the vehicles and/or the customer stocks.

The model studied here has, to the best of our knowledge, not been explicitly considered in the literature, but many variants have been. Most of the existing approaches tend to make restrictive assumptions, such as the so-called "fixed partition policy" where one defines sets of customers that are systematically serviced together (see Bramel and Simchi-Levi (1995)). Alternatively they adopt a hierarchical optimization scheme where planning product deliveries or pickups is decided before routing (see Campbell and Savelsbergh (2004)). Most approaches are heuristics with no guarantee on the deviation from optimality and are specific to the problem variant (see Gaur and Fisher (2004)).

We have developed a truncated branch-and-price algorithm: periodic plans are generated for vehicles by solving a multiple choice knapsack subproblem; the global planning of customer visits is generated by solving a master program. This exact optimisation approach is combined with rounding and local search heuristics to yield both primal solutions and dual bounds that allow us to estimate the deviation from optimality of our solution. We were confronted with the issue of symmetry in time that naturally arises in building a cyclic schedule (cyclic permutations along the time axis define alternative solutions). Central to our approach is a state-space relaxation idea that allows us to avoid this symmetry. Our algorithm provides solutions with reasonable deviation from optimality for large scale problems (260 customer sites, 60 time periods, 10 vehicles) coming from industry.

The paper is organized as follows. Section 1 describes our problem and specifies our assumptions. Section 2 includes a short review of the existing literature. In Section 3, we outline our decomposition approach. Then, the symmetry in time is eliminated by modelling an average behavior. In Section 4, we compare this aggregate time formulation to the discrete time formulation. Section 5 presents the specific features of our column generation approach to solve the master LP. Cutting planes and partial branching are used to improve the dual bounds as presented in Section 6. Our primal heuristics are presented in Section 7. Section 8 reports our results on an industrial test problem and compares them with current industrial practice. Finally, we conclude with directions for further research.

1 The tactical planning problem

The problem is to plan vehicle visits to customers over a discrete time horizon divided into time periods: $t = 1, \dots, T$. Each assignment of customers to a vehicle in a given time period defines what we call a *cluster*. The quantity that is picked up on a visit to the customer is the whole stock accumulated since the last visit. To a given cluster, we associate a *pickup pattern* that defines the quantities that are picked up at each customer site. The sum of these quantities cannot exceed the vehicle capacity. Customers are indexed by $i = 1, \dots, n$. For each of them, we know the customer site location (also indexed by i), the product accumulation rate per period, r_i, the maximum number of periods between two visits, $t_i^{\max}$, that results from their limited stock capacity, and the time required to pick up the customer stock, b_i, which in our application is independent of the quantity that is collected. Let N be the set of relevant sites, including customer sites numbered 1 to n, a depot indexed by 0 from which vehicles start, and a dumping site, indexed by $n+1$, where vehicles are unloaded. Let $N' = N \backslash \{0, n+1\}$ be its restriction to true customer sites.

We assume a given fleet of V identical vehicles of capacity W, devoted to collecting the single product from the geographically dispersed customers. The number of available vehicles is assumed to be sufficiently large to cover the pickup requirements. However, our objective function includes the minimization of the maximum number of vehicles that is used in any given period, which we denote by $V_{\max}$ ($V_{\max} \leq V$). From the model point of view, putting the emphasis on minimizing $V_{\max}$ can allow us to generate solutions using fewer than the V available vehicles, which translates into significant savings. But this term is also included to help the convergence of our solution method: in our primal heuristic having an important fixed cost for using an extra vehicle gives an incentive to fill each vehicle and to spread the workload evenly among periods.

The time horizon that we consider is infinite, but we search for a periodic solution. Each cluster assignment will be repeated over time every p time periods. We restrict the solution space by imposing that cluster periodicities, p, are selected from a restricted set P. This implies a bound, T, on the time horizon beyond which the solution is repeated.

T is the least common multiple (LCM) of the periodicities in P. For our study, we take $P = \{1, 2, 3, 4, 5, 6\}$ and, hence, $T = 60$. For each cluster and associated pickup pattern, we must select its periodicity $p \in P$ and its first occurrence, i.e., its starting date, $s \leq p$. Then, the solution is H periodic where H is the LCM of the periodicities used in the solution. T acts as the maximal length of the regeneration cycle, i.e. $H \leq T$. Hence, T can be seen as the finite time horizon that results from the restriction on the periodicities.

A *vehicle task* is defined by selecting the cluster of customers that is assigned to the vehicle, the associated pickup pattern that determines the quantities that are collected by the vehicle, a periodicity p with which this vehicle assignment will be reproduced, and its first occurrence, $s \leq p$. A complete plan consists in an assignment of tasks to vehicles that ensures that the customer stocks produced in each period, $t = 1 \ldots, T$, are picked up in some task (see Example 2 below). Beyond the minimization of the fleet size, we attempt "to regionalize" vehicle tasks by defining a cluster cost that estimates its regional compactness. Although the exact routing of vehicles is considered an operational issue, we compute an estimate of travel time required to visit the customers of a cluster starting from the depot and ending at the dumping site. This estimate is intentionally biased towards cluster centered around a seed point, denoted k, so as to achieve the desired regional compactness. Let $\{d_{ij}\}_{(ij)\in N\times N}$ denote the shortest travel times between sites; the matrix is assumed to be symmetric. A cluster, $S \subset N'$, is built around a cluster center, $k \in S$. We define its cost as follows. It is initialized with a setup cost defined as the length of a shortest path from the depot to the dumping site passing by the seed k, plus the seed collect time, i.e., the fixed cluster cost is

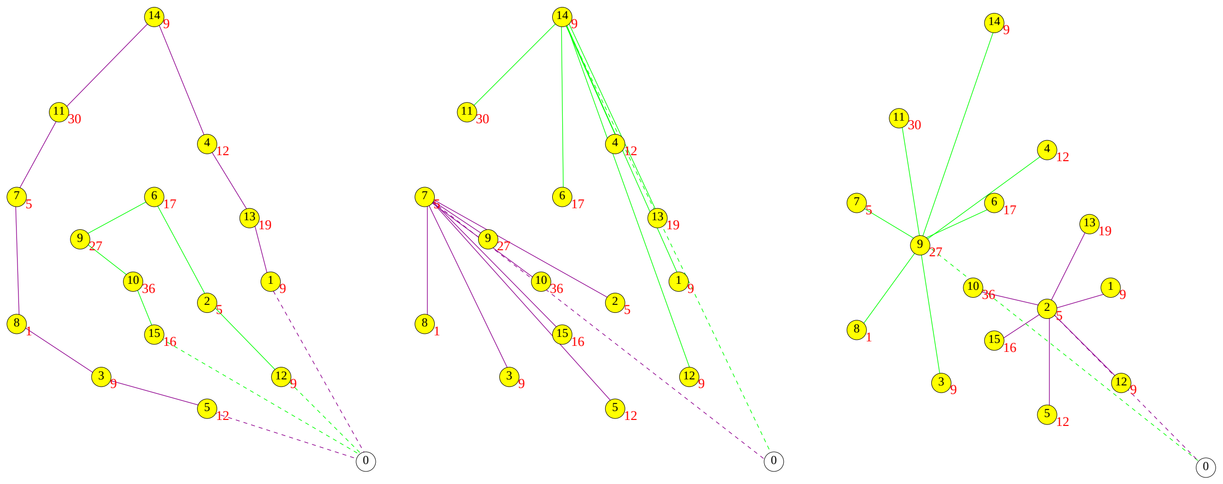
$$f_k = d_{0,k} + d_{k,n+1} + b_k \ ,$$

where $d_{0,k}$ and $d_{k,n+1}$ are the travel times between the depot 0 and site k and from site k to the dumping site $n+1$. Then, the connection cost for site i to the seed k is a measure of the insertion cost of site i in the path depot-seed-dumping site plus its collect time, i.e.,

$$c_{ik} = d_{i,k} + min\{d_{0,i} - d_{0,k}, d_{i,n+1} - d_{k,n+1}\} + b_i \ .$$

Example 1

Let us illustrate how this cost structure favors the grouping of customers that are geographically close to one another, while keeping sight of the resulting routing cost. In Figure 1, we compare three approaches for creating clusters: 1) the traditional VRP solution, 2) the clusters optimized with our cost structure, 3) the clusters obtained by minimizing distance to a center (forming stars) as done for facility location problems. The example concerns one period, a set of 15 customers whose demands are indicated on the side, and 2 vehicles of capacity 112. Beyond the pictorial customer groupings illustrated by Figure 1, it is interesting to compare the costs:



Comparing 3 approaches to clustering: standard routing (on the left), our clustering approach (in the center) and stars used in facility location models (on the right)

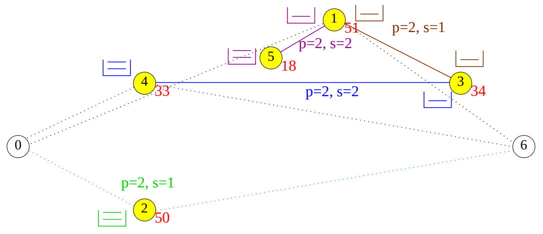
	routing solution	*clustering solution*	*facility location solution*
routing cost	**190.9**	*214.1*	*184.5*
our cluster cost	*196.8*	**200.4**	*147.0*
facility location cost	*198.8*	*208.7*	**135.5**

where the number in bold was the optimized objective, while the others are computed a posteriori or result from post-optimisation.

Example 2

Now consider a planning over $T = 6$ *periods with* $P = \{1, 2, 3, 6\}$. *Assume the instance has 5 customers with* $r = (51, 50, 34, 33, 18)$ *and* $t^{\max} = (1, 2, 2, 3, 5)$. *The vehicle capacity is* $W = 100$. *A plan consisting of 4 vehicle tasks is given in Figure 2. The figure on the left illustrates the vehicle tasks. Task 1 is to visit cluster* $\{1, 3\}$ *and to pick up* $(r_1 + r_3)$ *units. Task 2 is to visit cluster* $\{2\}$ *and to pick up* $(2\, r_2)$ *units. Task 3 is to visit cluster* $\{1, 5\}$, *and to pick up* $(r_1 + 2\, r_5)$ *units. Task 4 is to visit cluster* $\{3, 4\}$ *and to pick up* $(r_3 + 2\, r_4)$ *units. All tasks are two periodic, the first two start in period 1, the last two start in period 2. Both tasks 1 and 3 are performed by a first vehicle, while tasks 2 and 4 are performed by a second vehicle. The length of the regeneration cycle is* $H = 2$. *The table on the right illustrates the associated planning. For each customer, the period where there is a vehicle visit is marked by a* ■ *sign, and a* **v** *sign marks the period for which the associated stock production is collected in a following visit. This example illustrates how the combination of customer requests on a multi-period plan allows us to decrease the fleet size compared to a single period model. Indeed, the minimum number of vehicles required for a single period model (this is the solution of a bin packing problem with item size given by vector* r *and bin capacity* W*) is 3, while our plan uses 2 vehicles.*

A compact formulation of the problem can be derived in terms of the following vari-



A

solution in terms of tasks

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
$i=1$	■	■	■	■	■	■
$i=2$	■	v	■	v	■	v
$i=3$	■	■	■	■	■	■
$i=4$	v	■	v	■	v	■
$i=5$	v	■	v	■	v	■

ables: $x_{i\ell vps} = 1$ *iff* at customer site i, a quantity equal to a stock accumulation over $\ell \leq t_i^{\max}$ periods is collected by vehicle v that performs a task of periodicity $p \in P$ that starts in $s \leq p$; $y_{vps} = 1$ *iff* vehicle v is used for a task of periodicity $p \in P$ that starts in $s \leq p$; $z_{ikvps} = 1$ *iff* customer i is visited in a cluster of seed $k \in N'$ assigned to vehicle v with periodicity $p \in P$ and starting date $s \leq p$ (with $z_{kkvps} = 1$ *iff* the site k is the seed of vehicle v with periodicity $p \in P$ and starting date $s \leq p$); and $V_{\max}$ is the maximum number of vehicles that are used in any period. For notational convenience, an indicator vector of size T can be pre-computed for each pair (p,s) that marks the periods $t \in \{s, s+p, s+2\,p, \ldots\}$ in which a task indexed (p,s) should require a vehicle:

$$\delta_t^{ps} = \begin{cases} 1 & \text{if } \exists m \in N \text{ such that } s + m * p = t, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Similarly, we pre-compute an indicator vector of size T for each triplet (ℓ, p, s) saying whether the stock production of period t is collected by a task indexed by (p,s), when ℓ periods worth of stock is collected (such task would pick up the stock accumulated in periods $t \in \{s-\ell+1, \ldots, s-1, s, s-\ell+1+p, \ldots, s-1+p, s+p, \ldots\}$):

$$\theta_t^{\ell ps} = \begin{cases} 1 & \text{if } \exists m \in N \text{ and } \tau \in \{0, \ldots, \ell-1\} \text{ such that } s + m * p - \tau = t, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The compact formulation is:

$$\min \quad V_{\max} \;+\; \alpha \sum_{v,p,s} \frac{1}{p} \Big(\sum_k f_k z_{kkvps} + \sum_{i,k:i\neq k} c_{ik} z_{ikvps} \Big) \tag{3}$$

$$\sum_{\ell,v,p,s} \theta_t^{\ell ps}\, x_{i\ell vps} \;=\; 1 \qquad \forall i \in N', t = 1, \ldots, T \tag{4}$$

$$\sum_{k \in N'} z_{ikvps} \;=\; \sum_{\ell} x_{i\ell vps} \quad \forall i \in N', v, p, s \tag{5}$$

$$z_{ikvps} \;\leq\; z_{kkvps} \qquad \forall i \in N', k \in N', v, p, s \tag{6}$$

$$\sum_{i \in N', \ell} \ell\, r_i\, x_{i\ell vps} \;\leq\; W y_{vps} \qquad \forall v, p, s \tag{7}$$

$$\sum_{v,p,s} \delta_t^{ps}\, y_{vps} \;\leq\; V_{\max} \qquad \forall t = 1, \ldots, T \tag{8}$$

$$x_{i\ell vps} \;\in\; \{0,1\} \qquad \forall i, \ell, v, p, s \tag{9}$$

$$y_{vps} \;\in\; \{0,1\} \qquad \forall v, p, s \tag{10}$$

$$z_{ikvps} \;\in\; \{0,1\} \qquad \forall k, i, v, p, s \tag{11}$$

$$V \geq V_{\max} \;\in\; N \tag{12}$$

Constraints (4) ensure that, for each site, the stock produced in each period is collected. Constraints (5) enforce the relation between variables x and z. Constraints (6) define cluster centers. Constraints (7) enforce the bound on the vehicle capacity. Constraints (8) define $V_{\max}$. The objective (3) is to minimize the maximum number of vehicles used in any period plus the average cluster cost per period weighted by a coefficient $\alpha \geq 0$. The parameter α is typically chosen sufficiently small to model an hierarchical optimization where minimizing the number of vehicles is the primary goal.

2 Literature review

To clarify the position of our application in the diversity of variants of the Inventory Routing Problem (IRP), we classify previous studies based on basic criteria: the number of products, the length of the time horizon, the inventory policy, and whether demands are deterministic or not. Our presentation is focused on studies of variants of the inventory routing problem related to our application. We also provide a brief overview of the solution approaches that have been used. A more extensive review can be found in Michel (2006).

Here, a single product is transported, as in most of the problems encountered in the literature. In the multiple products case, one can often return to the single product situation: either one can aggregate products in a single type with an average consumption rate (see Gaur and Fisher (2004)), or decouple the problem into independent product problems (as in Campbell et al. (2002), Witucki et al. (1997)), or consider non distinctive products (as in Bell et al. (1983)).

At the tactical level where one minimizes the average costs on the long term, it is standard to consider periodic solution on an infinite time horizon. However, this can lead to an untractable search space if the periodicity of the solution is not restricted, since the regeneration cycle can be huge. Consider for instance a vehicle with unlimited capacity and 12 customers $i = 1, \ldots, 12$, where i must be visited every i periods. If a route visits all 12 customers in period 1, it will be re-used only in period $27720 = \mathrm{LCM}(\{1, \ldots, 12\})$. To overcome this drawback, one often restricts the solution space using restrictive replenishment strategies as for example: (*i*) direct shipping: a tour delivers a single customer; or (*ii*) zero stock policy: a customer is replenished if and only if his stock is down to zero (with this policy, however, the structure of the solution space remains too complex to permit a reasonable search, hence the following restriction is more common); (*iii*) the fixed partition policy of Bramel and Simchi-Levi (1995): the set of customers is partitioned into disjoint sets and each set is served separately, i.e., whenever a customer of the set is served, all the customers in that set must be served. Some extensions of the fixed partition policy are proposed in the literature. Let us mention that the order-up-to-level policy used in our application is also assumed in Bertazzi et al. (2002), Dror and Ball (1987), Webb and Larson (1995) (on each visit, the delivered quantity fills the stock up to capacity). Then, the delivered quantities are defined by the schedule and are not decision variables.

Using a deterministic model as ours is either motivated by the desire to simplify the problem, or by the assumption that variation in consumption rate (or filling rate) are small, or it results from having taken into account the stochastic character by way of safety threshold: in Webb and Larson (1995) a maximum interval between two visits is defined for each customer such that the probability to face a stock-out does not exceed a given threshold; a safety stock is considered in Custódio and Oliveira (2001), or a buffer space can be reserved in trucks as in Gaur and Fisher (2004).

Although none of the previous studies deals with the same model as ours under the same assumptions, the paper of Webb and Larson (1995) is probably the closest to our assumptions. They deal with a tactical planning problem where the stock management policy is an order-up-to-level policy assuming deterministic consumption rate. They minimize the average number of used vehicles. They partition their customers in subsets and define for each partition a set of routes that are repeated periodically. They show that this fixed partition policy combined with order-up-to-level policy is very restrictive. Their heuristic uses the concept of Clarke Wright savings. In our case, the customers are not partitioned a priori and our approach is based on an exact method.

It is also interesting to consider the size of the problems that have been solved. Our industrial problem involves 260 customers and 60 time periods, which results in close to 6000 visits that need to be scheduled in our planning. The difficulty of the vehicle assignment subproblem is proportional to the number of visits in a vehicle route, which is around 10 in our case. Compared to that, Christiansen and Nygreen (1998) considered a maximum of 15 customers in an ammonia distribution problem with an horizon of 30 time periods. Golden et al. (1984) schedule 3000 visits in a liquid propane distribution problem, but their time horizon is limited to 1 period. In Gaur and Fisher (2004), 207 supermarkets are considered, but vehicle tasks visit only 2 sites on average.

Consider the solution approaches that have been used to tackle these complex problems. Most are heuristics with no guarantee on the deviation from optimality and are specific to the problem variant. Exact approaches have been used to solve very small problems: Christiansen and Nygreen (1998) and Christiansen (1999) use a Branch-and-Price algorithm. Archetti et al. (2007) use a Branch-and-Cut algorithm. The existing heuristic approaches tend to adopt a hierarchical optimization scheme where planning is decided before routing (see Campbell and Savelsbergh (2004)). With the fixed partition policy, the choice of clusters also defines the routes and the planning.

Most methods somehow decomposes the original problem into the inventory management problem the one hand and a routing problem on the other hand. An explicit Dantzig-Wolfe decomposition is used by Witucki et al. (1997) and by Christiansen and Nygreen (1998) and Christiansen (1999). Bell et al. (1983) and Chien et al. (1989) base their heuristics on a Lagrangian relaxation approach that decomposes the problem into knapsack subproblems.

3 A Dantzig-Wolfe decomposition approach

In the line of the previous literature, a decomposition approach seems natural for our tactical planning problem. We apply the Dantzig-Wolfe reformulation principle (as presented in Vanderbeck and Savelsbergh (2006)) to compact formulation (3-8), dualizing the planning constraints (4) and the fleet size constraints (8). The problem decomposes into a master program taking care of the inventory planning issues on one hand and subproblems defining vehicle tasks on the other hand.

Let $\{(c^q, s^q, p^q, x^q)\}_{q\in Q}$ be the enumerated set of vehicle tasks, indexed by q. Each task is defined by its cost, c^q, its starting date, s^q, its periodicity, p^q, and a pickup pattern, x^q, where the picked up quantities are expressed in number of periods worth of accumulated stock. An indicator vector δ^q saying in which period a vehicle is required and an indicator matrix θ^q saying which customer stock production is picked up can be generated similarly to (1) and (2): $\delta^q_t = 1$ *iff* $\exists m \in N$ such that $s^q + m * p^q = t$ and $\theta^q_{it} = 1$ *iff* $\exists m \in N$ and $\tau \in \{0, \ldots, x^q_i - 1\}$ such that $s^q + m * p^q - \tau = t$. Thus, the tactical planning problem can be reformulated as:

$$Z_{IP}^{DW} = \min V_{\max} + \alpha \sum_{q\in Q} \frac{c^q}{p^q} \lambda_q \tag{13}$$

$$\sum_q \theta^q_{it}\lambda_q \geq 1 \qquad \forall i \in N',\ t = 1, \ldots, T \tag{14}$$

$$\sum_q \delta^q_t \lambda_q \leq V_{\max} \qquad \forall t = 1, \ldots, T \tag{15}$$

$$\lambda_q \in \{0, 1\} \qquad \forall\, q \in Q \tag{16}$$

$$V \geq V_{\max} \in N \tag{17}$$

where $\lambda_q = 1$ *iff* task q is selected, while $V_{\max}$ is the maximum number of vehicles used in a period. Z_{LP}^{DW} denotes the associated LP relaxation value. Note that the relaxation of (4) into "$\geq$ constraints" (14) does not induce any cost saving. The variables, λ_q, and associated columns are generated dynamically in the course of the LP optimization procedure using a column generation approach.

The column generation subproblem reduces to a variant of the multiple choice knapsack problem (noted MCKP), once we fix a starting date, s, a periodicity, p, and a cluster center (or seed), k. Let $x_{i\ell} = 1$ *iff* customer i is visited and the quantity that is collected is its production of ℓ periods. The associated profit, denoted by $g_{i\ell}$, depends on the connection cost to the seed k and on the dual values of constraints (14) which is denoted by π_{it}:

$$g_{i\ell} = \sum_{\tau=0}^{\ell-1} \sum_{t:\frac{t+\tau-s}{p}\in\mathbb{Z}} \pi_{it} - \frac{\alpha}{p} c_{ik} \ .$$

For each triplet (s, p, k), the MCKP takes the form:

$$\max \sum_{i,\ell} g_{i\ell} x_{i\ell} \tag{18}$$

$$\sum_{\ell} x_{k\ell} = 1 \tag{19}$$

$$\sum_{\ell} x_{i\ell} \leq 1 \qquad \forall\, i \neq k \tag{20}$$

$$\sum_{i,\ell} \ell\, r_i\, x_{i\ell} \leq W \tag{21}$$

$$x_{i\ell} \in \{0,1\} \qquad \forall\, i, \ell. \tag{22}$$

Constraint (19) determines the collected quantity for the seed k, constraints (20) enforce the selection of at most one collected quantity for the other customers, and constraint (21) is the knapsack constraint enforcing a bound on the vehicle load.

Dantzig-Wolfe reformulation (13-17) avoids the symmetry in the vehicle indexing v that was present in compact formulation (3-8), but it still suffers from a symmetry in the time period indexing t: equivalent solutions can be defined that differ only by a permutation in the choice of starting dates. In the search for integer solutions, the algorithm might enumerate these equivalent solutions. Moreover, this symmetry yields instability in the dual solutions π and σ associated respectively to constraints (14) and (15) for every period t, which is harmful for the convergence of the column generation procedure used to compute dual bounds. To avoid these drawbacks, we aggregate periods and model an average behavior. Technically speaking, we implement a state-space relaxation in the space of the columns: aggregating all columns that differ only by their starting dates, s^q, we project our column space as follows:

$$\{(c^q, s^q, p^q, x^q)\}_{q\in Q} \quad \rightarrow \quad \{(c^r, p^r, x^r)\}_{r\in R}.$$

To each column, $r \in R$, is associated a set, $Q(r)$, of columns, $q \in Q$, such that r is the projection of q:

$$Q(r) = \{q \in Q : c^q = c^r, p^q = p^r, x^q = x^r, s^q \in \{1, \ldots, p^r\}\} .$$

While the former formulation is referred to as the *discrete time master* problem, the reformulation obtained after performing this mapping is called the *aggregate master* problem. It is obtained by summing constraints (14) and (15) over t and then dividing the aggregate constraint by T. It takes the form:

$$\begin{aligned}
Z_{IP}^{ADW} = \min \quad V_{\text{aver}} & + \alpha \sum_{r \in R} \frac{c^r}{p^r} \lambda_r & & \quad (23)\\
\sum_{r \in R, \ell} \frac{\ell}{p^r} x_{i\ell}^r \lambda_r & \geq 1 & \forall\, i \in N' & \quad (24)\\
\sum_{r \in R} \frac{1}{p^r} \lambda_r & \leq V_{\text{aver}} & & \quad (25)\\
\lambda_r & \in N & \forall\, r \in R & \quad (26)\\
V \geq V_{\text{aver}} & \in N, & & \quad (27)
\end{aligned}$$

where λ_r represents the total number of times task r is assigned to a vehicle,

$$\lambda_r = \sum_{q \in Q(r)} \lambda_q \; , \qquad (28)$$

and V_{aver} is the average number of vehicles used per period rounded-up to the next integer. V_{aver} defines a surrogate measure of $V_{\max}$: given a solution $\{\lambda_q\}_{q \in Q}$ of (13-17), its projection by (28) into a solution $\{\lambda_r\}_{r \in R}$ of (23-27) is such that

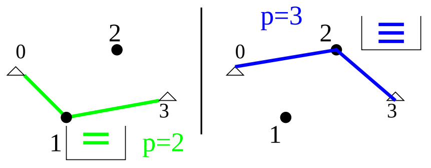
$$V_{\text{aver}} = \lceil \sum_{r \in R} \frac{1}{p^r} \lambda_r \rceil = \lceil \sum_{r, q \in Q(r)} \frac{\sum_q \delta_t^q}{T} \lambda_q \rceil \leq V_{\max} = \max_t \{\sum_q \delta_t^q \lambda_q\} .$$

Constraints (14) are replaced by (24): each task covers a fraction of the aggregate demand. Constraints (15) are replaced by (25): each task uses a fraction of a vehicle (the same fraction in each period, in this average model). The dual values, π_i, associated to constraints (24) represent now the average collect cost for customer i. Z_{LP}^{ADW} denotes the associated LP relaxation value. In a column generation solution approach of the aggregate master problem LP, the pricing core subproblem takes the form (18-22) as for the discrete master formulation, but one does not have to enumerate on each possible starting date anymore, as dual reward are time independent.

4 Comparing discrete and aggregate master programs

We show that the discrete and the aggregate master programs have the same optimal LP solution, but that the solution of the aggregate master problem by column generation

Figure 1: Two tasks: one with $s = 1$ and $p = 2$, the other with $s = 2$ and $p = 3$



aggreg. sol. λ_r	discrete sol. λ_q		t1	t2	t3	t4	t5	t6
$\lambda_A = 1$	$\lambda_{A_1} = \frac{1}{2}$	$s_{A_1} = 1$	■	v	■	v	■	v
	$\lambda_{A_2} = \frac{1}{2}$	$s_{A_2} = 2$	v	■	v	■	v	■
$\lambda_B = 1$	$\lambda_{B_1} = \frac{1}{3}$	$s_{B_1} = 1$	■	v	v	■	v	v
	$\lambda_{B_2} = \frac{1}{3}$	$s_{B_2} = 2$	v	■	v	v	■	v
	$\lambda_{B_3} = \frac{1}{3}$	$s_{B_3} = 3$	v	v	■	v	v	■

is much faster (see Table 1). Hence, we use the aggregate master problem to compute dual bounds. However, from an integer solution point of view, both formulations are not equivalent: the aggregate formulation is a relaxation of the problem. Hence, the discrete time formulation remains useful for computing primal bounds through heuristics.

Let us first illustrate the mapping between discrete and aggregate master solutions:

Example 3

Consider an instance with 2 customers, and an aggregate master solution involving two tasks. Task A is to collect the production of 2 *periods from customer* 1 *every* 2 *periods. Task B is to collect the production of* 3 *periods from customer* 2 *every* 3 *periods: see Figure 3. In short notation, we have*

- *Task A:* $0 - 1(2) - 3$ *with* $p_A = 2$,
- *Task B:* $0 - 2(3) - 3$ *with* $p_B = 3$,

where the digit in the parenthesis is the collected quantity. The LP solutions of both formulations are represented in the table given along Figure 3. For the aggregate formulation, each task is selected once. In the discrete formulation, p discrete time columns are associated to each aggregate task, one for each starting date: the value λ_q *of each column is* $\frac{1}{p^q}$. *In the right part of the table, we indicate, along with the column value, the periods where the vehicle is used by a* ■ *sign and we mark by a* **v** *sign the period for which the associated customer stock production is collected. Observe that* $V_{aver} = 1 \geq \frac{1}{2} + \frac{1}{3}$; *thus 1 vehicle is used in the integer solution to the aggregate master problem. However, no matter how one chooses the starting dates in a discrete master integer solution, there is always one period where both tasks require a vehicle. Thus, there is no feasible integer solution to the discrete problem using only 1 vehicle.*

With the intuition of this example, it is clear that every aggregate LP solution, $\{\hat{\lambda}_r\}_r$, of (23-27) can be translated into a discrete time solution, $\{\hat{\lambda}_q\}_q$, to (13-17): the mapping

$$\hat{\lambda}_q = \frac{1}{p^r}\hat{\lambda}_r \quad \forall r, q \in Q(r) \tag{29}$$

defines a solution such that $\hat{V}max = \hat{V}aver$ and $\hat{Z}_{LP}^{DW} = \hat{Z}_{LP}^{ADW}$. Note that an alternative discrete LP solution $(\overline{\lambda}, \overline{V}max)$ other than the above symmetric mapping (where $\hat{V}max = \hat{V}aver$) could only yield $\overline{V}max \geq \hat{V}aver$, and therefore an increased cost value. Hence, restricting the discrete LP-solution space to time symmetric solutions is not suboptimal. The reverse mapping is given by (28); it yields $\hat{V}aver \leq \hat{V}max$ and therefore $\hat{Z}_{LP}^{ADW} \leq \hat{Z}_{LP}^{DW}$. From the above mappings, we can conclude that if the discrete master solution is LP optimal for (13-17), then we must have $\hat{V}aver = \hat{V}max$ (otherwise, use transformation (28) followed by (29) to reduce $\hat{V}max$). Thus, we have shown:

Proposition 1 *The LP optimal solution of the discrete master problem (13-17) can be transformed into an LP optimal solution of the aggregate master problem (23-27) with the same cost $Z_{LP}^{DW} = Z_{LP}^{ADW}$, and vice versa.*

To evaluate the comparative advantage of the aggregate master problem over solving the discrete master LP by column generation, we have performed comparative tests on industrial and randomly generated instances. In Table 1, instances named "IND8" and "IND27" are extractions from real-life industrial data with respectively 8 and 27 sites (results are averages on multiple such extractions). Instances named "RAND100" are random instances with 100 sites imitating the real problem for the order of magnitude of filling rates r_i's and maximal time interval between visits $t_i^{\max}$'s. The site coordinates are generated according to 3 schemes: urban, rural or mixed. According to a scheme, a region is cut up into squares with specified probability that a site falls in these squares. Once the square area to which a site belong has been randomly generated, its coordinates within the square are generated according to an uniform distribution. Travel times are then assumed to be proportional to Euclidean distances. (These random instances are available on "http://hal.inria.fr/inria-00169311/fr/".)

Table 1 reports the number of generated columns, "Col", and the overall time spent in computing the LP optimal solution, "Time"; entries are average over all instances in each class. All computational times are on a PC bi-pro. Xeon 3GHz, 2Go. A limit of 1 hour is set. Missing inputs correspond to the case where this time limit is reached.

Table 1: Comparing LP solution times

	discrete form.		aggregate form.	
Instance	Col	Time	Col	Time
IND8	71	1.75s	20	0.25s
IND27	-	>1h	123	5.00s
RAND100	-	>1h	701	3.52s

The two formulations are not equivalent from an integer solution point of view as illustrated in Example 3. Here is another illustration of the relaxation implicit to integer reformulation (23-27):

Example 4

Consider a partial solution involving 2 tasks that serve customer 1 with $t_1^{\max} = 3$*:*

- *task A:* $0 - \cdots - 1(3) - \cdots - n+1$ *with* $p_A = 6,$ *and*
- *task B:* $0 - \cdots - 1(2) - \cdots - n+1$ *with* $p_B = 4.$

The following table presents the vectors θ^A_{1t} *and* θ^B_{1t} *indicating what production of customer* 1 *is picked up for fixed starting dates (*$s_A = 5$ *and* $s_B = 2$ *respectively) – we make use of the same notation as above:*

	t1	*t2*	*t3*	*t4*	*t5*	*t6*	*t7*	*t8*	*t9*	*t10*	*t11*	*t12*
θ^A_{1t}			**v**	**v**	■				**v**	**v**	■	
θ^B_{1t}	**v**	■			**v**	■			**v**	■		

Assume the aggregate solution has $\lambda_A = \lambda_B = 1$*. Although these starting dates avoid conflicts for vehicle use, the production of customer* 1 *is not properly collected. In fact, there is no integer solution to the discrete model using these 2 tasks as any selection of starting dates would leave some stock collections unsatisfied while over-covering others.*

In conclusion, even though the aggregate integer solution can sometimes be translated into a feasible solution for the discrete formulation, this cannot be guaranteed as the aggregate formulation defines a relaxation of the problem. Hence, we work with the discrete time formulation when it comes to constructing primal integer solutions.

5 Solving the Aggregate Master LP

We briefly present the column generation procedure that we use to solve the aggregate master problem LP detailing only the application-specific features. We initialize the master formulation with artificial columns associated to constraints (24): artificial column i is defined by the unit vector e_i (the ith column of identity matrix) for $i = 1, \ldots, n$, augmented with a 0 coefficient in constraint (25). Its cost defines an initial upper bound on the dual value π_i, as it can be seen by writing the dual of the master LP (see Vanderbeck (2005)). Briant et al. (2008) shows the impact on the convergence of the column generation procedure of an intelligent initialization by computing good bound estimates on the dual value π_i. Hence, we run a dual heuristic to estimate π_i values, which we use to define the initial costs of the artificial columns. In solving the master LP, we perform a combined phase 1 (for finding a feasible *LP* solution) and phase 2 (for finding an optimal *LP* solution): if artificial variables remain in the LP optimal solution, their cost is multiplied by 1.8 and the column generation starts again. Thus, initial artificial column costs are only estimates of dual value bounds that are increased dynamically while the artificial column remains in the LP optimum solution. If the master problem is unfeasible, this procedure fails to eliminate artificial columns from the master solution. Hence, after several cost increases, we perform a pure phase 1, keeping only the artificial columns in the objective function.

The dual heuristic goes as follows. In an "ideal" solution,
a) a customer i would be collected at the lowest possible frequency: $per_i = max\{p \in P, p \leq t_i^{\max}\}$;
b) a vehicle would use its full capacity and would collect on average $a = \lceil \frac{W}{(\sum_i r_i\, t_i^{\max})/n} \rceil$ customers that are close neighbors, where n is the number of customers.
More realistically, we assume for our dual heuristic that a cluster contains sites which are more spaced out geographically: we assume that a site i can be connected to one of the $2\,a$ nearest neighbors. Thus, we estimate the contribution of customer i to the inter-site travel cost to be $\frac{\sum_{j \in N_i} d_{ij}}{2\,a}$ where N_i is the set of the $2\,a$ closest neighbors to customer i. To this cost, we must add the collect time, b_i, and an estimate of customer i contribution to the depot-to-dumping-site return trip: $\frac{d_{0,i}+d_{i,n+1}}{a}$. Considering that, at LP optimality, $\sum_i \pi_i^* = V_{\text{aver}}^* + \alpha \sum_r \frac{c^r}{p^r} \lambda_r^*$, the average collect cost of i, π_i should also include a fixed cost for vehicle use. The latter is derived by estimating V_{aver}^*: if we assume that tasks are ideally made of customers sharing the same periodicity, $p \in P$, and that vehicles are devoted to one periodicity group, we can derive that they would be $\frac{n(p)}{a}$ tasks of periodicity p, where $n(p)$ is the number of customers i with $per_i = p$. Then, $\frac{n(p)}{p\,a}$ vehicles would be required to cover the $n(p)$ customers i with $per_i = p$. Thus, on average, the fixed vehicle cost attributed to a customer can be estimated by $\frac{\sum_p (\frac{n(p)}{p\,a})}{n}$. In summary, our estimator of the contribution of customer i to the total cost is

$$\frac{\sum_p (\frac{n(p)}{p\,a})}{n} + \alpha\, \frac{b_i + \frac{d_{0,i}+d_{i,n+1}}{a} + \frac{\sum_{j \in N_i} d_{ij}}{2\,a}}{per_i}\,. \tag{30}$$

As these π_i estimates are optimistic, we use as initial values for the π_i upper bounds, the value given by (30) multiplied by a factor 1.2. We emphasize that our dual heuristic is just a mean to provide a warm start for the column generation approach. The algorithm remains exact whether or not these estimated bounds on π_i values are satisfied by the optimal dual solution since there are dynamically updated.

Once the master problem is initialized with the above artificial columns, a standard column generation procedure follows. In search for the smallest reduced cost columns, we feed the dual information π and σ to the pricing problem solver. The latter iterates on each periodicity p and possible seed k, solving the associated multiple choice knapsack problem (18-22) having set for each $\ell \leq \min\{t_k^{\max}, p\}$, $g_{i\ell} = \ell\pi_i - \alpha c_{ik}$ for $i \neq k$, while $g_{k\ell} = \ell\pi_k$. The oracle used to solve the multiple choice knapsack problem is the dynamic program of Pisinger (1995) (we had to adapt the code so as to deal with real value profits instead of integer values). The associated solution value is denoted $\zeta_{pk}(\pi)$. The reduced cost of the optimal column, r^*, is $\overline{c}(\pi, \sigma) = \min_{p \in P, k=1,\ldots,n} \frac{1}{p}(\sigma + \alpha f_k - \zeta_{pk}(\pi))$.

Some preprocessing is performed to speed up the pricing process. If $\pi_k = 0$, then site k cannot be a seed in an optimal solution. In (18-22), $x_{i\ell}$ can be set to zero if $\ell\pi_i \leq \alpha c_{ki}$. A cut off value on $\zeta_{pk}(\pi)$ is defined from the best reduced cost value encountered on previous pairs (k, p), denoted $\overline{c}_{\text{best}}$: we want $\frac{1}{p}(\sigma + \alpha f_k - \zeta_{pk}(\pi)) \leq \overline{c}_{\text{best}}$, thus $\sigma + \alpha f_k - p\,\overline{c}_{\text{best}}$ defines a lower bound on $\zeta_{pk}(\pi)$. In particular, if the knapsack problem LP relaxation does not satisfy this bound, the problem can be cutoff. In practice, the enumeration of the multiple choice knapsack subproblems stops as soon as a column with negative reduced cost is found. If no such column is found, the optimality is proved. In our numerical tests, we noted that stopping the enumeration of the pricing problem as soon as a column with negative reduced cost is found (instead of searching the best column at each iteration) divides the time by a factor 3.4. Hence, we always use this strategy and rule out the option of generating multiple columns at each iterations. To increase the chances to quickly generate good columns, we enumerate periodicities p from the largest down to the smallest (in the *LP* solution, columns with larger periodicities are more likely to be used). The seeds, k, are sorted by decreasing ratio of estimated profit (the best seed item coefficient in the pricing problem is $\max_\ell g_{k\ell} = \pi_k * \min\{p, t_k^{\max}\}$) over the estimated cluster cost (which is $\frac{d_{0,k} + d_{k,n+1} + \sum_{i \in N_k} c_{ik}}{2\,a+2}$). A post-optimization improves the returned column: the seed selection is re-optimized and the smallest periodicity $p = \max_i\{\ell : x_{i\ell} = 1\}$ is computed (it can be shown that the master LP solution can be restricted to columns verifying this property, i.e., $\exists i, x_{i\,p} = 1$).

6 Adding cuts and performing partial branching

Aggregate master LP dual bounds can be slightly improved using a cutting plane procedure and partial branching. We derive cuts from constraints (24) using a rounding procedure. To illustrate the sort of fractional solutions that we aim to cut off, consider the following example.

Example 5
Assume a task of periodicity 6 has been generated that covers 5 periods worth of customer i's production, while $t_i^{\max} = 5$. This task alone covers $\frac{5}{6}$ of customer i's demand. To cover the average demand of customer i, the task is selected 1.2 times in the LP solution. An integer solution should use such task at least twice or it should cover the residual demand

with another task. A valid inequality to cut this LP solution is:

$$\sum_{r,\ell:\, \ell = p^r} x^r_{i\ell}\, \lambda_r + \frac{1}{2} \sum_{r,\ell:\, \ell \neq p^r} x^r_{i\ell}\, \lambda_r \geq 1,$$

saying that, to cover the stock production of a given customer i, one may use either one task which covers his whole production or one needs at least two tasks.

The cuts derived in Example 5 can be generalized into

$$\sum_{r,\ell:\, h\ell \,\mathit{mod}\, p^r = 0} \frac{\ell}{p^r}\, x^r_{i\ell}\, \lambda_r + \sum_{r,\ell:\, h\ell \,\mathit{mod}\, p^r \neq 0} (\left\lceil \frac{h\ell}{p^r} \right\rceil - \frac{h\ell}{p^r})\, x^r_{i\ell}\, \lambda_r \geq 1 \quad \forall i \in N',\ h = 1, \ldots, T-2\ . \tag{31}$$

These inequalities are in fact valid for all $h \in N$: they are obtained from (24) and the equality $\sum_r h\frac{\ell}{p} x^r_{i\ell}\lambda_r = h$ that derives from (24) in its equality form, by applying a super-additive function to a weighted sum of constraints (see Proposition 4.1, page 229, in Nemhauser and Wolsey (1988)). The super-additive function that we use, is:

$$F_\gamma; R \to R; F_\gamma(d) = \lfloor d \rfloor + \frac{(d - \lfloor -d \rfloor - \gamma)^+}{1-\gamma}\ ,$$

with parameter γ chosen such that $0 \leq \gamma = 1 - \epsilon < 1$ and $(.)^+ := \max\{0, .\}$. Its super-additivity is proved in Proposition 4.7, page 233, in Nemhauser and Wolsey (1988). Note that when $t_i^{\max} = 1$, (31) are dominated by (24) since, when $h\%p \neq 0$, $\lceil \frac{h}{p} \rceil - \frac{h}{p} \geq \frac{1}{p}$. Similarly, when $h = T$, (24) dominates (31), and when $h = T - 1$, (24) is equivalent to (31). Moreover, constraints (31) are $T-$periodic in h. Thus, we consider cuts (31) for integer h ranging from 1 to $T - 2$. As their number is polynomial, separation can be completed by enumeration. One can stop as soon as a violated cut is found. To have a better chance to find violated cut early in the process, the enumeration is carried in an order inspired by experimental observations: the inequalities with small values of h are more likely to be violated; so are the ones for customer i with a large value $t_i^{\max}$ but different from $p_{\max} = \text{argmax}\{p : p \in P\}$. After adding a cut, we return to the column generation procedure. The structure of the pricing problem does not change, as only the profits $g_{i\ell}$ are affected. In order to keep the feasibility of the master problem after adding cuts, a global artificial variable is used. Its cost is an estimation of solution cost (we set it equal to the dual heuristic solution cost).

The contribution of these cuts to dual bound improvement is limited as shown in our numerical experiments below. To further improve dual bounds, we perform a partial enumeration scheme: we branch only on variable V_{aver}. Given the structure of our objective that focuses on vehicles use, this branching has an important impact on the bound. Assume $V_{\text{aver}} = \beta \notin N$ in the root node aggregate master LP solution, we define two branches

$$\text{(Node 1)} \quad V_{\text{aver}} \leq \lfloor \beta \rfloor \qquad \text{or} \qquad V_{\text{aver}} \geq \lceil \beta \rceil \quad \text{(Node 2)}\ . \tag{32}$$

In node 1, the branching constraint $V_{\text{aver}} \leq \lfloor \beta \rfloor$ is very restrictive: this branch can often be proved infeasible. In node 2, the re-optimized V_{aver} value is rounded-up to the next integer compared to the root node solution and the aggregate master LP cost can sometime increase significantly.

To evaluate the impact of cuts and partial branching on the dual bound, we have carried on comparative tests on random and real-life based instances similar to those presented in Section 4. We have 5 instances extracted from our industrial data with 60 customers on average, this group is named "IND60", and 2 bigger instances with 172 and 157 customers, named "IND172" and "IND157". Moreover, we use 10 random instances with 100 customers named "RAND100". The number of instances is reported in parenthesis.

On this test bed, the dual bound improvements observed by adding cuts are small (less than 2%). However, cuts change the structure of the LP solution: in the LP solution before adding cuts, the tasks tend to be all 6-periodic, whereas tasks have periodicity of 1 up to 6 in the LP solution after adding cuts. In the dynamic cut generation procedure, on average only 7.63% of the valid inequalities (31) are added to the formulation. On the other hand, the improvement obtained through partial branching can get bigger (depending on the instance and, more specifically, on the fractional part of V_{aver}), it ranges from less than 1% up to more than 15%. To evaluate these bound improvements, we compute the gap to a primal solution (obtained as explained in Section 7). In Table 2, we present the gap (expressed in %) obtained at the root of the branch-and-price tree, "gap-root", the gap obtained after adding cuts, "gap-cut", and the gap obtained after partial branching, "gap-br", as well as combining the latter two, "gap-br-cut" (then, we call on the cutting plane procedure only at node 2 after rounding up V_{aver}). The root gap is improved by a 58.5% factor on average when using both cuts and partial branching.

Table 2: Dual bound improvements obtained from cutting planes and partial branching (measuring the gap to a primal solution in %).

Instance	gap-root	gap-cut	gap-br	gap-br-cut
RAND100 (10)	22.81	21.31	10.37	9.24
IND60 (5)	26.38	25.51	12.51	11.83
IND172 (1)	16.46	15.59	8.65	7.89
IND157 (1)	15.40	14.39	5.83	5.11
av. over 17 inst	**23.05**	**21.80**	**10.63**	**9.67**

We also attempt to evaluate the computational burden of adding cuts by comparing average computational times over the 17 instances. Standard column generation without cut takes 1.6 minutes on average, whereas with the cutting plane procedure computing times vary from 33 seconds to 41 minutes, taking 16 minutes on average. The bulk of the time (around $\frac{3}{4}$ of the time) is spent in the re-optimization of the master problem by column generation after each round of additional cuts. In order to improve the time spent in the cutting plane procedure (including re-optimization of the master problem), various strategies have been tested. We observe on our test bed that this average time is divided by 3 when we return the first violated cut found instead of the more violated cut (given the specific order in which we test cut violation). Furthermore, when we add 10 cuts during the same iteration (instead of one at the time), we divide again the time spent in the cutting plane procedure by 1.35. This time is further divided by 1.18 if we solve the pricing problem exactly only every 20 iterations.

When branching on V_{aver}, proving the infeasibility of Node 1 can be very time consuming too. To control this, we limit the number of artificial variable cost increases to 3, after which we perform a pure phase 1. In Table 3, we show how the computing time is shared between the different components of our dual bound computation. The overall time spent in computing our best dual bound (using cuts and partial branching) varies from 5 minutes for the smallest instances up to more than 2 hours for the largest. The columns of Table 3 indicate the average percentage of that time spent in the pricing problem, "PP", in solving the restricted master problem, "RM", in the cutting plane generation procedure (which includes separation and re-optimization), "CP" , and in the separation procedure alone, “Sep”. Then, we report on the percentage of that time spent in the different nodes: the root of the branch-and-price tree, "N0", the branch "N1" where V_{aver} is rounded down, and the branch "N2" where V_{aver} is rounded up and the cutting plane procedure is called. In the sequel we shall use the best dual bound obtained using cuts and partial branching to evaluate our primal solutions. However, we shall see that the performance of our primal heuristic is better when building it from the master LP solution without making use of the cutting plane procedure.

Table 3: Average time partition (in %)

PP	RM	CP	Sep	N0	N1	N2
10.84	8.14	73.49	4.14	5.87	16.17	77.91

7 Primal solutions to the tactical planning problem

Several types of primal heuristics can be derived in a column generation context. First, one can attempt to obtain an integer solution by solving the master problem restricted to the set of generated columns as an integer program (by Branch-and-Bound). Implementations of such approach vary mainly by the definition of the restricted column set $\overline{Q}$: it can contain some or all the columns generated during the LP optimization procedure (see Chabrier et al. (2002), Chabrier (2003), Ceselli et al. (2007)) or columns generated during other heuristic algorithms (see Taillard (1999), Schmid et al. (2007)). However, this restricted master (RM) solution method often fails because it can be impossible to construct a feasible integer solutions using the columns from the restricted column set $\overline{Q}$ that were generated to provide an LP solution. Other classical heuristics such as greedy, LP based rounding, or local search heuristics can be adapted to the context of a column generation approach. A greedy heuristic (GH) is a constructive method that iteratively adds a greedy selected column in the partial solution until feasibility is reached. The greedy criteria for selecting the next column to be added to the partial solution varies (see Perrot (2005), Belov and Scheithauer (2002), Ceselli et al. (2007) and Cimelière (2004)). A rounding heuristic (RH) is a depth search plunging into the branch-and-price tree: at each node, a branch is chosen heuristically by selecting the master variable with the fractional value that is closest to its rounded up value and rounding it up. One may choose (as we do) to fix the rounded down LP solution as a partial integer solution before any rounding up. After each heuristic branching, the residual master program is re-optimized: generating new columns in the process is an important feature for the success of the approach as it allows us to construct feasible solutions (see Belov and Scheithauer (2002), Borndörfer et al. (2003), Gamache et al. (1999), Gunluk et al. (2006), Kiwiel (2005), Perrot (2005), and Vanderbeck (2000)). It is also important to restrict the oracle to generating "proper" columns, i.e. columns whose coefficients do not exceed the right-hand-side of the constraint of the residual master problem (see Vanderbeck and Savelsbergh (2006)). A variant can be to round variables of the original formulation instead of that of the Dantzig-Wolfe reformulation (see Degraeve and Jans (2007) and Gunluk et al. (2006)). A local search (LS) can explore the neighborhood of the current master solution by deleting some of the "worst" columns of the incumbent solution and rebuilding a solution with one of the above constructive methods. Meta-heuristics can then be implemented from this LS paradigm (see Cimelière (2004) and Archetti et al. (2008)).

What is quite specific to our application is the fact that the primal heuristic aims at building an integer solution to the discrete master formulation, while any LP information (primal or dual) required by the heuristics is obtained by solving the aggregate master program. During the process of GH, RH or LS, the partial solution is iteratively augmented by selecting some tasks, including its starting period. Once a new task is recorded in the partial solution, master problem and pricing problems are updated. In the discrete master problem, the right-hand-side of constraints (14) is set to zero for pairs (i, t) already collected in the partial solution and a fixed number of vehicles is recorded as already used in the vehicle upper bound constraints (15) for some periods. In the aggregate master problem, the fractions of demand remaining to be covered are adjusted in constraints (24), while the average vehicle use is updated in constraints (25). We pass on to the pricing problem restrictions that are specific to some starting dates: new tasks must not cover demands that are already covered, or cannot use a vehicle in periods where no more vehicles are available. Hence, even when dealing with the aggregate master problem, one must iterate on different pricing sub-problems for each starting date. The reverse of such modifications are performed when deleting a task from the partial solution in the LS heuristic.

Our experiments with the RM approach and the GH were not successful (see Michel (2006)). For instances with 100 customers, no integer solution to the RM integer program could be found within 2 hours of computing time (the solver used was Xpress (2006)). The GH is quite fast but the optimality gaps are significant (49.36% on average). Therefore, in the rest of this section, we focus on the rounding heuristic (RH) and local search (LS).

We compared several implementations of the RH, varying the criteria for selecting the column to be rounded up. We also tested different ways of fixing the starting date of the selected columns. We varied the effort spent in re-optimizing the residual aggregate master LP after rounding. We tested a diversification strategy consisting in exploring different selections of the first column to be rounded up (at the root node). Finally, we tried calling the RH before reaching master LP optimality. In the process of the RH, one might encounter an integer solution to the aggregate master problem (this often happens after fixing a large part of the variables). Then, it remains to see whether there exists an associated integer solution to the discrete master residual problem. To this end, we solve to IP optimality the later discrete master problem restricted to the current pool of columns using the discrete variables: $\lambda_{rs} = 1$ if the aggregate column r is taken with the starting date s. The fact that a partial solution has already been fixed not only reduces the size of the residual problem but also breaks the symmetry. Hence, these restricted discrete master problems can be solved quickly in practice (contrary to our experience with the RM heuristic approach).

In Table 4, we compare these variants of the RH on the test problems of Section 6. The variant named “Basic” consists in calling the RH on the optimal LP solution to the aggregate master problem at the root node. The number of iterations in re-optimizing the master LP is bounded by 300. We diversify the search by calling the RH 3 times, ensuring that the first column being rounded up is different on each 3 passes by means of a tabu list. The rounded column and its starting date are both chosen using a deterministic criterion: we select the column r and the starting date s with the smallest score based on the ratio of column cost per unit of constraint satisfaction. The column cost is equal to $\delta_{rs} + \alpha\frac{c^r}{p^r}$ where $\delta_{rs} = 1$ if one must use an additional vehicle after this fixation. As a measure of satisfaction of constraints (24) and (25), we use $\frac{\sum_{i,\ell} \ell x_{i\ell}}{p^r} \times p^r = \sum_{i,\ell} \ell x_{i\ell}$. The RH implementation named "Random" consists in selecting the rounded-up column, r, with a probability proportional to $min(\lambda_r, 1)$ and fixing its starting date randomly (using the uniform probability distribution). Variant named "100cg" is the basic variant where the aggregate master LP re-optimization is limited to 100 iterations of the column generation procedure. Variant named "every200" consists in applying the basic variant every 200 iterations of the column generation procedure (if there is no artificial columns in the LP solution). Variant named "cut" consists in applying the RH after the cutting plane procedure (then a solution typically has most of its variables that have a value smaller than 0.2).

Table 4 reports on the average optimality gap, the average number of vehicles used in the primal solution, denoted $V_{\max}$, and the total computing time in hours:minutes:seconds. The fastest variant is obviously "100cg". Note that the computing time for "cut" is large due to the difficulty in re-optimizing the master problem after adding cuts. With "Random", we have no solution for IND157 (a "_" indicates that no solution was found). The variant with the best optimality gap and a minimal number of vehicles is "Basic". In Table 5, we report even better results obtained by calling "Basic" at the root node and after branching on V_{aver}: the average gap is 11.66%. For all instances but one, we can show that the number of vehicles used in our primal solution is minimum because we have branched on V_{aver} and Node 1 has been proved infeasible. By imposing some restrictions on the solution space, such as further restricting the set P to $P = \{1, 2, 3\}$, we sometime get even smaller optimality gaps (we then obtain a gap of 9,67% on average).

Table 4: Comparing variants of the rounding heuristic.

Instance	RAND100 (10)			IND60 (5)			IND172			IND157			**av. over 17 inst**	
RH variant	gap	$V_{\max}$	Time	gap	V	Time	gap	V	Time	gap	V	Time	gap	V
Basic	14.12	4	10:05	18.39	3.2	3:33	6.89	6	37:56	10.86	7	42:03	**14.76**	**4.05**
Random	41.43	5.5	10:01	38.31	4	3:50	17.83	7	1:12:00	_		14:42		
100cg	15.04	4	4:41	27.2	3.4	1:47	8.99	6	20:58	14.77	7	9:23	21.18	4.11
every200	14.30	4.1	19:57	25.88	3.4	5:57	6.89	6	37:56	6.58	7	1:45:49	16.82	4.18
cut	32.55	4.9	46:23	19.86	3.2	14:46	7.04	6	1:54:32	25.85	8	2:25:23	26.92	4.65

In the LS heuristic, the neighborhood of the current IP solution is defined by removing a few columns and re-building a complete integer solution with the above basic rounding heuristic procedure. For the choice of columns to delete, we tested two variants: either a random selection of 6 ($= p_{\max}$) columns are deleted from the primal solution, or we made a deterministic selection as follows. We delete columns from the primal solution that seem to be "poor". We arbitrarily assess a column as poor if its load is less than $\frac{1}{3}W$. If none qualifies as poor, we pick a column at random. This first selection of columns is denoted by set Q^0. We delete them all. Then, we select as candidate for deletion along Q^0, other

Table 5: Basic Rounding Heuristic called at Nodes 0 and 2.

Instance	gap	$V_{\max}$	Total time	% of time spent in RH
RAND100 (10)	12.29	4	18:22	90
IND60 (5)	11.75	3	6:10	78
IND172 (1)	5.574	6	1:22:02	79
IND157 (1)	10.86	7	1:30:16	91
av. over 17 inst	**11.66**	**4**		

columns that either concern the same periods (a set denoted Q^1) or share the same vehicle (a set denoted Q^2). I.e., Q^1 is the set of columns from the primal solution that have the same periodicity and starting date than a column in Q^0; Q^2 is the set of columns from the primal solution that have the same periodicity than a column in Q^0 and result in a decrease in the number of vehicle used when deleted from the solution. Our purpose is to favor the exchanges of customers between tasks used in the same periods and to attempt to free a vehicle. $Q^1 \cup Q^2$ defines the set of extra candidate columns for deletion. From Q^1, we pick at most 1 column for each pair of periodicity and starting date, that with the minimum score $\frac{dist^r}{W-load^r} * (1 + \frac{nbDel^r}{2})$, where $dist^r$ represents the distance between r and the closest deleted columns from Q^0 (i.e., the sum of the travel times between customers in the two columns), $load^r$ is the load, and $nbDel^r$ is the number of times that column has been deleted in past trials. Similarly, we pick at most one column from Q^2 for each periodicity p and starting date s. Rebuilding a solution with the RH is computationally expensive. Hence, we restrict the number of trials in our exploration of the neighborhood: we stop after 10 exchange trials, if we do not obtain an improved incumbent solution. In rebuilding a solution with the RH, the first rounded-up column is not selected among columns that have just been deleted or that have already been selected as first column in previous trials.

In Table 6, we compare these two variants in trying to improve the first solution found by the RH. For this test, we stop the LS algorithm at the first improvement in the incumbent solution (with at most 10 trials in exploring the neighborhood). Table 6 reports: the solution improvement, "impr", (expressed as a percentage); the number of instances for which we succeed in finding an improving solution, "succ", expressed as a fraction of the total number of instances; the average number of trials, "trials", that is needed in our exploration of a neighborhood before finding an improving solution (if we do find one) – it is bounded by 10 –; and the total computing time. We always find some improving solution when using the deterministic variant. The improvements vary from 1% to 15%. In Table 7, we present the results using the deterministic LS on all solutions obtained during all the passes of the basic RH at the root node. The overall computational effort is however bounded by restricting to 500 the number of columns that can be generated during the primal heuristic. The percentage of time spent in LS includes the time to rebuild a primal solution using the RH approach. The comparative performance of RH and LS is summarized in Table 8. It shows that the optimality gap decreases from 14.76 to 11.22 thanks to LS. Moreover, we obtain the minimal number of vehicles with the last two methods.

Table 6: Comparison of two LS variants applied to the first primal solution generated with RH.

	Random selection of deleted columns				Deterministic selection of deleted columns			
Instance	% impr in primal bd	succ	trials	time	% impr in primal bd	succ	trials	time
RAND100 (10)	1.60	7/10	5.1	6:33	8.43	10/10	1.9	4:47
IND60 (5)	4.25	3/5	5	3:11	8.27	5/5	2.4	1:58
IND172 (1)	0.00	0/1	-	25:47	0.34	1/1	6	21:28
IND157 (1)	0.34	1/1	2	27:17	0.26	1/1	1	26:43
av. over 17 inst	2.11	11/17	4.4		**7.43**	**17/17**	**2.2**	

Table 7: Results with deterministic LS applied to each primal solution generated during the 3 passes of the basic RH.

Instance	gap	$V_{\max}$	Total time	% of time spent in RH	% of time spent in LS
RAND100 (10)	11.88	4	46:04	19	76
IND60 (5)	11.74	3	13:27	22	74
IND172 (1)	5.735	6	1:00:50	48	38
IND157 (1)	7.389	7	54:50	71	24
aver. over 17 inst	**11.22**	**4**			

Table 8: Comparison of average results with RH and LS.

Method	average gap	Total time
RH at the root node only	14.76	11:41
RH at B-a-P nodes 0 and 2	11.66	22:47
RH + LS at the root node only	11.22	37:52

8 Results a large industrial test case

Finally, we apply our approach to a large real-life industrial instance with 260 customers and an implicit horizon of 60 periods. This test problem represents the on-the-ground situation for a restricted geographical area during year 2005. At the planning level, a period corresponds to a week. More than 60% of the customers have a maximum time lag between visits that is smaller than 6 weeks. The others have a $t^{\max}$ ranging from 7 to 14 weeks. Applying partial branching and cutting planes to compute the best possible dual bound requires 8h34m of computing time on our PC bi-pro. Xeon 3GHz, 2Go. The "DB+br+cut" dual bound has value 838.17. The cutting plane procedure alone requires 5h21m.

In a different run, skipping the time consuming cutting plane procedure, we apply our primal heuristics. Applying the basic RH combined with partial branching gives a primal solution with a gap of approximately 9% from the above dual bound in 4h29m (this run does not include the exploration of Node 1 that has been proved infeasible in our dual bound computation). The number of vehicles is equal to 9 (it is optimal as proved by the fact that the branch with $V_{\text{aver}} \leq 8$ is infeasible). In another trial, we restricted ourself to the root node and apply the basic RH followed by LS. We again obtain a solution with a gap of approximately 9% from the above dual bound and a number of vehicles equal to 9 in 3h40m.

We obtain even better results by restricting the search space to solution using periodicities in the set $P = \{1, 2, 3\}$. Applying the RH plus partial branching or RH and LS at the root node followed by a post-optimization procedure, we obtain a primal solution with a gap of 6% (comparing our primal bound to the dual bound obtained for $P = \{1, 2, 3, 4, 5, 6\}$) in 1h49m and 2h53m respectively. The post-optimization procedure consists in transforming one task of periodicity 3 into two tasks of periodicity 6 while this improves the solution (it can allow to visit a site every 6 periods instead of 3). The feasible splitting scenarios are built into an integer program that provides a solution minimizing cluster costs. In our solution we observe that the customers are well gathered around the seed, except for some customers that are on the path from the depot or to the dumping site. The results are summarized in Table 9.

In Table 10, we compare this solution to the one used by our industrial partner on an average week of year 2005. From what we know, this industrial solution was built on a day-to-day basis looking at the customers requiring the most urgent service and trying to achieve some regional clustering with optimization tools. We received the total number of vehicles used in each period, the number of customer visits and the total travel distance of the operational solution used by the industrial partner in 2005. We estimate the average behavior on a week by dividing these numbers by 52. For comparison, we compute

Table 9: Comparison of average results with RH and LS approaches on the industrial test case (with 260 customers).

Algorithm	gap (in %)	$V_{\max}$	total time
RH + partial branching with $P = \{1,2,3,4,5,6\}$	9.25	9	4h29m
RH + LS at the root node with $P = \{1,2,3,4,5,6\}$	8.92	9	3h40m
RH + partial branching with $P = \{1,2,3\}$ + post-optim	6.67	9	1h49m
RH + LS at the root node with $P = \{1,2,3\}$ + post-optim	6.23	9	2h53m

Table 10: Comparison between the operational solution derived from our best tactical solution and the industrial operational solution implemented in 2005 – (*) In the industrial solution, a visit is not counted if, once on site, the driver decides not to collect the stock because he finds it too low.

	av # of customer visits per week	$V_{\max}$	av travel distance per week
sol of the tactical planning model	98.5	9	711 km
industrial solution	59 (*)	10	782 km

an operational solution from our tactical solution by heuristically solving a travelling salesman problem for each cluster. We observe that our solution requires 9 vehicles instead of 10 in the averaged-out industrial solution. The travel distance of our solution is 10 percent smaller than the industrial distance. We planned more customer visits (5910 visits) than in the industrial solution that involves 3540 customer visits. Thus, we tend to collect a site before it reaches its maximum filling capacity while having a shorter total travel distance. Indeed, visiting a site that is close to a vehicle route while it is not yet full takes far less travelling than having to do a detour when it requires service. Moreover, visiting customers on a more regular basis can be an edge against uncertainty and helps to avoid stock-out situation.

Conclusion

We have solved a large scale industrial planning problem untreated in the inventory routing literature, optimizing simultaneously the design of vehicle tasks and the management of stocks at customer sites. We address the problem by generating periodic tasks for individual vehicles and combining them into a global schedule in a master program. We show that a mathematical programming based approach (more precisely a truncated Branch-and-Price algorithm combined with primal heuristics) is a viable option to obtain solutions for large scale problems. Contrary to the heuristic approaches traditionally used in the literature on inventory routing, we bound the deviation between our primal solution and the optimal solution to our mathematical model. Moreover, our preliminary comparison with an industrial solution shows that our primal solution may have an important impact on industrial practice.

An important feature of our approach is a formulation modeling an average behavior to avoid symmetry. Moreover, our tactical planning formulation models an objective of "regional clustering" while leaving specific routing issues for the operational level. This original model is motivated by industrial practice and goals as described by our partner. Our mathematical model can potentially be used in sensitivity analysis to see whether the given fleet size V can be reduced. Alternatively one can iterate on different values of V and generate the Pareto optimality curve by solving our model (with a large α) for each feasible V. In an effort to build robust solutions, one might modify our model to minimize the maximum vehicle capacity that is used. Balancing the slack capacity in each vehicle task is another way to hedge against uncertainty. Further research would be (i) to perform a direct comparison with other constructive or meta heuristics especially developed for our model; (ii) to develop an operational solver using our tactical solution as a target.

Acknowledgments

We are very grateful to our industrial partner, F. Rodes, for sharing with us problem description and real-life data. We acknowledge his financial support that helped in conducting this study. We thank A. Miller and P. Pesneau, who reviewed drafts of this paper. We owe special thanks to the anonymous referees who provided in depth feedback and made constructive remarks that were particularly helpful in improving the paper.

References

Archetti, C., L. Bertazzi, G. Laporte, M. G. Speranza. 2007. A branch-and-cut algorithm for a vendor managed inventory routing problem. *Transportation Science* **41** 382–391.

Archetti, C., M. W. P. Savelsbergh, M. G. Speranza. 2008. An optimization-based heuristic for the split delivery routing problem. *Transportation Science* **42** 22–31.

Bell, W., L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, P. Prutzman. 1983. Improving the distribution of industrial gases with on-line computerized routing and scheduling optimizer. *Interfaces* **13** 4–23.

Belov, G., G. Scheithauer. 2002. A cutting plane algorithm for the one dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research* **141** 274–294.

Bertazzi, L., M.Savelsbergh, M. G. Speranza. 2008. *The Vehicle Routing Problem: Latest Advances and New Challenges*, vol. 43, chap. Inventory Routing. Operations research/computer science interfaces series ed. Springer, 49–72.

Bertazzi, L., G. Paletta, M. G. Speranza. 2002. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science* **36** 119–132.

Borndörfer, R., M. Grötschel, A. Löbel. 2003. *Mathematics - Key Technology for the Future*, chap. Duty scheduling in public transit. Springer, 653–674.

Bramel, J., D. Simchi-Levi. 1995. A location based heuristic for general routing problems. *Operations Research* **43** 649–660.

Briant, O., C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck. 2008. Comparison of bundle and classical column generation. *Mathematical Programming* **113**(2) 299–344.

Campbell, A. M., L. W Clarke, A. Kleywegt, M. W. P. Savelsbergh. 1998. *Fleet Management and Logistics*, chap. The Inventory Routing Problem. Kluwer, 95–113.

Campbell, A. M., L. W Clarke, M. W. P. Savelsbergh. 2002. *The Vehicle Routing Problem*, chap. Inventory routing in practice. SIAM Monographs on Discrete Mathematics and Applications, 309–330.

Campbell, A. M., M. W. P. Savelsbergh. 2004. A decomposition approach for the inventory routing problem. *Transportation Science* **38** 488–502.

Ceselli, A., G. Righini, M. Salani. 2007. A column generation algorithm for a vehicle routing problem with economies of scale and additional constraints. *Sixth Triennial Symposium on Transportation Analysis*. Phuket, Thailand.

Chabrier, A. 2003. Heuristic branch-and-price-and-cut to solve a network design problem. *Proceedings CPAIOR*. Montréal, Canada.

Chabrier, A., E. Danna, C. Le Pape. 2002. Coopération entre génération de colonnes et recherche locale appliquées au problème de routage de véhicules. *Huitièmes Journées Nationales sur la résolution de Problèmes NP-Complets (JNPC)*. Nice, France, 83–97.

Chien, T. W., A. Balakrishnan, R. T. Wong. 1989. An integrated inventory allocation and vehicle routing problem. *Transportation Science* **23** 67–76.

Christiansen, M. 1999. Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science* **33** 3–13.

Christiansen, M., B. Nygreen. 1998. A method for solving ship routing problems with inventory constraints. *Annals of Operations Research* **81** 357–378.

Cimelière, F-L. 2004. Optimisation du traitement de l'ordre de fabrication dans l'industrie textile. Ph.D. thesis, Université Bordeaux 1, France.

Cordeau, J-F., G. Laporte, M. W. P. Savelsbergh, D. Vigo. 2007. *Transportation*, vol. 14, chap. Vehicle Routing. Handbooks in OR and Management Science, Elsevier, 367–428.

Custódio, A-L., R-C. Oliveira. 2001. A system of logistic management integrating inventory management and routing. *Optimization 2001*. Aveiro, Portugal.

Degraeve, Z., R. Jans. 2007. A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot sizing problem with set-up times. *Operations Research* **55** 909–920.

Dror, M., M. Ball. 1987. Inventory/routing: reduction from an annual to a short-term problem. *Naval Research Logistics* **34** 891–905.

Federgruen, A., D. Simchi-Levi. 1995. *Network Routing*, *Handbooks in OR and Management Science*, vol. 8, chap. Analysis of Vehicle Routing and Inventory Management Problems. Elsevier ed. Amsterdam, 297–371.

Gamache, M., F. Soumis, G. Marquis, J. Desrosiers. 1999. A column generation approach for large scale aircrew rostering problem. *Operations Research* **47** 247–263.

Gaur, V., M. L. Fisher. 2004. A periodic inventory routing problem at a supermarket chain. *Operations Research* **52** 813–822.

Golden, B., A. Assad, R. Dahl. 1984. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale System 7* 181–190.

Gunluk, O., T. Kimbrel, L. Ladanyi, B. Schieber, G. Sorkin. 2006. Vehicle routing and staffing for sedan service. *Transportation Science* **40** 313–326.

Kiwiel, K. 2005. An inexact bundle approach to cutting stock problems. Tech. rep., Systems Research Institute, Warsaw.

Michel, S. 2006. Optimisation des tournées de véhicules combinées à la gestion de stock. Ph.D. thesis, Université Bordeaux 1.

Nemhauser, G. L., L. A. Wolsey. 1988. *Integer and Combinatorial Optimization*, chap. II.1.4 : The theory of valid inequalities: superadditive functions and valid inequalities. John Wiley & Sons, 229–237.

Perrot, N. 2005. Integer programming column generation strategies for the cutting stock problem and its variants. Ph.D. thesis, Université Bordeaux 1, France.

Pisinger, D. 1995. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research* **83** 394–410.

Schmid, V., K. F. Doerner, R. F. Hartl, M. W. P. Savelsbergh, W. Stoecher. 2007. An effective heuristic for ready mixed concrete delivery. *Sixth Triennial Symposium on Transportation Analysis*. Phuket, Thailand.

Taillard, E. 1999. A heuristic column generation method for the heterogeneous vrp. *RAIRO Operations Research* **33** 1–14.

Vanderbeck, F. 2000. Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Operations Research* **48** 915–926.

Vanderbeck, F. 2005. *Column Generation*, chap. Implementing Mixed Integer Column Generation. Kluwer's series in Operations Research, Kluwer, 331–358.

Vanderbeck, F., M. P. W. Savelsbergh. 2006. A generic view of dantzig-wolfe decomposition in mixed integer programming. *Operations Research Letters* **34** 296–306.

Webb, I., R. Larson. 1995. Period and phase of customer replenishment: A new approach to the strategic inventory/routing problem. *European Journal of Operational Research* **85** 132–148.

Witucki, M., P. Dejax, M. Haouari. 1997. Un modèle et un algorithme de résolution exacte pour le problème de tournées de véhicules multipériodique: une application à la distribution des gaz industriels. *Deuxième congrès international franco-québecois de génie industriel*. Albi, France.

Xpress. 2006. *User guide and Reference Manual, Dash Optimization*. http://www.dashoptimization.com.