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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Large Neighborhood Search with Constraint Programming for a Vehicle Routing Problem with Synchronization Constraints

Hossein Hojabri
Michel Gendreau
Jean-Yves Potvin
Louis-Martin Rousseau

June 2016

CIRRELT-2016-31

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

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Large Neighborhood Search with Constraint Programming for a Vehicle Routing Problem with Synchronization Constraints

Hossein Hojabri[1,2], Michel Gendreau[1,3], Jean-Yves Potvin[1,2,*], Louis-Martin Rousseau[1,3]

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

[2] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[3] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-Ville, Montréal, Canada H3C 3A7

Abstract. This paper considers an extension of the vehicle routing problem with time windows, where the arrival of two vehicles at different customer locations must be synchronized. That is, one vehicle has to deliver some product to a customer, like a home theater system, while the crew on another vehicle must install it. This type of problem is often encountered in practice and is very challenging due to the interdependency among the vehicle routes, but has received little attention in the literature. A constraint programming-based adaptive large neighborhood search is proposed to solve this problem. The search abilities of the large neighborhood search and the constraint propagation abilities of constraint programming are combined to efficiently determine the feasibility of any proposed modification to the current solution. Numerical results are reported on instances derived from benchmark instances for the vehicle routing problem with time windows with up to 200 customers.

Keywords: Vehicle routing, synchronization, large neighborhood search, constraint programming.

Acknowledgements. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

---

* Corresponding author: Jean-Yves.Potvin@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2016

1 Introduction

The Vehicle Routing Problem with Multiple Synchronization constraints (VRPMS) can be used to model many real-world applications, like home care delivery, aircraft fleet assignment, vehicle routing and scheduling, and forest operations, as reported in Rousseau et al. (2013). Here, we consider a distribution problem where the crew in a delivery vehicle cannot unload or install goods at a customer location without another vehicle or another crew with the required capabilities. That is, the routes of two different types of vehicles must be synchronized at a certain number of customer locations. A dynamic version of this problem for pickup vehicles (i.e., items are collected at customer locations rather than being delivered) is addressed in Rousseau et al. (2013). Due to the dynamic nature of the problem, new customer requests are inserted one by one in the current routes, as they occur. Constraint programming, a computational paradigm based on constraint propagation, is used to determine if an insertion is feasible or not. Since only one customer is considered at a time, this approach proved to be viable. Surprisingly and to the best of our knowledge, no solution approach, neither exact nor heuristic, has been proposed for the static version of the problem where all customers are known in advance. This a priori knowledge is required in the case of delivery applications, because the vehicles are loaded before they depart from the depot. The goal of this paper is thus to provide a contribution in this regard. The solution approach will exploit both Adaptive Large Neighborhood Search (ALNS) and Constraint Programming (CP).

Apart from the paper of Rousseau et al. (2013), only a few papers in the literature are somewhat related to our work. Ioachim et al. (1999) propose a column generation approach for an aircraft fleet assignment and routing problem where a number of flight subsets during the weekdays must depart at the same time. The master problem is a set partitioning problem with synchronization constraints, while the subproblem is a shortest path problem with time windows and linear costs on the time variables. An extension of this work for a periodic airline fleet assignment with time windows is also reported in Bélanger et al. (2006).

Bredström and Rönnqvist (2008) propose a mixed integer programming (MIP) model for a vehicle routing and scheduling problem with temporal precedence and synchronization constraints. The authors solve the problem through an optimization-based heuristic. Schmid et al. (2010) report a MIP for the delivery of concrete to construction sites from several plants by a fleet of heterogeneous vehicles. Two hybrid approaches are introduced to solve the problem by combining an exact algorithm with a variable neighborhood search (VNS). El Hachemi et al. (2011) consider an application in the forest industry where trucks and log-loaders must be synchronized. A decomposition approach is presented to solve a weekly problem in two phases. The first phase determines the assignment of forest areas to woodmills. Then, the daily routing and scheduling for the transportation of logs is done with either a constraint-based local search alone or a hybrid method consisting of CP coupled with the constraint-based local search. Other types of synchronization requirements are also found in urban mass transit systems where drivers must change buses at relief points (Freling et al. 2003, Haase et al. 2001) or in public transit rail systems where passengers must transfer several times to reach their destination (Wong et al. 2008).

Li et al. (2005) report an integer programming model for a manpower allocation problem with time windows and job-teaming constraints. To perform a job, the required workers must show up together at the job location before it can be executed. Furthermore, nobody can leave before the job is completed. The authors propose two construction heuristics embedded within a simulated annealing framework. They provide comparisons with optimal solutions produced by a commercial solver and lower bounds obtained with a network flow model on their own set of test instances. Dohn et al. (2009) also solve the problem with a branch-and-price algorithm where branching on the time variables automatically imposes synchronization. A related problem is addressed in Rasmussen et al. (2012) where a MIP model is developed to schedule home care staff with different synchronization requirements.

The interested reader will find in Drexl (2012) a comprehensive survey on synchronization, with a particular emphasis on vehicle routing applications. In this survey, different types of synchronization requirements are presented (spatial, temporal, etc.), as well as different applications (dial-a-ride, school bus routing, truck-and-trailer, etc.), but nothing quite similar to our problem.

The remainder of this paper is organized as follows. Our problem is first introduced in section 2. A description of the solution methodology is provided in sections 3 and 4. A few refinements to this methodology are proposed in section 5. Then, numerical results are reported in section 6. Finally, concluding remarks are made in section 7.

2 Problem definition and formulation

Our problem can be seen as an extension of the Vehicle Routing Problem with Time Windows (VRPTW) with the addition of synchronization requirements. In a standard VRPTW, the routes are independent, in the sense that a modification to a given route has no impact on the other routes. This is not the case for the VRPMS because the routes are interdependent due to the synchronization requirements at certain customer locations.

Let us assume that two different types of vehicles are available, called regular (delivery) and special vehicles. The regular vehicles deliver goods to customers and have limited capacity. The special vehicles do not deliver goods but rather provide a service related to the goods delivered by the regular vehicles. The set of customers is divided into two subsets: regular customers visited only by regular vehicles, and special customers visited by both a regular and a special vehicle. A regular customer is represented by a single vertex in the transportation network, while a special customer is represented by two vertices, one for the delivery by the regular vehicle and another for the service provided by the special vehicle.

A regular vertex has a demand and a time window, while a special vertex has only a time window, which is defined relative to the delivery time of the regular vehicle. More precisely, if t is the delivery time at the regular vertex of a special customer, then the service of the special vertex must begin within the time window $[t-\delta,\ t+\gamma]$, where δ and γ are parameters. Different synchronization requirements can be represented with different values of δ and γ. For example, if the service of the special vehicle cannot start before the delivery, then the time window's lower bound of the special vertex is set to t (i.e., $\delta = 0$); if the service of the special vehicle cannot start before the end of the delivery, then the time window's lower bound of the special vertex is set to t plus the service time of the regular vehicle (i.e., δ is set to minus the service time of the regular vehicle), etc.

The problem then consists in constructing routes for the two types of vehicles such that:

1. Each route begins and ends at a single depot;
2. Each regular customer is served in the route of exactly one regular vehicle;
3. Each special customer is served in the route of exactly one regular vehicle and one special vehicle;
4. The total demand on the route of a regular vehicle cannot exceed its capacity;
5. A regular vehicle must start its delivery at a regular or special customer within the time window of the corresponding regular vertex;
6. A special vehicle must start its service at a special customer within the time window of the corresponding special vertex, which is defined relative to the delivery time of the regular vehicle at the regular vertex;
7. A regular (special) vehicle is allowed to wait up to the lower bound of the time window of the regular (special) vertex if it arrives earlier.

The objective is to minimize the total distance traveled by all vehicles (regular and special).

In the following, a small example is given before providing a more formal CP-based formulation of our problem.

2.1 Example

We consider a small example where the load and time window constraints of the regular vehicles are relaxed. We have two regular vehicles $\{v_1^r, v_2^r\}$, one special vehicle $\{v_1^s\}$, a single depot 0 and 5 customers $\{1, 2, 3, 4, 5\}$, two of which are special customers, namely $\{1, 2\}$. In the underlying network, vertex 0 is the depot, vertices 1 to 5 correspond to regular vertices, while vertices 6 and 7 correspond to the special vertices of customers 1 and 2. Therefore, the vertex pairs (1,6) and (2,7) require synchronization. In Figure 1, each pair is aggregated into a single grey circle. Without loss of generality, we assume that the service or dwell time is equal to 1 and $\delta = \gamma = 0$ for every special vertex.

In the solution shown, solid arcs are used to represent the routes of regular vehicles and dashed arcs for the route of the special vehicle. Each arc is labeled with its travel time and each vertex i is labeled with the vehicle's arrival time t_i. The return time of each vehicle to the depot, denoted t_0, is also indicated on the last arc of the corresponding route. The routing plan is the following:

$v_1^r : 0, 5, 1, 0$
$v_2^r : 0, 2, 3, 4, 0$
$v_1^s : 0, 7, 6, 0$
$t_1 = 8$, $t_2 = 3$, $t_3 = 8$, $t_4 = 12$, $t_5 = 4$, $t_6 = 8$.

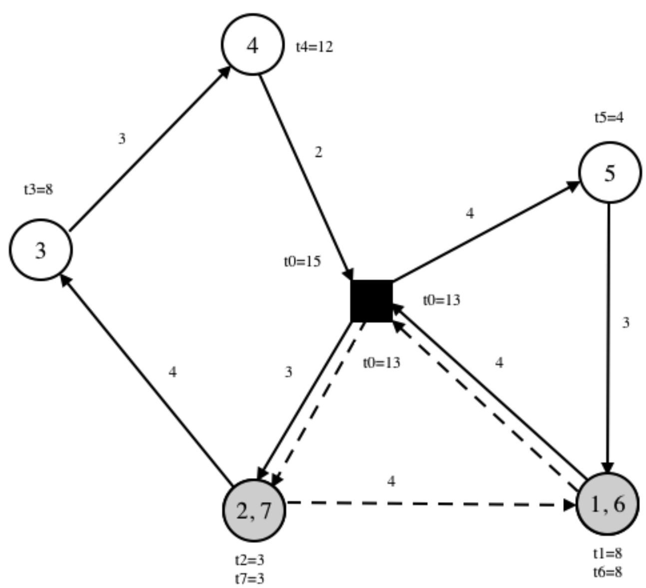

Figure 1: VRPMS example

To better understand the synchronization requirements, we follow the route of regular vehicle v_1^r. Its route starts with regular customer 5 where the delivery begins at $t_5 = 4$. After spending 1 time unit to serve the customer, it takes 3 more time units to arrive at special customer 1 where a special vehicle is also needed. The special vehicle arrives at the same time $t_1 = t_6 = 8$. Thus, vehicle v_1^r is perfectly synchronized with the special vehicle v_1^s. Then, vehicle v_1^r departs from special customer 1 at time 9 for the depot. Finally, it arrives at the depot at time 13. Note that the special vehicle v_1^s is already synchronized with the regular vehicle v_2^r at special customer 2 at time $t_2 = t_7 = 3$.

2.2 Model

In this section, a CP-based formulation of the problem is proposed. This formulation is at the core of the reconstruction operator of our ALNS (see Section 3). Some familiarity with constraint programming is assumed, otherwise the reader is referred to Rousseau et al. (2013).

Let $G = (N, A)$ be a complete and directed graph with vertex set $N = V_r \cup V_s \cup V_r^+ \cup V_s^+ \cup V_r^- \cup V_s^-$, where V_r and V_s are the sets of regular and special vertices, respectively, with $V_r \cup V_s = V$. We recall that one vertex is associated with each regular customer and two vertices with each special customer, namely a regular vertex in V_r and a special vertex in V_s. Also, V_r^+ and V_r^- contain multiple copies of the depot, one for the start and end of the route of each regular vehicle. That is, $|V_r^+|$ and $|V_r^-|$ are both equal to the number of regular vehicles. Similarly, V_s^+ and V_s^- contain multiple copies of the depot for the start and end of the route of each special vehicle. Again, $|V_s^+|$ and $|V_s^-|$ are both equal to the number of special vehicles. We also define $V_r^+ \cup V_s^+ = V^+$ and $V_r^- \cup V_s^- = V^-$. Finally, with each arc $(i, j) \in A$ is associated a non-negative distance d_{ij} and travel time t_{ij}. The notation for the CP model is summarized below.

Sets and parameters:
d_{ij}: Distance between vertices i and j, $i,\ j \in N, i \neq j$;
t_{ij}: Travel time between vertices i and j, $i,\ j \in N, i \neq j$;
q_i: Demand of vertex $i \in V_r$;
d_i: Service or dwell time at vertex $i \in V$;
a_i: Lower bound of the time window at regular vertex $i \in V_r$;
b_i: Upper bound of the time window at regular vertex $i \in V_r$;
δ_i: Decrement to derive the lower bound of the time window at special vertex $i \in V_s$;
γ_i: Increment to derive the upper bound of the time window at special vertex $i \in V_s$;
r_i: Regular vertex associated with special vertex $i \in V_s$;
V_r: Set of regular vertices;
V_s: Set of special vertices;
V: set of regular and special vertices;

K_r: Set of regular vehicles;
K_s: Set of special vehicles;
K: Set of regular and special vehicles;
Q: Capacity of each regular vehicle $k \in K_r$.

Note that d_{ij} and t_{ij} are fixed to ∞ when arc (i, j) connects two copies of the depot (except when the two copies stand for the start and end of the same vehicle route, in which case $d_{ij} = 0$; these arcs are used when there are empty vehicle routes in the solution). Also, the demand and the service or dwell time at each copy of the depot are set to 0

Variables:
s_i: Successor of vertex $i \in V \cup V^+$;
k_i: Vehicle visiting vertex $i \in N$;
t_i: Start time of service at vertex $i \in N$;
l_i: Vehicle load after serving vertex $i \in V_r \cup V_r^+ \cup V_r^-$.

Constraints:

General constraints:

$$AllDifferent(s_{i:i \in V \cup V^+}) \tag{1}$$
$$NoSubTour(s_{i:i \in V \cup V^+}) \tag{2}$$
$$i \in V_r \cup V_r^+ \Rightarrow s_i \in V_r \cup V_r^- \tag{3}$$
$$i \in V_s \cup V_s^+ \Rightarrow s_i \in V_s \cup V_s^- \tag{4}$$
$$i \in V_r \cup V_r^+ \cup V_r^- \Rightarrow k_i \in K_r \tag{5}$$
$$i \in V_s \cup V_s^+ \cup V_s^- \Rightarrow k_i \in K_s \tag{6}$$
$$s_i = j \Rightarrow k_j = k_i, \qquad i \in V \cup V^+ \tag{7}$$

Capacity constraints:

$$l_i = 0, \qquad i \in V_r^- \tag{8}$$
$$l_i \leq Q, \qquad i \in V_r \cup V_r^+ \tag{9}$$
$$s_i = j \Rightarrow l_j = l_i - q_j, \qquad i \in V_r \cup V_r^+ \tag{10}$$

Time window and synchronization constraints:

$$t_i = 0, \qquad i \in V^+ \tag{11}$$
$$a_i \leq t_i \leq b_i, \qquad i \in V_r \tag{12}$$
$$t_{r_i} - \delta_i \leq t_i \leq t_{r_i} + \gamma_i, \qquad i \in V_s \tag{13}$$
$$s_i = j \Rightarrow t_i \leq t_j - t_{ij} - d_i, \qquad i \in V \cup V^+ \tag{14}$$

Objective function:

$$\min \sum_{i \in V \cup V^+} d_{i,s_i} \tag{15}$$

In a solution, each vertex must have exactly one predecessor and one successor. The nature of variables s_i is such that any vertex i can only have one successor, but we must also ensure that it has only one predecessor. To do so, it must be forbidden for two different vertices to have the same successor, which corresponds to Constraint (1) called *AllDifferent*. Constraint (2) prohibits subtours, see the detailed description in Pesant et al. (1998). Constraints (3)-(6) define variables s_i and k_i for regular and special vertices. Constraint (7) states that a vertex and its successor must be served by the same vehicle. Constraints (8)-(10) impose load and capacity restrictions. Without loss of generality, vehicles are assumed to start their routes at the depot at time 0 through constraint (11). Constraint (12) imposes time window constraints on regular vertices. The synchronization requirements between regular and special vehicles at special vertices is stated in Constraint (13). Constraint (14) imposes the coherence of each route schedule. Finally, the objective function (15) minimizes the total distance traveled by all vehicles.

3 ALNS

ALNS extends the large neighborhood search framework of Shaw (1998), a problem-solving approach which can also be related to the ruin-and-recreate principle of Schrimpf et al. (2000). The basic idea is to search for a better solution at each iteration by destroying a part of the current solution and by reconstructing it in a different way. When solving VRPs, a new solution is obtained by first removing a number of vertices and then by reinserting these vertices into the solution. Typically, a number of different operators are available and the adaptive extension chooses an operator in a randomized, but informed, way at each iteration. That is, the selection probability of an operator is derived from its associated weight, which is adjusted during the search depending on the operator's previous successes and failures.

A noteworthy feature of our ALNS is the use of CP to reinsert the removed customers. Thus, a systematic exploration of feasible ways to reinsert the vertices is performed.

The ALNS algorithm can be described as follows.

1. Generate an initial solution s
2. $s^* \leftarrow s$

3. While a stopping criterion is not met do:
   3.1 Adaptively select a destruction operator and apply it to s
   3.2 Reinsert customers with CP
   3.3 If a new solution $s' \neq s$ is returned by CP then
   $s \leftarrow s'$
   If s' is better than s^* then $s^* \leftarrow s'$
4. Return best solution s^*

The main components of this algorithm will be detailed in the following. Then, further refinements to this basic ALNS scheme will be reported.

3.1 Initial Solution

The routes of the delivery vehicles are first created by inserting regular vertices using the well-know I1 insertion heuristic of Solomon (1987), originally developed for the VRPTW. Then, the time windows for the special vertices are defined through the delivery times in the regular routes. Solomon's insertion heuristic is reapplied on the special vertices to create routes for the special vehicles. At this point a complete solution is obtained.

3.2 Adaptive operator selection

A destruction operator removes a number of vertices from the current solution at each iteration of the ALNS. Since more than one destruction operator is typically available, the adaptive mechanism is aimed at choosing the destruction operator in a way that accounts for its previous outcomes. A weight is associated with each operator for this purpose. Let us assume that we have h operators, each with a weight w_j, $j = 1, ..., h$. The removal operator i is then selected with probability

$$\frac{w_i}{\sum_{j=1}^{h} w_j}, i = 1, ..., h.$$

That is, the probability of selecting a given operator increases with its weight. Starting with a unit weight for each operator, the weights are updated after a number of consecutive iterations called a segment. The weight of operator i at the start of a given segment sg is based on the one used in the previous segment $sg - 1$ and is computed as follows:

$$w_i^{sg} = \gamma \cdot w_i^{sg-1} + (1 - \gamma) \cdot \frac{\pi_i^{sg-1}}{n_i^{sg-1}},$$

where n_i^{sg-1} is the number of times operator i was used in segment $sg - 1$, γ is a value between 0 and 1 and π_i^{sg-1} is the score of operator i at the end of segment

$sg-1$. Parameter γ controls the inertia in the weight update equation. When γ is close to 1, the history prevails and the weights do not change much. Conversely, when γ is close to 0, the update is driven by the most recent score.

The score, which is reset to zero at the beginning of each segment, is incremented when operator i is used at a given iteration t to produce a new solution. More precisely, the new score at iteration $t+1$ becomes

$$\pi_i^{t+1} = \pi_i^t + \begin{cases} \sigma_1 & \text{if a new best solution has been produced,} \\ \sigma_2 & \text{if the solution produced is better than the current solution,} \end{cases}$$

where σ_1 and σ_2 are parameters. Based on preliminary experiments, the number of iterations in a segment was set to 50 in our implementation, while γ, σ_1 and σ_2 were set to 0.5, 2 and 1, respectively.

3.3 Destruction operators

A total of 10 destruction operators were considered in this study, but only two were kept at the end, called *Route removal* and *Historical node-pair removal*, after the results of a sensitivity analysis experiment reported in Section 6. The reader is referred to Pisinger and Ropke (2007) and Shaw (1998) for a description of the eight previously reported operators that were excluded from the final implementation, namely *Cluster removal*, *Random removal*, *Worst removal*, *Historical route-pair removal*, *Distance-oriented related removal* and *Time-oriented related removal*. In the following, we describe *Route removal*, *Historical node-pair removal*, *Synchro Vertex removal* and *Synchro Route removal*, because they were specifically designed for our problem (although the last two were excluded from the final implementation). They are all based on a parameter which is the number of vertices to be removed. To simplify the description, this parameter is always denoted NV, but it is clear that a different parameter value can be associated with each operator.

3.3.1 Route removal

This operator removes vertices in the same route(s). First, a route is randomly selected and all its vertices are removed. If NV is not reached yet, all remaining vertices are sorted according to their distance to a randomly chosen vertex among the previously removed ones. Then, one of the closest vertices is selected, using a randomization factor r^D. More precisely, the vertex in position $\lfloor r^D * ListSize \rfloor$ in the sorted list is selected, where $ListSize$ is the number of vertices in the list, r is a random number between 0 and 1, and D is a parameter (set to 3 in our experiments). The selected vertex, as well as the other vertices in the same route, are then removed. This procedure is repeated until NV vertices are removed. The randomization factor provides a form of diversification but, more importantly, when one of the two vertices of a special customer is randomly selected, the closest vertex is necessarily the other vertex from the pair (at distance 0). Thus, without randomization, the operator would behave like the *Synchro Route removal* operator (see below). In the computational results, *Route removal* proved to be the best performing operator.

3.3.2 Historical node-pair removal

This operator removes vertices based on historical information. That is, a score $g(i,j)$ (initially set to infinity) is assigned to each arc (i,j). This score corresponds to the objective value of the best solution found up to now in which vertices i and j are visited consecutively. Then, for each vertex i, we consider the value $g_{i,\mathrm{succ}_i} + g_{\mathrm{pred}_i,i}$, where $succ_i$ and $pred_i$ are the predecessor and successor of vertex i in the current solution, respectively. The NV vertices with the largest values are then removed.

3.3.3 Synchro Vertex removal

First, $\lceil NV/2 \rceil$ special vertices are randomly selected and removed with their corresponding regular vertices. If the number of special vertices is less than $\lceil NV/2 \rceil$, then the remaining vertices are randomly chosen among the regular ones. Given that a special vertex must be synchronized with a regular vertex, the idea is that better insertion places are more likely to be found in the following reinsertion phase if both vertices are removed together. Unfortunately, this operator did not provide good results and was not kept in the final implementation.

3.3.4 Synchro Route removal

This operator focuses on the routes of special vehicles. First, a route is randomly selected among all the routes served by special vehicles. Then, the route is emptied by removing all its (special) vertices along with their corresponding regular vertices. This is repeated until NV vertices are removed. Like Synchro Vertex removal, this operator did not perform well and was not kept at the end.

The following section is devoted to the reinsertion of the removed vertices.

4 CP-based reinsertion

Here, a partially destroyed solution is reconstructed using CP, as initially proposed in Pesant and Gendreau (1996, 1999) and Rousseau et al. (2002), using the model presented in section 2.2. CP has proven useful for many routing variants, like the Traveling Salesman Problem (TSP), the TSP with time windows, as well as the VRP and VRPTW. The addition of synchronization constraints make our problem even more suitable for a CP-based approach, see Rousseau, et al. (2013). For example, the insertion of a special vertex in the route of a special vehicle causes a delay not only to the special vertices that are visited later along this route, but also to the regular vertices which are associated with these special vertices. That is, the schedule of regular vehicles is also modified. The constraint propagation capabilities of CP can thus be fully exploited here.

In the reconstruction phase, only a subset of vertices needs to reinserted in the solution, which restricts the domain of feasible values for each variable. For example, if vertex i is part of the partial solution (it has not been removed), then the value of the successor variable s_i in the model of section 2.2 can only be chosen among the current successor in the partial solution or one of the removed vertices. Once the domain of each variable is appropriately set, CP then implicitly considers all feasible ways to reinsert the removed vertices in the solution. However, due to the propagation and pruning abilities of CP, the number of possibilities can be drastically reduced using appropriate variable and value selection heuristics to guide the reinsertion. In our implementation, we instantiate first the more constrained variables with their less constraining value. Here, a variable s_i associated with a removed vertex has a smaller number of possible successors, and is more constrained, if it is associated with a vertex far from the other vertices. The value of the selected variable is then set to the closest vertex in the current partial solution. This approach proved to be slightly better than the autosearch feature of the ILOG CP Optimizer.

5 Further refinements

To improve the performance of our ALNS, additional refinements have been introduced. They are described in the following.

5.1 Constrained objective function

Since CP looks for the optimal reinsertion of the removed vertices at each iteration of ALNS, very long run times are to be expected. To alleviate this problem, the minimization of the objective function is taken off from the CP model and introduced as a constraint, to produce a pure constraint satisfaction problem:

$$NewObj < \alpha \times CurrObj \tag{16}$$

where $CurrObj$ and $NewObj$ are the values of the current and new solutions, respectively, and parameter α has a positive value which must be less than or equal to 1. The choice of α is very important. If too small, CP might run for a long time for nothing because there is no feasible solution. If too large, CP will return a solution just a bit better than the current solution. After preliminary experiments, α was finally set to $(1 - 0.02/(nbIter^{\lambda} + 1.1))$ with $\lambda = 0.45$. This formula depends on the iteration number $nbIter$, because larger improvements are expected to be found at the start. As the search progresses, smaller improvements are looked for.

5.2 Solution acceptance

It is known that the search process of ALNS can be enhanced by allowing worse solutions to be accepted, see Pisinger and Ropke (2007). Similarly, we force CP to look for a worse solution with some probability *Prob*. This probability increases with the number of consecutive CP failures (a failure occurs when CP cannot find a feasible and improved reinsertion of the removed vertices). The formula is the following, where p is a parameter:

$$Prob = (nbFail/(nbFail + 1))^p$$

In practice, the following constraint replaces constraint (16) with probability *Prob* (where w and r are parameters):

$$CurrObj < NewObj < CurrObj \times (1 + \frac{nbFail^w}{r}) \tag{17}$$

It should be noted that CP searches for increasingly worse solutions as the number of consecutive failures *nbFail* increases. If a solution is returned, *NbFail* is reset to 0. In our experiments, $p = 4$, $w = 1.5$ and $r = 1,000$.

5.3 Variable neighborhood size

To help intensify the search, the number of vertices to be removed NV is increased by 1 each time the reinsertion through CP fails. Conversely, NV is decreased by one each time the reinsertion succeeds. However, the NV value must stay within the bounds of the interval $[Min_{NV}, Max_{NV}]$, which is set to $[16, 26]$ for the 25- and 50-customer instances, $[13, 23]$ for the 100-customer instances, and $[8, 18]$ for the 200-customer instances in the computational experiments. The initial value of NV is randomly selected within the appropriate interval.

5.4 CP search time

Experimentally, we found that our ALNS is more efficient when CP is stopped after a given time limit $CPlimitTime$ (i.e., if CP has not returned a solution after that amount of time, it is unlikely that it will find one if it is run longer). This approach leads to much shorter run times and also produces better solutions because ALNS can explore a larger fraction of the search space. $CPlimitTime$ is allowed to vary in the interval $[MinCPlimit, MaxCPlimit]$, which is set to $[1s, 6s]$ in our experiments. The time limit is initially equal to $MinCPlimit$ and is incremented by one when CP fails to find a solution and the current value is not $MaxCPlimit$. It is decremented by one when CP returns a solution and the current value is not $MinCPlimit$.

6 Computational experiments

This section first describes our test instances followed by a sensitivity analysis experiment on the destruction operators. Then, numerical results are reported.

6.1 Test instances

Given that the static version of our VRPMS is addressed for the first time, we generated test instances with $n = 25$, 50, 100 and 200 customers from the VRPTW instances of Solomon (1987) and Homberger and Gehring (1999). In these Euclidean instances, the travel time is the same as the distance. There are also different classes of instances. The customers are randomly located in the instances of type R, clustered in the instances of type C, and both randomly located and clustered in the instances of type RC. Furthermore, instances of type 1 have a short scheduling horizon, while those of type 2 have a long one. Overall, there are six different classes of instances: R1, R2, C1, C2, RC1 and RC2. A variable number of instances is found in each class, for a total of 56 instances for each size.

Some additional parameters are required to transform the original VRPTW instances into VRPMS instances: Sp, which is the percentage of special customers; δ_i and γ_i which are used to define the time window of each special vertex $i \in V_s$.

Test instances were created from the VRPTW instances by setting parameter Sp to 5%, 25% and 50%. More precisely, the number of special customers is equal to $\lceil Sp \times n \rceil$. Then, the first customer is a special customer and the following special customers are chosen using a constant interval defined by $1/Sp$. For example, if $n = 100$ and $Sp = 5\%$, then $1/Sp = 20$ and the five special customers are 1, 21, 41, 61, and 81.

The VRPTW instances are designed in such a way that the instances of larger size are obtained by adding customers to the instances of smaller size. For example, the instances of size 50 are obtained by adding 25 customers to the instances of size 25. For a given Sp value, the same applies to the special customers in the VRPMS instances. That is, all special customers in the instances of size 25 are also found in the corresponding instances of size 50. With regard to synchronization, δ_i and γ_i were set to 0 and 10, respectively, for each special vertex $i \in V_s$ (as in Rousseau et al. (2013)).

6.2 Sensitivity analysis

A sensitivity analysis experiment was performed to select a good combination of operators among the 10 available destruction operators. For this purpose, we used the 100-customer instances with 25 special customers. Let us denote LB the average solution value obtained with the original ALNS implementation with $h = 10$ operators. Then, at each iteration, ALNS is run h times with $h-1$ operators by removing each operator in turn. The best among the h runs is then compared with LB. If the average solution value is better than LB, the missing operator is detrimental because an improvement is observed without it. Consequently, this operator is put apart and the procedure is repeated with the $h-1$ remaining operators with LB

reset to the improved average solution value. Otherwise, when the average solution value of the best among the h runs is worse than LB, each missing operator degrades solution quality and the procedure is stopped. In our case, the procedure ran until all operators were dismissed except *Route removal* and *Historical node-pair removal*.

An incremental approach was then considered. First, ALNS was run 10 times with a single destruction operator each time. The best operator proved to be *Route removal* and this operator was kept. Then, ALNS was run 9 times by adding a different operator to *Route removal*. At this point, *Historical node-pair removal* with *Route removal* proved to be the best pair and the average solution value was also better than the one obtained with *Route removal* alone. Thus, *Historical node-pair removal* was kept. The next step showed that the addition of a third operator always led to a degradation in solution quality. Thus, we stopped again with *Route removal* and *Historical node-pair removal*.

6.3 Numerical results

For the computational experiments, our algorithm was run on a 3.07GHz Xeon(R) X5675 and was stopped after 20,000 iterations. The abbreviations used in Table 1 are the following:

- Size: Number of customers
- Class: Class identifier
- nbSyn: Number of special customers
- InObj: Total distance traveled by the regular and special vehicles in the initial solution
- InRgObj: Total distance traveled by the regular vehicles in the initial solution
- InSpObj: Total distance traveled by the special vehicles in the initial solution
- FnObj: Total distance traveled by the regular and special vehicles in the final solution
- FnRgObj: Total distance traveled by the regular vehicles in the final solution
- FnSpObj: Total distance traveled by the special vehicles in the final solution
- InNbRgV: Number of regular vehicles used in the initial solution
- InNbSpV: Number of special vehicles used in the initial solution
- FnNbRgV: Number of regular vehicles used in the final solution
- FnNbSpV: Number of special vehicles used in the final solution

- RunTime: Computation time in seconds
- LitBestObj: Objective value (total distance) of the best known VRPTW solution in the literature
- LitBestnbVeh: Number of vehicles of the best known VRPTW solution in the literature

The numerical results reported in Table 1 correspond to averages taken over all instances in a given class. For example, each value under *LitBestObj* corresponds to the average of the best objective value (distance) reported in the literature over all VRPTW instances in a given class. It is a lower bound for the corresponding value under *FnRgObj*, which is the average of the best objective value (distance) produced by our ALNS on the VRPMS instances when considering only the routes of the regular vehicles. In the last column of the table, the gap in percentage between these two values is calculated as:

$$Gap\% = 100 \times ((FnRgObj - LitBestObj)/FnRgObj)$$

Due to the synchronization constraints, the total distance traveled by the regular vehicles in our solutions is obviously greater than the total distance in the corresponding VRPTW solutions. Admittedly, the gap between the two values is larger than we expected for instances with only a few special customers. On the other hand, the gap does not increase that much with a larger number of special customers (in a few cases, it even decreases). For example, if we consider the instances of size 50 in class R1, the gap with 3 special customers is 8.19%, but it only increases to 10.67% and 12.94% with 13 and 25 special customers, respectively. That is, only a few special customers with synchronization requirements is enough to break the structure of the VRPTW solutions. However, additional special customers are then easily accommodated. The percentage of improvement over the initial solution also indicates that the ALNS performs well, although its performance degrades when the problem size increases. This is indicated by the decrease in the percentage of improvement over the initial solution and the increase in the gap with the best known VRPTW solution. This trend is easily explained because the number of customers removed from the current solution at each iteration gets smaller as the problem size increases (see section 5.3), due to the combinatorial explosion in the number of possible reinsertions. Still, this is the first time that CP is used to solve VRPMS instances of that size. It should be noted that Rousseau et al. (2013) address the dynamic version of VRPMS, so that CP handles only a few customers at a time (i.e., customers who have already been visited or those who have not called yet for service are not part of the current solution).

7 Conclusion

In this work, a VRPTW variant incorporating synchronization constraints between two different types of vehicles was addressed. The ALNS framework was coupled with CP to solve the problem. Under this hybrid scheme, CP was used to reconstruct partially destroyed solutions. Numerical results on instances derived from standard VRPTW benchmark instances empirically demonstrated the optimization capability of the proposed algorithm. New developments could look at ways to increase the efficiency of CP through finer adjustments within the ALNS framework (e.g., adaptive CP time limit). We also want to get closer to the real-world by considering the integration of time-dependent travel times that would account for rush hours, for example.

Acknowledgements. Financial support was provided by the Canadian Natural Sciences and Engineering Research Council. This support is gratefully acknowledged.

References

Bélanger, N., Desaulniers, G., Soumis, F., Desrosiers, J., Periodic Airline Fleet Assignment with Time Windows, Spacing Constraints and Time Dependent Revenues. European Journal of Operational Research 175, 1754-1766, 2006.

Bredström, D., Rönnqvist, M., Combined Vehicle Routing and Scheduling with Temporal Precedence and Synchronization Constraints. European Journal of Operational Research 191, 19-29, 2008.

Dohn, A., Kolind, E., Clausen, J., The Manpower Allocation Problem with Time Windows and Job-Teaming Constraints: A Branch-and-Price Approach. Computers and Operations Research 36, 1145-1157, 2009.

Drexl, M., Synchronization in Vehicle Routing: A Survey of VRPs with Multiple Synchronization Constraints. Transportation Science 46, 297-316, 2012.

El Hachemi, N., Gendreau, M., Rousseau, L.M., A Heuristic to Solve the Synchronized Log-truck Scheduling Problem. Computers and Operations Research 40, 666-673, 2013.

Freling, R., Huisman, D., Wagelmans, A.P.M., Models and Algorithms for Integration of Vehicle and Crew Scheduling. Journal of Scheduling 6, 63-85, 2003.

Haase, K., Desaulniers, G., Desrosiers, J., Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. Transportation Science 35, 286-303, 2001.

Homberger, J., Gehring, H., Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. INFOR 37, 297-318, 1999.

Ioachim, I., Desrosiers, J., Soumis, F., Bélanger, N., Fleet Assignment and Routing with Schedule Synchronization Constraints. European Journal of Operational Research 119, 75-90, 1999.

Li, Y., Lim, A., Rodrigues, B., Manpower Allocation with Time Windows and Job-Teaming Constraints. Naval Research Logistics52, 302-311, 2005.

Pesant, G., Gendreau, M., A Constraint Programming Framework for Local Search Methods. Journal of Heuristics 5, 255-279, 1999.

Pesant, G., Gendreau, M., A View of Local Search in Constraint Programming. In Principles and Practice of Constraint Programming, Freuder, E. C., eds., Constraint Programming 96, Lecture Notes in Computer Science 1118, Springer Berlin-Heidelberg, 353-366, 1996.

Pesant, G., Gendreau, M., Potvin, J.Y., Rousseau, J.M., An Exact Constraint Logic Programming Algorithm for the Traveling Salesman Problem with Time Windows. Transportation Science 32, 12-29, 1998.

Pisinger, D., Ropke, S., A General Heuristic for Vehicle Routing Problem. Computers & Operations Research 34, 2403-2435, 2007.

Shaw P., Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In Principles and Practice of Constraint Programming, Goos, G., Hartmanis, J., van Leeuwen, J., eds., Constraint Programming 98, Lecture Notes in Computer Science 1520, Springer, Berlin-Heidelberg, 417-431, 1998.

Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J., The Home Care Crew Scheduling Problem: Preference-based Visit Clustering and Temporal Dependencies, European Journal of Operational Research 219, 598-610, 2012.

Rousseau, L.M., Gendreau, M., Pesant, G., The Synchronized Vehicle Dispatching Problem. INFOR 51, 76-83, 2013.

Rousseau, L.M., Gendreau, M., Pesant, G., Using Constraint-based Operators to Solve the Vehicle Routing Problem with Time Windows. Journal of Heuristics 8, 43-58, 2002.

Rousseau, L.M., Gendreau, M., Pesant, G., Focacci, F., Solving VRPTWs with Constraint Programming Based Column Generation. Annals of Operations Research 130, 199-216, 2004.

Schmid, V., Doerner, K., Hartl, R., Salazar-González, J., Hybridization of Very Large Neighborhood Search for Ready-Mixed Concrete Delivery Problems. Computers and Operations Research 37, 559-574, 2010.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G., Record Breaking Optimization Results Using the Ruin and Recreate Principle. Journal of Computational Physics 159, 139-171, 2000.

Solomon, M. M., Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research 35, 254-265, 1987.

Table 1: Numerical results

			Objective Values						Number of vehicles					Literature			
			Initial solution			Final solution			Initial solution		Final solution						
Size	Class	nbSyn	InObj	InRgObj	InSpObj	FnObj	FnRgObj	FnSpObj	InNbRgV	InNbSpV	FnNbRgV	FnNbSpV	RunTime	LitBestObj	LitBestnbVeh	Imp %	Gap%
25	R1	2	621.66	559.13	62.53	546.20	483.66	62.53	5.92	1.00	5.33	1.00	10419.69	463.45	4.92	12.14	4.18
	R2		577.89	515.35	62.53	455.78	392.88	62.90	4.18	1.00	2.82	1.09	12642.79	350.25	2.73	21.13	10.85
	C1		280.53	228.33	52.20	244.56	193.75	50.81	3.20	1.20	3.00	1.00	12202.74	190.59	3.00	12.82	1.63
	C2		356.23	287.77	68.46	284.80	216.34	68.46	2.38	1.00	2.00	1.00	10566.73	214.45	2	20.05	0.87
	RC1		572.33	407.01	165.32	519.52	354.62	164.89	3.88	1.50	3.38	1.25	8080.07	350.24	3.25	9.23	1.24
	RC2		558.28	393.80	164.47	485.69	321.01	164.68	3.00	1.00	2.88	1.13	10258.97	319.28	3	13.00	0.54
25	R1	7	832.64	559.13	273.51	700.86	492.61	208.26	5.92	3.83	5.25	3.00	10877.69	463.45	4.92	15.83	5.92
	R2		779.83	515.35	264.48	597.32	408.03	189.28	4.18	2.09	2.64	1.55	12755.30	350.25	2.73	23.40	14.16
	C1		360.15	215.47	144.68	322.60	201.88	120.72	3.11	2.89	3.00	2.00	9164.76	190.59	3	10.43	5.59
	C2		402.83	287.77	115.06	326.33	218.39	107.94	2.38	1.00	2.00	1.00	11079.08	214.45	2	18.99	1.81
	RC1		698.27	407.01	291.25	625.21	359.47	265.74	3.88	3.00	3.38	2.63	9091.15	350.24	3.25	10.46	2.57
	RC2		693.40	393.80	299.59	580.98	339.93	241.05	3.00	2.00	3.00	1.50	10917.50	319.28	3	16.21	6.07
25	R1	13	950.12	559.13	390.99	821.76	487.21	334.55	5.92	4.67	5.50	3.67	10750.62	463.45	4.92	13.51	4.88
	R2		913.19	515.35	397.84	729.40	416.95	312.45	4.18	2.55	2.73	2.00	12526.27	350.25	2.73	20.13	16.00
	C1		389.50	215.47	174.03	360.24	208.56	151.68	3.11	3.00	3.00	2.56	9467.61	190.59	3	7.51	8.61
	C2		476.72	287.77	188.95	382.06	220.76	161.30	2.38	1.13	2.00	1.00	10948.81	214.45	2	19.86	2.86
	RC1		731.55	407.01	324.54	657.93	361.75	296.18	3.88	3.38	3.38	3.00	8875.68	350.24	3.25	10.06	3.18
	RC2		709.59	393.80	315.78	603.32	331.88	271.44	3.00	2.25	2.88	2.13	11562.53	319.28	3	14.98	3.80
50	R1	3	1056.21	949.14	107.07	922.42	834.44	87.98	10.67	1.58	9.17	1.00	24211.29	766.13	7.75	12.67	8.19
	R2		893.48	796.94	96.53	774.30	687.07	87.23	5.73	1.18	5.18	1.00	28466.10	634.03	4.11	13.34	7.72
	C1		526.62	442.08	84.54	472.58	390.35	82.23	5.50	1.33	5.00	1.00	19701.93	361.69	4.56	10.26	7.34
	C2		603.43	494.47	108.96	486.97	382.62	104.35	3.57	1.57	3.00	1.00	28671.21	357.50	2.75	19.30	6.57
	RC1		1117.91	903.25	214.66	970.82	778.10	192.72	8.50	2.50	7.38	1.75	22790.59	730.31	6.50	13.16	6.14
	RC2		900.77	710.50	190.28	788.02	598.18	189.84	4.75	1.13	4.63	1.00	25899.25	568.65	4.33	12.52	4.94
50	R1	13	1383.40	949.14	434.26	1192.34	857.62	334.72	10.67	5.58	8.92	3.33	24862.12	766.13	7.75	13.81	10.67
	R2		1234.54	796.94	437.60	1052.73	730.28	322.45	5.73	2.64	4.82	1.82	28353.46	634.03	4.11	14.73	13.18
	C1		710.67	415.80	294.87	669.96	410.54	259.42	5.33	4.78	5.22	3.67	22919.57	361.69	4.56	5.73	11.90
	C2		779.74	494.47	285.27	640.86	423.36	237.50	3.57	2.29	3.29	1.86	28425.44	357.50	2.75	17.81	15.56
	RC1		1426.83	903.25	523.58	1246.51	811.15	435.36	8.50	5.38	7.75	4.00	23099.72	730.31	6.50	12.64	9.97
	RC2		1268.13	710.50	557.64	1023.87	620.86	403.01	4.75	3.13	4.75	1.88	28383.38	568.65	4.33	19.26	8.41
50	R1	25	1615.13	949.14	665.99	1388.63	879.97	508.66	10.67	7.83	9.55	5.27	24031.43	766.13	7.75	14.02	12.94
	R2		1469.22	796.94	672.28	1253.27	769.69	483.58	5.73	3.64	4.82	2.64	28575.68	634.03	4.11	14.70	17.63
	C1		786.88	415.80	371.08	725.79	402.27	323.52	5.33	5.00	5.22	4.56	22933.04	361.69	4.56	7.76	10.09
	C2		850.11	477.88	372.22	715.60	409.73	305.88	3.50	2.88	3.13	2.25	28159.50	357.50	2.75	15.82	12.75
	RC1		1626.42	903.25	723.17	1356.21	804.34	551.87	8.50	7.00	7.50	5.25	23106.40	730.31	6.50	16.61	9.20
	RC2		1316.27	710.50	605.77	1159.71	663.17	496.54	4.75	4.13	4.63	2.88	28600.91	568.65	4.33	11.89	14.25
100	R1	5	1716.51	1533.15	183.36	1578.35	1414.86	163.48	17.17	2.83	16.58	2.33	22910.22	1201.34	11.92	8.05	15.09
	R2		1425.63	1241.35	184.28	1320.72	1165.88	154.84	7.09	2	7.09	1.27	26502.14	951.03	2.73	7.36	18.43
	C1		1228.52	1032.97	195.55	1102.41	925.82	176.58	11.44	1.89	11.11	1.56	15756.68	828.38	10	10.27	10.52
	C2		999.41	784.54	214.87	927.12	728.47	198.65	5	2	4.75	1.5	19759.78	589.86	3	7.23	19.03
	RC1		1949.66	1720.98	228.68	1753.47	1545.3	208.17	16	2.75	15.38	2	21250.34	1384.16	11.5	10.06	10.43
	RC2		1643.43	1427.36	216.07	1478.95	1275.38	203.57	7.63	2.38	7.5	1.38	27211.93	1119.24	3.25	10.01	12.24
100	R1	25	2190.18	1533.15	657.03	1990.33	1481.76	508.56	17.17	8.83	16.5	5.33	26160.64	1201.34	11.92	9.12	18.92
	R2		1856.53	1241.35	615.18	1728.42	1199.49	528.94	7.09	3.91	6.82	3.55	32734.15	951.03	2.73	6.9	20.71
	C1		1749.11	1032.97	716.13	1648.22	1016.94	631.28	11.44	7.78	11.11	6.33	26989.21	828.38	10	5.77	18.54
	C2		1331.89	784.54	547.35	1191.42	760.81	430.61	5	3.25	4.88	2.88	20793.23	589.86	3	10.55	22.47
	RC1		2667.44	1720.98	946.47	2309.58	1636.15	673.43	16	10.5	15.38	5.75	23832.58	1384.16	11.5	13.42	15.40
	RC2		2222.07	1427.36	794.71	2016.23	1346.43	669.8	7.63	4.75	7.25	3.63	30263.5	1119.24	3.25	9.26	16.87
100	R1	50	2564.23	1533.15	1031.08	2352.13	1534.71	817.42	17.17	12.58	16.5	9	25280.26	1201.34	11.92	9.12	21.72
	R2		2201.18	1241.35	959.83	2102.83	1280.96	821.88	7.09	5.27	6.73	4.91	37187.46	951.03	2.73	6.9	25.76
	C1		1937.9	1032.97	904.92	1846.06	1027.96	818.1	11.44	10.11	11.33	9.22	23994.94	828.38	10	5.77	19.42
	C2		1407.95	812.11	595.84	1280.11	775.05	505.06	5.29	3.71	4.86	3	18857.2	589.86	3	10.55	23.89
	RC1		3045.9	1720.98	1324.93	2765.25	1671.02	1094.22	16	13.75	15.75	10	22925.14	1384.16	11.5	13.42	17.17
	RC2		2545.56	1427.36	1118.2	2412.31	1428.16	984.15	7.63	6	7.38	5.13	29500.99	1119.24	3.25	9.26	21.63
200	R1	10	5512.69	4948.78	563.91	5007.83	4490.66	517.17	19.50	3.10	19.10	2.70	73859.84	3612.36	18.2	9.16	19.56
	R2		4878.61	4104.26	774.35	4574.85	3896.42	678.43	8.70	2.30	8.60	2.30	100078.21	2929.41	4	6.23	24.82
	C1		4190.81	3282.92	907.89	4025.22	3279.62	745.60	20.40	4.50	20.30	3.70	70183.02	2718.41	18.9	3.95	17.11
	C2		3117.46	2452.38	665.07	3023.82	2389.47	634.35	9.40	3.30	9.40	3.10	110125.09	1831.59	6	3.00	23.35
	RC1		5031.35	4177.94	853.41	4771.49	4075.15	696.34	23.00	4.80	22.60	3.60	89337.45	3178.68	18	5.16	22.00
	RC2		4147.28	3410.05	737.23	3965.46	3251.23	714.23	8.90	2.10	8.60	2.00	117207.21	2536.2	4.3	4.38	21.99
200	R1	50	6745.06	4948.78	1796.28	6158.05	4670.64	1487.42	19.50	9.00	19.30	7.10	103458.27	3612.36	18.2	8.70	22.66
	R2		6123.93	4104.26	2019.67	5757.94	4007.35	1750.59	8.70	4.90	8.50	4.40	131889.12	2929.41	4	5.98	26.90
	C1		5381.98	3282.92	2099.06	5219.41	3324.07	1895.34	20.40	13.20	20.40	12.00	100708.04	2718.41	18.9	3.02	18.22
	C2		3838.84	2452.38	1386.45	3754.99	2410.63	1344.36	9.40	6.80	9.10	6.70	148177.25	1831.59	6	2.18	24.02
	RC1		6249.50	4177.94	2071.56	5872.90	4222.21	1650.69	23.00	11.70	22.80	8.40	123685.93	3178.68	18	6.03	24.72
	RC2		5137.58	3447.64	1689.93	4928.41	3405.71	1522.70	9.11	5.11	8.89	4.67	142865.16	2536.2	4.3	4.07	25.53
200	R1	100	8141.25	4948.78	3192.47	7524.63	4669.21	2855.42	19.50	14.00	19.10	12.10	102755.50	3612.36	18.2	7.57	22.63
	R2		6943.14	4104.26	2838.88	6592.95	3943.41	2649.55	8.70	6.20	8.50	6.10	139656.54	2929.41	4	5.04	25.71
	C1		6232.89	3282.92	2949.96	6055.69	3254.89	2800.80	20.40	17.60	20.30	17.40	76668.36	2718.41	18.9	2.84	16.48
	C2		4439.16	2452.38	1986.78	4341.19	2430.60	1910.59	9.40	7.20	9.30	7.20	90417.87	1831.59	6	2.21	24.64
	RC1		7520.83	4177.94	3342.89	7114.68	4380.03	2734.66	23.00	18.20	22.80	13.50	145373.80	3178.68	18	5.40	27.43
	RC2		5865.87	3410.05	2455.81	5697.54	3342.74	2354.80	8.90	7.40	8.80	7.30	163038.10	2536.2	4.3	2.87	24.13