Column generation based heuristic for tactical planning in multi-period vehicle routing

M. Mourgaya and F. Vanderbeck *

Applied Mathematics, University Bordeaux 1

January 20, 2006

Abstract

The Periodic Vehicle Routing Problem (PVRP) consists in establishing a planning of visits to clients over a given time horizon so as to satisfy some service level while optimizing the routes used in each time period. The tactical planning model considered here restricts its attention to scheduling visits and assigning them to vehicles while leaving sequencing decisions for an underlying operational model. The objective is twofold: to optimize regional compactness of the routes in a desire to specialize routes to restricted geographical area and to balance the workload evenly between vehicles. Approximate solutions are constructed using a truncated column generation procedure followed by a rounding heuristic. This mathematical programming based procedure can deal with problems with 50 to 80 customers over 5 working days which is the range of size of most PVRP instances treated in the literature with meta-heuristics. The paper highlights the importance of alternative optimization criteria not accounted for in standard operational models and provides insights on the implementation of a column generation based rounding heuristic.

Key words: Branch and bound, Combinatorial optimization, Heuristics, Integer programming, OR in strategic planning, Routing.

---

*Corresponding author: fv@math.u-bordeaux1.fr, IMB, Université Bordeaux 1, 345 cours de la libération, F- 33405 Talence Cedex, France. Tel. 33 (0)5 40 00 21 22, Fax 33(0)5 40 00 21 23.

The Periodic Vehicle Routing Problem (PVRP) combines the problematic of the standard vehicle routing problem with that of planning customer visits over a given time horizon: (1) one must decide on the dates for customer visits so as to satisfy some service level requirements (one tries to avoid stock-out situation or surplus at customer sites); then, (2) one must assign customer to vehicles and, finally, (3) one must optimize the route of each vehicle for each day of the time horizon. Vehicle loads that are defined as the sum of the customer demands included in the route are bounded. The PVRP admits several variants in terms of the objectives, the specific constraints and the type of solutions that are sought (a classification is presented in Mourgaya and Vanderbeck, 2005).

In this study, a hierarchical optimization approach is adopted: decisions (1) and (2) are first stage decisions while (3) shall be optimized in a second stage where decisions (2) can also be reconsidered (decisions (2) and (3) define a Vehicle Routing Problem – VRP). The first stage problem, called *tactical planning*, is the focus of this paper. Our objective is to minimize the Euclidean distance between customers visited by the same vehicle (a surrogate measure for distance traveled that favors routes associated with compact cluster of customers) while keeping a balanced workload between vehicles. A column generation formulation is used and heuristic primal solutions are derived by iteratively rounding the LP solution. In a previous publication (Mourgaya and Vanderbeck, 2005), we developed a constructive heuristic for a real-life money transport application (it is much faster than the present approach but gives rise to lower quality solutions).

Our hierarchical approach differs from previous studies of the PVRP where decisions (1), (2), and (3) are taken simultaneously to optimize a standard operational objective such as minimizing travel costs (Christofides and Beasley, 1984; Tan and Beasley, 1984; Ball, 1988; Russel and Gribbin, 1991; Gaudioso and Paletta, 1992; Chao et al., 1995; Cordeau et al., 1997; Vianna et al., 1998). In these references, there is either a single objective criteria or a hierarchical optimization of two criteria (with workload balancing used as a second criteria). The solution approaches used are typically a constructive heuristic followed by a local search procedure with the emphasis put on developing the improvement procedure. Tabu Search and Genetic Algorithms are proposed respectively in Cordeau et al. (1997), Angelelli and Speranza (2002) and Vianna et al. (1998). The comparative results of Cordeau et al. (1997) show that their Tabu Search method gives the best results for the aggregate operational model.

The literature also provides indications on the size of problems and subproblems that can be handled using the tools of exact optimization, constructive heuristics or local search heuristics. For the VRP, Toth and Vigo (2002) say that "the largest VRP instances that can be consistently solved by the most effective exact algorithms proposed so far contain about 50 customers". Recently, Fukasawa et al. (2004) solved instances involving up to 200 customers and 17 vehicles by branch-and-price-and-cut (combining column and cut generation) within small deviation from optimality. Note that for a given number of customers, the problem gets easier as the number of vehicle increases as it implies smaller clusters. The variant with time window constraints are solved in Chabrier et al. (2002), Feillet et al. (2003), and Irnich and Villeneuve (2003). When time windows are very tight, the problem gets easier and exact column generation algorithms are able to handle problems with around 100 customers. Problems with looser time windows are harder. They have been solved for size ranging up to 50 customers.

For a given number of customers, PVRP instances are typically harder to solve than VRP: not only one is confronted with the extra complexity layer of having to schedule the visits, but the number of vertices to include in routes is the number of customers multiplied by the number of visits to those customers. Hence, the above cited PVRP literature proposes only heuristics that have been tested on instances whose size ranges between 20 and 163 customers (with one exception: 417 customers) and 1 to 9 vehicles. Here again, for a fixed number of customers, the problem gets easier as the number of vehicles increases. By contrast, some of the real-life problem instances, like the one treated in Mourgaya and Vanderbeck (2005), can involve several thousand of customers each with their own visit requirements which sum up to more than 10000 visits to plan over say a 20 period time horizon with around 20 vehicles (yielding an average of around 40 customer visits per vehicle).

The contribution of this paper is twofold. On the modeling side, it emphasizes the criteria that are important in the eyes of the practitioners and were ignored in the operational models treated in the literature (the paper offers some numerical evaluation of the trade-off between these conflicting objectives). On the algorithmic side, it illustrates how rounding heuristic can be applied in a column generation context. Few applications of this approach can be found in the literature. Rounding can be either apply directly to the variables of the column generation reformulation as in Vanderbeck (1999) or to the variables of the underlying original formulation as in Elhedhli and Goffin (2005). The implementation described herein is more involved due to the complexity of the problem treated. Faced with a strongly NP-hard pricing problem, a quickly growing number of columns and some symmetry issues, we had to develop practical strategies for the initialization, the column generation procedure, the rounding procedure, and heuristic restrictions of the solution space. These strategies are evaluated by way of numerical comparisons. We hope that our study can provide insights for the application of a column generation based rounding heuristic to other applications.

The paper is organized as follows. Section 1 presents the tactical planning model in details and discuss the practical relevance of the optimization criteria used in this study. Sections 2 to 5 describe the ingredients of our column generation based heuristic: column generation formulation, subproblem solver, initialization and rounding procedure. Sections 6 and 7 present computational comparison of various implementation strategies and our attempts to assess solution quality by comparison to dual bounds and to the PVRP solutions of the literature. We then conclude and discuss possible extensions of such approach to larger instances.

1 The tactical planning model

The motivation for using a hierarchical approach to the PVRP are (*i*) to account for other optimization criteria that arise in tactical planning (such as minimizing the number of vehicle used, optimizing workload balancing, and regional specialization) and (*ii*) to deal with larger problems (leaving specific operational constraints such as time windows for visits, delivery time, visit sequence restrictions and the like for the operational model are essential simplifications for the tractability of the tactical planning model).

We consider that service level requirements take the form of alternative feasible visit schedules for each customer that we call *scenarios*. They account for (*i*) prescribed frequency of visits, (*i*) minimum and maximum time lag between successive visits, (*iii*) specificity of access to the location (such as avoiding market day), and (*iv*) a desired periodicity across time horizons by controlling initial and final time lag. In assigning customers to vehicles, our objective is twofold: we want to balance the workload between vehicles and we want to concentrate routes in limited geographical area, an objective that we call *regionalization.* We can also see if the vehicle fleet can be reduced in size and what are the consequences on vehicle workloads and regionalization.

The workload balancing between vehicles is of crucial importance in practice: not only does it ensure a form of fairness between the different drivers that shall have similar work days as a result; but it also yields "*robust solutions*" (unpredicted extra demands can easily be incorporated in a balanced solution while, in an unbalanced solution, routes that are filled-up cannot accommodate extra demands). Moreover, the workload can be a surrogate measure of some operational constraints (such as bounded travel time in instances where workload is correlated with cardinality of the cluster and travel time). Hence, balancing the workload can help, in some cases, preparing the operational solutions. To enforce it, we bound the load of a route: a route is defined by the associated *cluster* of customers, that we note C, and its workload is

$$\text{workload}(C) = \sum_{i:i \in C} d_i \leq W \ ,$$

where d_i is the demand of customer i that is assumed to be constant on each visit. (Customer demands could depend on the time since the last visit, something that we ignore in our aggregate planning model.) By lowering the workload bound W (lower than vehicle capacity), we can achieve an even repartition of the workload between vehicles.

The objective of *regionalization* reflects practical considerations, such as driver specialization. Indeed, it is common practice to let the driver select his own route, having prescribed the sequence of customer visits. Hence, it is important to assign drivers to routes concentrated in an area they are familiar with. Practitioners tend to dislike routes that have been optimized for length and spread over quite different areas while crossing one another. On the other hand, organizing routes by area code is something they would appreciate. Moreover, grouping visit points into well regionalized area in the tactical model will hopefully lead to routes of short length in an underlying operational model.

Cluster regionalization is achieved by minimizing the sum of the Euclidean distances between customer locations, i, that are assigned to the same cluster, C. In other words, the cluster costs are

$$\text{cost}(C) = \sum_{ij:i\in C, j\in C, i<j} c_{ij}$$

where c_{ij} is the distance between customer locations i and j (we assume a symetric distance matrix at the tactical planning level). Then, minimizing the total cost of the clusters allows to achieve both the *separation* of the clusters and *homogeneity* within the clusters. Indeed, the sum of all distances is a constant. Hence, minimizing the total distance between points belonging to the same cluster maximizes the sum of the distance between points belonging to distinct clusters: $\underbrace{\sum_{i<j} c_{ij}}_{constant} = \underbrace{\sum_{C} \sum_{ij:i\in C, j\in C, i<j} c_{ij}}_{\min} + \underbrace{\sum_{C} \sum_{ij:i\in C, j\notin C, i<j} c_{ij}}_{\max}$.

The *second stage problem* in the proposed hierarchical approach is an operational model optimizing sequencing decisions (3). It accounts for all operational constraints that depend on the sequence. Observe that solving a Traveling Salesman Problem (TSP) for each cluster, is not satisfactory. Indeed, for the cluster formed in the first stage there might be no feasible sequencing that allow to meet operational constraints such as bounded travel times or time windows. An alternative approach consists in reconsidering vehicle assignment decisions (2) in the operational model. However, solving a VRP for each day of the planning horizon is not satisfactory neither since ignoring the clustering decisions taken in the first stage while solving the second leads to loosing the good regionalization and workload balance of the solution constructed at the first stage. Hence, what is needed is an approach that optimize decisions (2) and (3) while trying to minimize deviation from the clustering decisions that were taken at the first stage. Such ad-hoc VRP solver is beyond the scope of this paper. It is left for further research.

2 An Approach based on a column generation reformulation

The initial step of our procedure is a constructive heuristic that aims at building Pareto optimal solutions with regards to the two optimization criteria: workload balancing and regionalization. Then, we set the maximum workload bound to its value in this initial solution and optimize the regionalization objective using a LP based rounding heuristic. The constructive heuristic used at the outset is relatively efficient at balancing the workload but is too myopic to produce good quality regionalization. Thus, the rounding heuristic can be seen as an improvement heuristic applied after fixing the workload balancing criteria. As it is based on solving a mathematical programming formulation of the problem, this heuristic has global optimization properties that lack to standard local search improvement procedures and was also lacking in the greedy constructive procedure used at initialization.

The input to our model summarizes as follows: There are n customers indexed by $i = 1, \dots, n$; T periods indexed by $t = 1, \dots, T$; and V identical vehicles indexed by $k = 1, \dots, V$. For each customer i, we are given a set of scenarios S_i. A scenario $s \in S_i$ is an indicator vector whose component $s_t = 1$ if scenario s involves a visit in period t. Let d_i denote the quantity demanded in any visit to customer i. Let c_{ij} denote some measure of distance between customers i and j. To restrict the search space, the adjacency list of each customer only includes its close neighbors: we limit the neighborhood to those customer for which $c_{ij} \leq \frac{\max_{ij} c_{ij}}{2}$. Finally, let W denote the maximum workload of a vehicle.

The tactical planning model can be formulated in terms of variables $x_{itk} = 1$ if customer i is visited in period t by vehicle k, $y_{ijt} = 1$ if customers i and j are in the same cluster in period t, and $z_{is} = 1$ if visit scenario s is chosen for customer i. It takes the form

$$
\begin{array}{llrcll}
 & & \min \sum_{i,j,t} c_{ij}\, y_{ijt} & & & (1) \\
\text{[P]} & \text{s.t.} & \sum_{s \in S_i} z_{is} & \geq & 1 \quad \forall i & (2) \\
 & & \sum_{k=1}^{V} x_{itk} - \sum_{s \in S_i} s_t\, z_{is} & \geq & 0 \quad \forall i, t & (3) \\
 & & x_{itk} + x_{jtk} - y_{ijt} & \leq & 1 \quad \forall i, j, k, t & (4) \\
 & & \sum_{i} d_i\, x_{itk} & \leq & W \quad \forall k, t & (5) \\
 & & x_{itk}, y_{ijt}, z_{is} & \in & \{0, 1\} \quad \forall i, j, s, t, k\ . & (6)
\end{array}
$$

Constraints (2) force the selection of a scenario for each customer; (3) ensure that a vehicle visits the customer on each day required by the selected scenario; (4) define edges that are within clusters; (5) limit the workload assigned to a vehicle.

This natural formulation suffers from two drawback however: a weak Linear Programming (LP) relaxation and a symmetry inherent to the introduction of index k that artificially differentiate what are identical vehicles. A Dantzig-Wolfe reformulation (Vanderbeck, 2000) allows to get stronger dual bounds by LP relaxation and to avoid symmetry. Problem [P] decomposes when one dualizes constraints (2-3). Constraints (4-5) define a subproblem for each vehicle on which a column generation reformulation can be based.

Let Q be the set of feasible cluster that can be assigned to a vehicle. Let them be defined by an indicator vector x^q: $x_i^q = 1$ if customer i is in cluster q. Thus,

$$Q = \{x \in \{0,1\}^n : \sum_i d_i x_i \leq W\} .$$

Let c_q be the cost of cluster q:

$$c_q = \sum_{ij} c_{ij} y_{ij}(x^q)$$

where $y_{ij}(x^q)$ is a edge selection vector satisfying constraints (4) for $x_{.tk} = x^q$. Then, [P] can be reformulated as

$$
[\mathrm{M}] \qquad
\begin{array}{lrcll}
\min \sum_t \sum_{q \in Q} c_q \lambda_q^t & & & & (7) \\
\text{s.t.} & \sum_{s \in S_i} z_{is} & \geq & 1 & \forall i \quad (8) \\
& \sum_{q \in Q} x_i^q \lambda_q^t - \sum_{s \in S_i} s_t \, z_{is} & \geq & 0 & \forall i, t \quad (9) \\
& \sum_{q \in Q} \lambda_q^t & \leq & V & \forall t \quad (10) \\
& \lambda_q^t & \in & \{0,1\} & \forall t, q \in Q \quad (11) \\
& z_{is} & \in & \{0,1\} & \forall i, s \quad (12)
\end{array}
$$

where $\lambda_q^t = 1$ if cluster q is selected for a vehicle in period t.

Observe that there remains some symmetry in formulation [M]: given a planning, one can apply a time shift to it and get an alternative solution. We refer to it as the *time symmetry*. In particular, any cluster $q \in Q$ (and associated column x^q) is valid for any period t. Hence, variables λ_q^t are drawn from the same set Q for each t. In an attempt to break this symmetry, we tried fixing a priori the scenario selection of a chosen customer to initialize the planning. This customer, called the *client_of_reference*, is selected as that requiring a number of visits closest to $\frac{T}{2}$ (ties are broken by considering the largest demand). The idea being to select the customer that might have the largest impact on further planning decisions. It is to be noted that there is no guarantee that an optimal solution exists with this fixed scenario for the *client_of_reference* (a counter exemple arises in our numerical test).

In view of the exponential size of set Q, the LP relaxation of [M] must be solved by dynamic column generation. The so-called master program, [M], is initialized with a subset of columns and associated variables (possibly artificial columns). At each iteration of the column generation procedure, the master LP restricted to the current set of columns is solved to optimality. Its dual solution is then used to price out columns that have not yet been included in the formulation: this is done by solving an optimization problem over (4-5) in search for the most negative reduced cost column. If this pricing subproblem identifies a negative reduced cost column, the latter is added to the master program and the procedure reiterates. Otherwise, we hold a proof of optimality of the current master LP solution. (See Desaulniers et al., 2005, for details of implementation.)

Observe that the intermediate LP solution to the restricted master does not define valid dual bounds for [P]. Only the optimal LP value to the unrestricted master does. However, valid dual bounds for [P] can standardly be obtained by Lagrangian relaxation: the intermediate Lagrangian dual bounds take the form

$$LB = Z_{LP}^{RM} + V \sum_t (\zeta^t(\pi) + \sigma_t) \tag{13}$$

where Z_{LP}^{RM} is the LP solution value of the current restricted master and $\zeta^t(\pi)$ denotes the minimum reduced cost of λ_q^t variables, i.e. the solution of the pricing-subproblem associated with period t, and (π, σ) is the dual solution of the restricted master LP associated to constraints (9) and (10) respectively.

3 The pricing problem

The pricing problem decomposes into T column generation subproblems, one for each period t, that take the form:

$$\zeta^t(\pi) \quad = \min\{ \sum_{ij} c_{ij} y_{ij} - \sum_i \pi_{it} x_i : x_i + x_j - y_{ij} \leq 1 \; \forall ij, \tag{14}$$

$$\sum_i d_i\, x_i \leq W,\ x_i \in \{0,1\}\ \forall i,\ y_{ij} \in \{0,1\}\ \forall ij\}$$

where π_{it} are dual values associated with constraint (9) (edges (i,j) are only defined for $i < j$). Problem (14) is the linearised form of a quadratic knapsack problem: the objective can be witten has $\sum_{ij} p_{ij} x_i\, x_j$ with $p_{ii} = \pi_{it}\ \forall i$ and $p_{ij} = p_{ji} = -\frac{c_{ij}}{2}\ \forall i < j$. This cost structure with negative coefficients makes the problem strongly NP-hard (Kellerer et al., 2004). Note that to iterate during the column generation procedure, one does not necessarily need to solve (14) to optimality. Any negative reduced cost column allows to go on. Hence, we developed a primal heuristic for (14). However, to obtain dual bounds (13), we need the exact solution to (14) or at least good dual bounds, $\underline{\zeta}^t(\pi)$, on the pricing subproblems (replacing $\zeta^t(\pi)$ by $\underline{\zeta}^t(\pi)$ in (13) defines a valid dual bound).

To construct primal solution to (14), we use a greedy insertion heuristic. We select a seed point around which we build a cluster with workload at least equal to $0.8 * W$ (the 80% filling factor is a parameter). In fact, we shall try several seed points selected for having largest ratio of collected price per unit of demand, $\frac{\pi_{it}}{d_i}$. Once the seed point is selected, we iteratively insert customers in the cluster C selecting that with maximum ratio $\frac{\pi_{it}-e_i}{d_i}$ where e_i is the edge cost resulting from inserting i: $e_i = \sum_{j\in C} c_{ij}$. We only examine candidates that are in the adjacency list of customer currently in the cluster (remember that we restricted adjecency lists to "close-by neighboors" as explained in Section 2). The best candidate for insertion is added to the cluster either if it has negative reduced cost ($e_i - \pi_{it} < 0$) or if the workload lower bound is not reached yet. Enforcing a workload lower bound is not required but it helps stabilizing the column generation procedure.

We attempted to compute dual bounds on subproblem (14) using Lagrangian relaxation. Once constraints $x_i + x_j - y_{ij} \leq 1$ are dualized (with associated dual variables $\mu_{ij} \geq 0$), the problem decomposes into

$$\begin{aligned}
\underline{\zeta}^t(\mu) &= \min\{\sum_{ij}(c_{ij} - \mu_{ij})y_{ij} : y_{ij} \in \{0,1\}\ \forall ij\} && (15)\\
&- \max\{\sum_i(\pi_{it} - \sum_{j\in\delta(i)}(\mu_{ij} + \mu_{ji}))\, x_i : \sum_i d_i\, x_i \leq W,\ x_i \in \{0,1\}\forall i\} && (16)\\
&- \sum_{ij} \mu_{ij} && (17)
\end{aligned}$$

The former subproblem (15) is trivial to solve (its LP relaxation has an integer solution). We further relax the second (16) into a continuous knapsack problem, yielding dual bound

$\zeta^t_{LP}(\mu)$. The best dual bound $\zeta^t_{LP}(\mu)$ is achieved by setting μ to the solution of the associated Lagrangian dual problem. The latter is an LP since $\zeta^t_{LP}(\mu)$ is defined based on the LP relaxation of both (15) and (16). It includes constraints $0 \leq \mu_{ij} \leq c_{ij}$ (see Mourgaya, 2004).

Instead of obtaining the optimal μ through solving the Lagrangian dual as an LP, we use a hybrid sub-gradient procedure that maintains $\mu_{ij} \in [0, c_{ij}]$ (so as to satisfy Lagrangian dual constraints) and hence the complementary primal solution is $y = 0$. We start with $\mu_{ij} = 0$ and iteratively adjusts μ_{ij}'s using the recursive rule:

$$\mu^{l+1}_{ij} = (1 - \alpha_l)\, \mu^l_{ij} + \alpha_l\, c_{ij}(x^l_i + x^l_j - 1)^+ \ ,$$

where x^l_i is the continuous knapsack subproblem solution at iteration l, $(*)^+ = max\{0, *\}$, and $\alpha_l \in (0, 1)$ is a geometrically decreasing factor. This updating rule ensures $\mu_{ij} \in [0, c_{ij}]$, with μ_{ij} increasing when $x_i + x_j > 1$ and decreasing otherwise. Moreover, one does not need to explicitly compute a matrix μ_{ij} (which can be cumbersome in case the instance is large), but one can maintain only the aggregate values defined in (16) and (17): the dual cost associated with nodes, $g_i = \sum_{j\in\delta(i)}(\mu_{ij} + \mu_{ji})$ for all i, and $s = \sum_{ij} \mu_{ij} = \frac{1}{2}\sum_i g_i$. The above updating rule can be directly implemented on aggregate values g_i: initially set $g^{l+1}_i := (1-\alpha_l)\, g^l_i$; then, for all ij such that $(x_i + x_j) > 1$, do $g^{l+1}_i := g^{l+1}_i + \alpha_l c_{ij}(x_i + x_j - 1)$ and $g^{l+1}_j := g^{l+1}_j + \alpha_l c_{ij}(x_i + x_j - 1)$. Finally, $s^{l+1} = \frac{1}{2}\sum_i g^{l+1}_i$. We have implemented this dual bounding procedure. However, in practice, the resulting Lagrangian bounds (13) remain too weak to be useful.

4 Initialization and column generation strategies

The master program is initialized with a primal solution obtained with a constructive heuristic. We also consider including columns from other primal solutions computed in the course of this heuristic. In addition, we include artificial columns that are useful to guarantee the existence of a feasible solution during the course of the rounding heuristic and may help to stabilize the column generation procedure.

The constructive heuristic is that of Mourgaya and Vanderbeck (2005) for the bi-criteria tactical planning problem (first phase in a hierarchical approach to the PVRP). It optimizes both workload balancing and regionalization (although a parameter, called “tendency”, allows to favor one or the other). It is a hybrid method using priority list, greedy insertion and Pareto optimality tools. It maintains a population of solutions that is a subset of Pareto optimal solutions selected for their diversity (avoiding in particular symmetric solutions). Customers are considered for insertion in priority order, treating first large demand or distant customers with little flexibility in their selection of scenarios. Insertions are done following a cheapest insertion greedy rule. The population size is kept reasonably small by cleaning up the set of partial solutions: To limit the number of solutions generated at each iteration, one keeps only Pareto optimal solutions, selects only a part of the Pareto optimal curve according to the “tendency”, and eliminate symmetric solutions.

We tried two initial solutions: either we set the “tendency” to obtain a solution that favors regionalization or we set it to emphasize workload balancing (given that regionalization shall be optimized through the rounding heuristic). In any case the maximum workload W is set as that of the solution of the constructive heuristic. Another variation is whether or not we record all primal solutions provided by the population of solutions of the constructive heuristic, or only the best one.

We define so called global artificial columns and local artificial columns. Global artificial columns are associated with periods t: they have coefficient 1 in constraints (9) for all i and a given t while their coefficient is zero for other periods. The global artificial column cost is a big constant, M, that we set to an estimate of the total cost: $M = T \sum_i f_i e_i$ where f_i is the visit frequency of customer i and e_i is an estimate of the cost of inserting i in a cluster. (We compute the average number of neighbors in a cluster including i as $n_i = \frac{W - d_i}{\overline{d}}$ where $\overline{d}$ is the average demand; then we set $e_i = \frac{1}{2} \sum_{j=1}^{n_i} c_{ij}$ having sorted the edges by increasing cost.)

There are $n\,T$ local artificial columns associated with a pair (i, t): they have a coefficient 1 in the associated constraint (9) and zero elsewhere. Their costs define an upper bound on the dual variable, π_{it}, associated to the corresponding constraint (9) – this can be seen by writing the dual of the master LP augmented with the local artificial columns. Hence, they are there to stabilize the column generation procedure by controlling the dual variable variation. However, it is not easy to estimate π_{it} values a priori. We can only come up with a rough estimation which is an estimate of the cost of covering customer i, denoted c_i, divided by the number of visits to i, i.e. $f_i\,T$. We set $c_i = f_i\,T\,(e_i + \frac{\sigma}{1+n_i})$, where σ stands for the fixed cost of opening a cluster, i.e. an estimate of the dual value of constraints (10), which we set to $\sigma = \frac{M}{T\,V}$.

The above estimation of dual variables have been compared to the actual dual values on the last iteration of the column generation procedure. Their achievement is barely to reflect the order of magnitude of the experimental values. When artificial columns remain in the solution on termination of the column generation procedure, their cost is multiplied by a factor $\alpha = 1.5$ and we return to column generation.

We have tried different strategies for managing the insertion of new columns. The straightforward implementation suggested by the presentation of the procedure made in Section 2 is to generate a single column (at most) for each period t: we solve the pricing subproblem heuristically and record the best primal solution found if it has negative reduced cost. We refer to it as the *single column* strategy. A natural alternative is to insert in the master all negative reduced cost columns that have been constructed in the course of the primal heuristic for the subproblem. We call it the *multiple columns* strategy.

Finally, we experimented with a more sophisticated strategy that consists in solving only one column generation subproblem per iteration instead of T when possible. We observed that, due to the *time symmetry*, dual price are not so different from one period to another. Hence, we tend to solve similar subproblems for each period. Instead, we developed a *cyclic generation* strategy that goes as follows. We iterate on periods t. If we get a negative reduced cost, we stop the iteration on t. We test this cluster q as a candidate column for the other periods and we insert in the master the variables λ_q^t associated with period t for which q defines a negative reduced cost column. The periods t are considered in cyclic order. On the first iteration, we solve the pricing subproblem associated with period 0. We iteratively update t, using $t = (t+1)\,mod\,T$, until we find negative reduced cost columns. If we do find some when $t = \tilde{t}$, we shall start the next round of column generation with period $t = (\tilde{t}+1)\,mod\,T$. The column generation procedure is stopped if all periods where considered in an iteration without success.

5 The rounding heuristic

We construct a primal integer solution to [M] (7-12) by rounding its LP (approximate) solution step by step in an iterative procedure. Having initialized the master as explained above, we optimize its LP relaxation using the column generation procedure with heuristic subproblem solver: i.e. the procedure stops when no more negative reduced cost columns can be found by our heuristic subproblem solver although some might still exists. Then, we collect the current primal solution to the restricted master LP (which is not necessarily optimal for the unrestricted problem). We fix to their current value all λ_q^t that are currently integer (if any) and we select one factional λ_q^t that shall be rounded up. This candidate is chosen as the largest fractional value λ_q^t. In case of ties, we select the cluster with smallest average cost $\frac{c_q}{||x^q||_1}$. The fixed variables define a partial solution that imposes some restrictions on scenario selection in (9) and reduces the number of remaining vehicles in (10).

The residual problem that remains after fixing the partial solution is submitted to the same treatment: we solve approximately its LP relaxation by column generation, round one λ_q^t in its LP solution. This procedure is applied iteratively until the approximate LP solution of the residual problem is integer and hence no more rounding is needed. In this case, a primal integer solution has been found. However, the procedure will be stopped earlier if the residual problem is infeasible: the remaining vehicles may not suffice to cover the remaining customer visit requirements. Such case is detected when the artificial columns remain in the solution even after their cost has been increased several times (we stop the procedure with a diagnostic of non-feasibility when the restricted master LP value reaches an a priori upper bound on the primal solution). It is to be observed that artificial columns also play a role to patch for temporary unfeasibility: after a rounding step, the current master column may not suffice to define a feasible LP solution to the residual problem while we have not generated appropriate clusters.

The rounding heuristic can be understood as depth-search in a branch-and-price tree with heuristic selection of the branch to explore. We may diversify the search by exploring more than one branch. We do this only at the primary selection node (the root node when no variables were integer in the initial master LP approximate solution). We explore up to 3 branches by considering alternative selection for the first λ_q^t that is rounded-up.

6 Numerical comparison of various implementation

The above implementation options in terms initialization, column generation strategies, number of passes in the rounding heuristic, and problem specifications have been compared numerically. Here we report on the trade off between solution quality and computing time (on a Pentium 4 Xeon, 3.06 GHz, 2 Go of RAM). The restricted master linear programs are solved using the commercial LP solver Xpress-MP (2001).

In terms of problem specifications, we have the option of relaxing or enforcing restrictions such as (i) limiting the neighborhood of a node to those customers for which $c_{ij} \leq \frac{\max_{ij} c_{ij}}{2}$ (the option of not using such limitation is called *unrestricted_ neighborhood*); (ii) imposing a 80% filling factor of cluster (an option referred to *well_filled_ clusters*); (iii) using a *client_ of_ reference* to break *time symmetry*; and (iv) setting workload upper bound W to that of the constructive heuristic solution or relaxing it to 120% of that value (in an option called *relaxed workload*).

Regarding initialization, the options are (i) to set the constructive heuristic so as to favor workload balancing (the default) or regionalization (in the latter case the workload bound W that is defined by the initial heuristic solution is not as tied); (ii) to use a single heuristic solution or the columns of all constructed primal solutions (the latter initialization is denoted *extra_ col*); (iii) to include global artificial columns ($g\_ a\_ c$) or local artificial columns ($l\_ a\_ c$). In terms of column generation strategies we have (i) the *single column* strategy, (ii) the *multiple columns* strategy for which we specify how many columns (at most) were recorded per subproblem and (iii) the *cyclic generation* strategy which can be combined with the *multiple columns* option.

To evaluate the contribution of each implementation strategy, we have defined a version of reference, denoted V00, and we defined other versions (V01 to V12) that differ by a single option (except for V10 and V12 that combine 2 interesting options). Table 1 defines the versions. The columns give the mode of initialization, the number of columns recorded for each subproblem, the number of passes in the rounding heuristic. The last column specifies other options used. Bold face characters emphasize the incremental difference with V00.

Version	initialization of master	nb col	nb pas.	other
V00	sol heur + g_a_v	3	1	
V01	sol heur + g_a_v	3	**3**	
V02	sol heur + g_a_v	**1**	1	
V03	sol heur + g_a_v + **extra_col**	3	1	
V04	sol heur + **l_a_v**	3	1	
V05	sol heur + g_a_v	3	1	**relaxed workload**
V06	sol heur **regionalization**+ g_a_v	3	1	
V07	sol heur + g_a_v	3	1	**cyclic generation**
V08	sol heur + g_a_v	3	1	**client_of_reference**
V09	sol heur + g_a_v	3	**3**	**well_filled_clusters**
V10	sol heur + g_a_v + **extra_col**	3	**3**	
V11	sol heur + g_a_v	3	1	**unrestricted_neighbor.**
V12	sol heur + g_a_v	3	**3**	**unrestricted_neighbor.**

Table 1: Definition of the different versions that have been implemented

The comparative tests are performed on a set of test problems that includes instances from the literature and randomly generated instances. We have selected six instances from Cordeau, Gendreau and Laporte (1997). Their size is defined in Table 2, where N is the number of clients, V is the number of vehicles, and T is the length of the time horizon. We also provide for further reference the cost of the operational solution optimized for distance computed by Cordeau, Gendreau and Laporte (1997) and their reported computing times. We have considered these instances to relate our work to the literature. But they do not reflect real life situations: there are academic test problems with specific symmetry (such as customer points drawing a tulip).

On the other hand, ten random instances with 50 clients, 3 vehicles and 5 periods were generated according to the profile of the industrial data used in Mourgaya and Vanderbeck (2005): frequency of visits are drawn in $\{\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{5}{5}\}$ with higher probability for $\frac{2}{5}$ and $\frac{3}{5}$; most clients ($\frac{2}{3}$ of them) have low integer demand in $\{1, 2\}$ while the other $\frac{1}{3}$ of the clients have demand in $\{3, \ldots, 8\}$; the number of visits per vehicle range in $[6, 12]$ with an average of 9; the clients coordinates are generated in a square of size 1000 but not uniformly: there are areas with greater concentration of clients.

Instance				Time	Best solution
name	N	V	T		
p01	50	3	2	1m06s	524.61
p02	50	3	5	0m36s	1322.87
p03	50	1	5	0m36s	524.61
p17	40	4	4	3m01s	1597.75
p25	51	3	6	4m20s	3777.15
p26	51	3	6	4m16s	3795.33

Table 2: Instances from Cordeau, Gendreau and Laporte (1997)

The computational results are reported in Table 3. For each random instance, we give the cost of the solution of the initial constructive heuristic (*heur*), and for each implementation, the cost of the solution given by the rounding heuristic. Then, Table 3 also provides average primal solution cost (*sol*), number of cases where the rounding heuristic improves on the constructive heuristic solution (*improv*), number of generated columns (*nb col*) and average computational time (*time*) taken by the rounding heuristic – excluding the initial call to the constructive heuristic which, for such problem size, is negligible anyway. To evaluate the quality of the solutions in terms of workload balancing, we also compute the relative workload deviation as follows : $\frac{W_{max}-W_{min}}{W_{max}} * 100$ (*dev W*). For the instances from the literature we only report the average results. The two last lines of the table provide global averages over the two groups of instances.

Comparing the various forms of initialization, we conclude by comparing V00 to V03 that adding extra columns of other initial heuristic solutions does not seem to bring any benefit on average (although some rounding heuristic solutions are better in V03). The use of local artificial columns in V04 has a limited stabilization effect (it may be because our dual value estimate are poor), but the overall results in terms of quality and time are slightly better. The comparison of average results for random instances is summarized in Table 4.

	heur	V00	V01	V02	V03	V04	V05	V06	V07	V08	V09	V10	V11	V12
Randomly generated instances														
inst-1	70990	53842	53842	53985	47620	70990	40196	42111	52875	54881	70990	47620	46724	46724
inst-2	104910	77413	77766	75710	74582	77957	38104	37127	77292	107559	79473	79564	49548	49895
inst-3	93980	77858	71431	71386	73298	72399	93980	48315	71813	74070	71521	71036	50785	51293
inst-4	111365	104856	105736	105426	111365	111365	48590	30314	104814	105944	106485	107003	64363	62791
inst-5	146677	146677	146677	146677	146677	146677	146677	27330	146677	151115	146677	146677	45479	40110
inst-6	111053	47458	47877	111053	111053	49392	42386	41043	48355	54808	52837	47413	49500	47764
inst-7	113340	53734	51791	52445	59523	54689	46382	48043	113340	54010	52537	51528	56345	54488
inst-8	157837	52450	51762	53962	50816	53546	157837	49313	52624	54263	53487	52654	56495	53565
inst-9	119298	41490	38290	40145	40340	45342	29715	26236	43348	33073	46369	38603	36681	39513
inst-10	149021	149021	48594	149021	149021	51565	46222	44367	45932	56310	45673	50872	55750	55555
Averages for the 10 random instances														
sol	117847	80480	69377	85981	86429	73392	69009	39420	75707	74603	72605	69297	51167	50170
nb col		2558	4193	1756	2974	2344	3159	3331	2852	2787	3219	4716	2989	5590
time		1m36s	3m24s	1m29s	1m55s	1m28s	9m20s	24m4s	1m30s	1m38s	1m56s	12m59s	1m29s	6m9s
dev W	07.29	18.34	16.70	13.32	14.22	17.10	45.15	61.98	14.62	11.27	11.34	16.77	15.05	16.37
improv.		8	9	7	6	7	7	10	8	9	9	9	10	10
Averages for the 6 instances from the literature														
sol	7767	4881	4881	5114	5161	5005	4864	4838	5547	5015	5058	5145	5560	5551
nb col		2903	3539	2201	3106	2078	2618	2922	3262	2705	3897	3616	3009	3241
dev W	03.34	06.57	05.20	06.37	05.86	04.75	24.53	12.09	03.44	06.56	03.80	05.86	03.46	03.46
improv.		5	5	4	4	5	5	5	4	5	4	4	4	4
Global averages														
sol	125670	52130	45191	55656	55954	47747	44955	26452	49397	48508	47275	45240	34064	33438
improv.		13	14	11	10	12	12	15	12	14	13	13	14	14

Table 3: Numerical comparison of the various implementations

	solution quality	nb columns	time
1 init primal solution	80480	2558	1m36s
all init primal solutions	86429	2974	1m55s
1 primal solution + l_a_c	73392	2344	1m28s

Table 4: Initialization strategies (average results for random instances)

strategy	solution quality	nb columns	time
single column	85980	1756	1m29s
multi column	80479	2558	1m36s
cyclic column	75706	2852	1m30s

Table 5: Column generation strategies (average results for random instances)

With regards to column generation strategies, we observe that the multiple columns strategy yields improved solution quality (going from V02 to V00 yields a 6.40 % solution improvement in average) while not increasing computing time. However, when we increased further the number of columns kept for each subproblem, we observed no more benefit in solution quality but larger computing times. The cyclic column generation strategy (combined with multiple columns) used in V07 gives further improvement in solution quality for the same average computing time. This analysis is summarized in Table 5.

Comparing V00 to V01, we note that using 3 passes instead of 1 for the rounding heuristic allows to get better quality solutions (13.80% improvement on average), but at the cost of increasing computing time by a factor 2 on average. See Table 6. Our Attempt to reduce the *time symmetry* by defining a *client_ of_ reference* as in version V08 seems to work (the procedure gives better results than V00 for an equivalent computational time). But this can also give solutions worse than those found with the constructive heuristic (see instance 2 for V08).

In terms of relaxing restrictions on the problem specification, we first analyze the consequence of relaxing the workload bound in V05 to 120% of its value in V00 or even further in V06 where the initial solution is optimized for regionalization and the workload bound is set to that of this initial solution. With a looser workload bound, the regionalization

	solution quality	nb columns	time
one pass solution	80480	2558	1m36s
multi pass for rounding heur	69376	4193	3m24s

Table 6: Rounding heuristic strategies

objective can be much better optimized by 14.25% from V00 to V05 and by 51.01 % from V00 to V06, but of course the workload balancing deteriorates: from a deviation 18.34% to 45.15% between V00 and V05 and to 70 % in V06. Moreover, the computational time increases because the set of solutions gets larger when the vehicle capacity is relaxed. With V06, each instance requires in average 25 minutes.

Other restrictions involve well filled clusters and truncated neighborhood. If we impose well filled clusters, as in V09, we get better solution quality (a 9.7% improvement) but a slight increase in computing time. Also note that the workload balance improves. In V11, we consider the complete graph instead of the restricted neighborhoods of V00. We get much higher quality solutions (a 36% improvement) and surprinsingly no increase in computing times. However, when the problem size increases, this option does become computationnally expensive.

We now look at combination of strategies. In V10, we use multi-passes in rounding while initializing the master with the columns of all available primal solutions. This gives quite good solutions (a 14 % improvement), the extra columns are useful to re-optimize the residual problem after fixing different partial solutions. But computing time increases drastically (a 700% increase): the large master LP's take the bulk of the computing time. The same conclusion holds if we combine multiple rounding heuristic passes and unrestricted neighborhoods as in V12: we observe a 37% solution improvement and a 280% time increase.

7 Validation of the rounding heuristic

The above tests show that the rounding heuristic can bring large improvements on the regionalisation criteria over a myopic constructive heuristic such as that of Mourgaya and Vanderbeck (2005). We now attempt to assess solution quality by comparing them to dual bounds. We shall also see how the algorithm behaves as problems grow larger. Finally, we take a critical look at the hierarchical approach, looking at operational solutions that can be generated from our planning and comparing them to solutions produced in the literature with a direct optimization of the operational model. However, to this end, we can only use a VRP solver to construct operational solutions from our plannings. As we mentioned in the introduction, such solver is inappropriate because it completely ignores the cluster formed in the first phase.

The Lagrangian dual bounds (13) computed in the course of our heuristic column generation approach have not proved helpful: although we attempted to compute good dual bounds on the pricing subproblems to plug in (13), the resulting Lagrangian bounds remain very low (often negative). In exact column generation procedures, it is commonly observed that the Lagrangian bounds only become tight when the procedure gets close to termination. The fact that our intermediate Lagrangian bound values remain meaningless here might mean that our heuristic column generation stops the procedure before entering the convergence tail where dual prices variation are less extreme and bounds (13) get tighter.

To obtain dual bounds, we proceed to solve the subproblem exactly using the commercial MIP solver Xpress MP (Xpress-MP, 2001). Then, we can carry on the column generation procedure to termination where the restricted master LP bound defines a valid dual bound. However, this can only be done for small size instances. In Table 7, we report on tests made on 10 randomly generated instances with 20 clients. We compare results obtained when solving the pricing subproblem exactly in implementation V01, with results obtained with V01 using the heuristic subproblem solver. The comparison also gives an idea of the value of an exact subproblem solver for the rounding heuristic.

The columns of Table 7 give the constructive heuristic solution (construct heur), the rounding heuristic solution when using a heuristic versus an exact pricing problem solver (rounding-heur-sp, rounding-exact-sp), the dual bound provided by the column generation procedure (dual bd) and the computing times of both procedures based on heuristic versus exact pricing solver (time-heur-sp , time-exact-sp). We also compute for each solution its deviation to the dual bound (in the parenthesis). When an exact subproblem solver is used, the optimality gap between dual bounds and rounding heuristic is 14.39%

	construct heur	rounding-heur-sp	rounding-exact-sp	dual bd	time-heur-sp	time-exact-sp
inst-1-c20	24297 (13.58 %)	17695 (17.28 %)	24297 (13.58 %)	21392	0m04s	0m25s
inst-2-c20	33707 (21.25 %)	33707 (21.25 %)	29486 (06.07 %)	27798	0m08s	0m45s
inst-3-c20	11567 (23.76 %)	9657 (09.51 %)	11567 (23.76 %)	8819	0m04s	0m36s
inst-4-c20	25779 (17.57 %)	24702 (12.66 %)	25064 (14.31 %)	21926	0m05s	0m22s
inst-5-c20	17821 (64.30 %)	17821 (64.30 %)	17821 (64.30 %)	10846	0m04s	0m21s
inst-6-c20	21982 (00.00 %)	21982 (00.00 %)	21982 (00.00 %)	21982	0m04s	0m11s
inst-7-c20	22780 (39.27 %)	22780 (39.27 %)	17483 (06.89 %)	16356	0m05s	0m39s
inst-8-c20	26648 (127.2 %)	12071 (02.93 %)	11727 (00.00 %)	11727	0m03s	0m09s
inst-9-c20	14651 (176.0 %)	6094 (14.83 %)	6094 (14.83 %)	5307	0m05s	0m20s
inst-10-c20	34980 (120.8 %)	34980 (120.8 %)	15858 (00.13 %)	15837	0m06s	0m20s
average	23421 (60.37%)	20149 (30.28%)	18138 (14.39%)	16199	0m05s	0m25s

Table 7: Random instances with 20 clients: exact versus heuristic subproblem solver

on average. With heuristic subproblem solver it is 30.28% on average. On the other hand, the improvement over the constructive heuristic solution is 40 % in average. However, we suspect that the duality gaps are smaller for larger instances where the solutions space is wider and a rounding heuristic such as ours has more chance to identify a good solution.

Table 8 analyzes the behavior of our algorithm for increasing instance sizes. Randomly generated instances with respectively 20, 50 and 80 customers are considered. Values for the constructive heuristic solution, the rounding heuristic solution and the computing times are provided. The computing times for solving the master and the subproblems are specified. The computational time quickly gets out of hand when the size increases. The combinatorial explosion in the number of generated columns makes the resulting master impractical to solve: the computational bottleneck is solving the master LPs with Xpress-MP. We observed that master LP solution time can vary a lot from one iteration to another. Incidently, the dual LP solver is much faster than the primal for these problems.

Hence, to obtain the results of Table 8, we had to truncate the procedure by limiting the number of column generation iterations for the instances with 80 customers: 500 iterations for the initial column generation procedure and 50 in each re-optimization phases after rounding a column. As an indication, an instance with 80 customers was run without any iteration limit: it was stopped after 22h09m, 31844 columns had been generated, 98% of the computing time was spent in solving these large master linear programs; the initial constructive heuristic solution was 352713 and the rounding solution was 135589 (an improvement of 67% to compare against a 22% improvement for the truncated rounding heuristic). This seems to indicate that the rounding heuristic is capable of producing much better solutions than a constructive heuristic, but it needs to be get close to LP optimality before rounding.

	construct heur	rounding heur	nb col	total time	time master	time subpb
inst-1-20	11568	9658	376	4s00t	0s69t	1s08t
inst-2-20	25780	24702	474	5s01t	0s82t	1s06t
inst-3-20	26648	12072	335	2s70t	0s30t	0s69t
inst-4-20	14651	6094	578	5s31t	0s93t	1s36t
inst-5-20	24297	17696	395	4s20t	0s54t	0s85t
inst-1-50	93980	71431	4235	3m46s92t	1m5s57t	1m31s93t
inst-2-50	113340	51791	6234	6m25s43t	2m56s46t	2m05s02t
inst-3-50	157837	51762	5305	4m10s28t	1m16s12t	1m46s67t
inst-4-50	119298	38290	3997	2m36s23t	0m37s54t	1m08s86t
inst-5-50	149021	48594	4769	4m2s82t	1m33s71t	1m27s19t
inst-1-80	353181	143749	30659	1h18m16s95t	0h30m09s11t	0h29m08s08t
inst-2-80	321879	321879	25398	0h41m27s66t	0h14m56s07t	0h17m09s43t
inst-3-80	208109	118512	27411	0h58m28s37t	0h28m50s73t	0h19m39s26t
inst-4-80	248448	184820	31468	1h01m16s43t	0h27m03s74t	0h20m30s56t
inst-5-80	310797	167398	19580	7h03m30s32t	6h45m15s47t	0h11m45s32t

Table 8: Average results on larger instances: the algorithm is truncated.

Finally, in Table 9, the operational solutions that we obtained by applying a phase 2 solver to our phase 1 solutions are compared to the operational solutions of Cordeau et al. (1997) reported in Table 2. Phase 2 is solved using the column generation based heuristic for the VRP developed by Michel (2005). Solutions are compared according to three criteria : (i) regionalization (this criteria is available for our solutions; we had to compute it for the solutions from the literature); (ii) the relative deviation of the workload (available for our solutions and computed for the solutions from the literature); (iii) distance travel (this criteria is available for the solutions from the literature, while, for our solutions, it is obtained by applying the VRP solver to the phase 2 problem). As expected, solutions from the rounding heuristic have a better regionalization criteria (an improvement of 16% in average) and workload balance (an improvement of 77% in average) but the distance criteria is worse compared to those optimized for distance from the outset (a 20 % increase).

	Tactical planning				Operation. sol.	
	Regionalization		Workload Dev %		Distances	
inst	sol round	sol lit	sol round	sol lit	sol round	sol lit
p01	3023	3804	4.58	6.87	788	525
p02	5529	6297	25.34	81.87	1675	1323
p03	4376	4256	1.28	6.87	796	525
p17	3401	5554	0.00	0.00	1660	1598
p25	6478	7601	0.00	30.00	4401	3777
p26	6478	7660	0.00	15.00	4403	3795
average	4881	5862	5.2	23.43	2287	1924

Table 9: Comparison to operational solutions on instances from the literature

8 Conclusions

We proposed a hierarchical approach to the periodic vehicle routing problem. While this problem was treated as an operational problem (optimizing travel distance) in the literature, our two-phase approach allows to bring in alternative criteria at the tactical planning level that are quite important in the eyes of industrial practitioners: namely regionalization (serving issues such as driver specialization) and workload balancing (serving issues such as robustness). It is not clear how to evaluate the trade-off between these conflicting objectives that do not directly translate into monetary terms. (How much is worth an improved quality in robustness or regionalization compared to an increase in travel distance.) Moreover, an ad-hoc second phase needs to be developed to measure the full impact of serving alternative objectives in the first phase. No previous results on this tactical planning model were available in the literature, nor did we have industrial solutions to compare our solutions to.

The core of the paper is the LP-based rounding heuristic applied to a column generation re-formulation. We developed primal and dual bounds on the column generation subproblems, heuristic setting of an initial dual solution to the master program, column generation strategies such as cyclic generation, and used other features such as restricted neighborhoods, a lower bound on cluster filing, or a symmetry breaking attempt. The numerical comparative study brings light on the impact of the various features. This can be inspiring for research on related problems. The rounding heuristic has global optimization properties: it has been shown to have potential for substantial improvement in solution quality compared to a constructive greedy procedure (that is somewhat myopic). Its computing time is reasonable for problems with 50 customers: around 1 to 6 minutes to be compared with similar computing times of a Tabu Search method reported in Table 2. However, with the current implementation, the computing time quickly becomes too large for practical purposes as problems get larger (mostly because of the combinatorial explosion in the number of columns).

In an effort to extend this approach to larger problems, one could expect progress from (a) further developing the heuristic subproblem solver: in particular, a local exchange procedure could be useful; (b) implementing a dynamic management of the columns that remain in the master: as solving restricted master LP's is the identified computational bottleneck, the aim would be to define a reasonable size *core master* by intelligently deleting columns based on reduced cost information and /or their history of being part of previous LP solutions; this intensification of the search could be complemented by a diversification procedure in the form of an exchange of columns; (c) relying on a computationally cheaper solver for the master: a sub-gradient procedure can also provide intermediate fractional primal solutions in the form of a combination of previous subproblem solutions; it can be the base of a rounding heuristic as in Caprara et al. (1999), or its use can be restricted to the initial phase followed by using LP tools on a core master problem; (d) finding appropriate ways to reduce the *time symmetry*.

References

Angelelli, E . and Speranza, M. G. (2002), The periodic vehicle routing problem with intermediate facilities, *European Journal of Operational Research* 137, 233-247.

Ball, M.O. (1988), Allocation/Routing : models and algorithms, in B.L Golden et A.A Assad (eds), *Vehicle Routing : Methods and Studies*, 199-221, North-Holland, Amsterdam.

Caprara, A. Fischetti, M. and Toth, P. (1999), A Heuristic Method for the Set Covering

Problem, *Operations Research* 47, 5, 730-743.

Chabrier, A. Danna, E. and Le Pape, C. (2002), Coopération entre génération de colonnes avec tournées sans cycle et recherche locale appliquée au toutage de véhicules. Actes JNPC 2002.

Chao, M., Golden, B.L and Wasil, E. (1995), An improved heuristic for the period vehicle routing problem, *Networks* 26, 25-44.

Christofides, N. and Beasley, J. E. (1984), The period Routing Problem, *Networks* 14, 237-256.

Cordeau, J.F. Gendreau, M. and Laporte, G. (1997), A tabu search Heuritic for periodic and multi depot vehicle routing problems, *Networks* 30, 105-119 .

Desaulniers, G. Desrosiers, J. and Solomon M.M. (2005) editors, *Column Generation*, Kluwer.

Elhedhli, S. and J.-L. Goffin (2005). Efficient Production-Distribution System Design, *Management Science*, 51:7, 1151-164.

Feillet, D. Dejax, P. Gendreau M. and Gueguen, C. (2004), An exact algorithm for the Elementary Shortest Path Problem with Resource Constraints: application to some vehicle routing problems, *Networks* 44(3), 216-229.

Fukasawa, J. Lysgaard, J. Poggi de Aragão, M. Reis, M. Uchoa, E. and Werneck, R.F. (2004), Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem, in G. Nemhauser and D. Bienstock (eds), Integer Programming and Combinatorial Optimization, *Lecture Notes in Computer Science*, Vol. 3064.

Gaudioso, M. and Paletta, G. (1992), A heuristic for the periodic vehicle routing problem, *Transportation Science* 26(2), 86-92.

Kellerer, H., U. Pferschy, and D. Pisinger (2004). "Knapsack Problems", Springer.

Irnich, S. and Villeneuve, D. (2003). The Shortest Path Problem with Resource Constraints and k-Cycle Elimination for $k \geq 3$, Technical Report G-2003-55, GERAD, 2003.

Michel S. (2005), PhD thesis in preparation, Université Bordeaux 1.

Mourgaya, M. (2004). *The periodic vehicle routing problem: planning before routing*, PhD thesis, Université Bordeaux 1.

Mourgaya, M. and Vanderbeck F. (2005). Periodic Vehicle Routing Problem: classification and heuristic. To appear in RAIRO Operations Research.

Russel, R. and Gribbin, D. (1991), A multiphase approach to the period routing problem, *Networks* 21, 747-765.

Russel, R. and Igo, W. (1979), An assignment routing problem, *Networks* 9, 1-17.

Tan, C.C.R. and Beasley, J.E. (1984), A heuristic algorithm for the period vehicle routing problem, *Omega Int J of Management Science* 12(5), 497-504.

Toth, P. and Vigo, D. (2002), *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications.

Vanderbeck, F. (1999). Computational Study of a Column Generation algorithm for Bin Packing and Cutting Stock problems *Math. Prog.* A, 86:565-594.

Vanderbeck F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48:1, pp111-128.

Vianna, D.S. Ochi, L.S. and Drummond, L.M.A. (1998), A parallel hybrid evolutionary metaheuristic for the period vehicle routing problem, *Lecture Notes in Computer Science* 1586, 183-191.

Xpress-MP (2001), *User guide and Reference Manual*, Release 12, Dash Optimization.