CIRRELT

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

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation

An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips

Nabila Azi
Michel Gendreau
Jean-Yves Potvin

February 2010

CIRRELT-2010-08

Bureaux de Montréal :

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

Bureaux de Québec :

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

www.cirrelt.ca

An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips

Nabila Azi[1,2,*], Michel Gendreau[1,3], Jean-Yves Potvin[1,2]

[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, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

Abstract. The vehicle routing problem with multiple trips consists in determining the routing of a fleet of vehicles where each vehicle can perform multiple routes during its workday. This problem is relevant in applications where the duration of each route is limited, for example when perishable goods are transported. In this work, we assume that a fixed-size fleet of vehicles is available and that it might not be possible to serve all customer requests, due to time constraints. Accordingly, the objective is first to maximize the number of served customers and then, to minimize the total distance traveled by the vehicles. An adaptive large neighborhood search, exploiting the ruin-and-recreate principle, is proposed for solving this problem. The various destruction and reconstruction operators take advantage of the hierarchical nature of the problem by working either at the customer, route or workday level. Computational results on Euclidean instances, derived from well-known benchmark instances, demonstrate the benefits of this multi-level approach.

Keywords. Vehicle routing, multiple trips, adaptive large neighborhood search, multi-level.

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

---

* Corresponding author: Nabila.Azi@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec,
Bibliothèque et Archives Canada, 2010

1 Introduction

Studies on the classical vehicle routing problem (VRP) and its many variants represent a significant fraction of the operations research literature [20]. However, most studies assume that a vehicle can do no more than a single route during a given scheduling period (typically, a day). In this work, we consider a vehicle routing problem where a vehicle is allowed to perform multiple routes (VRPM) starting from and ending at a central depot. Furthermore, each customer has an associated time window for service (VRPMTW). The vehicle can arrive at the customer location before the lower bound of the time window, and must wait up to this lower bound, but is not allowed to arrive after the upper bound. These problems arise, for example, in e-grocery services where perishable goods are delivered to customers who must be on-site. This application leads to multiple short vehicle routes, where the last customer in each route must be visited within a given time limit from the start of the route.

One of the first study on the VRPM is found in Taillard et al. [19]. First, different solutions to the classical VRP are produced with a tabu search heuristic. Then, the routes obtained are combined to form workdays for the vehicles by heuristically solving a bin packing problem, an idea previously found in [7]. Due to the heuristic nature of the bin packing solutions, one or more workdays might extend over the scheduling period, leading to overtime which is penalized in the objective. In [6], the authors report a tabu search heuristic for solving a real-world application where additional characteristics are taken into account, like an heterogeneous fleet of vehicles with limited access restrictions, maximum legal driving time per day (penalized, if there is overtime), etc... In a later work [5], the same authors apply a streamlined version of their tabu search heuristic on the vehicle routing instances with multiple trips used in [19]. The results show that their algorithm is competitive with Taillard et al.'s method. A multi-phase heuristic, similar in spirit to [19], but using a more involved bin packing heuristic, is proposed in [11]. In this work, a savings-based construction method and a (giant) tour partitioning method are first applied to generate a pool of VRP solutions. The routes obtained are then used to produce solutions to the VRPM through the bin packing heuristic. It is worth noting that different exchange heuristics are also applied to improve both the VRP and VRPM solutions. A hybrid genetic algorithm for the VRPM is reported in [12], where the genetic operators are specifically designed for multi-trip vehicle routing solutions. A problem-solving method based on an adaptive memory made of elite solutions [14], a concept related to the population-based approach of genetic algorithms, is also reported in [10]. In [1], a site-dependent periodic vehicle routing problem is described where a vehicle can perform more than one route per day over an horizon of a few days. The problem is addressed with a tabu search heuristic.

Recently, an adaptive guidance algorithm was proposed for a variant of the VRPMTW. This variant stems from a real-world distribution problem where different types of pair-wise incompatible commodities must be delivered to customers [4]. A decomposition approach generates simpler subproblems which are then solved with specific heuristics. Two adaptive guidance mechanisms are defined : one is based on penalization of critical time intervals (i.e., intervals where many routes are strongly active) and improvement of routes with critical commodities (i.e., commodities that do not seem to be well packed across multiple routes). Finally, an exact approach for the VRPMTW is reported in [2].

Here, the pervasive issue of a (possible) inability of the fixed-size fleet of vehicles to accommodate all customers, due to the finite time horizon, is addressed by visiting only a subset of customers rather than by allowing overtime. This approach is required to account for the time window constraints that may preclude the existence of any feasible solution, even by considering overtime. An Adaptive Large Neighborhood Search (ALNS) [13, 15] is proposed to address the VRPMTW. The ALNS is designed to account for the hierarchical nature of the problem through the application of operators that modify the current solution at the customer, route and workday levels. It is empirically shown that this multi-level approach leads to much better solutions than the classical customer-based approach. The rest of the paper is organized as follows. The problem is first precisely defined in section 2. Then, our algorithm is described in section 3. Computational results are reported in section 4 and a conclusion follows.

2 Problem definition

Our problem can be stated as follows. We have a directed graph $G = (V, A)$, where $V = \{0, 1, 2, ..., n\}$ is the set of customer vertices with the depot 0 and where A is the arc set. With each customer $i \in V - \{0\}$ is associated a gain g_i, a demand q_i, a service or dwell time s_i and a time window $[a_i, b_i]$, where a_i and b_i are the earliest and latest time, respectively, to begin the service (with $a_0 = 0$ and $b_0 = \infty$). Thus, a vehicle has to wait if it arrives at customer i before a_i. With each arc $(i, j) \in A$ is associated a distance d_{ij} and a travel time t_{ij} (in this work, distances and travel times are the same). We also have a set $K = 1, 2, ..., m$ of vehicles, each of capacity Q, to deliver goods from the depot to customers. The duration of each route is limited by forcing the last customer to be served within t_{max} time units of the route start time. This restriction leads to short routes that must be combined and sequenced to form vehicle workdays. Also, a setup time for loading the vehicle, noted σ_r, is associated with each route r in the solution. Here, the setup time is proportional to the sum of service times over all customers in the route.

The objective considered is hierarchical: first, the number of served customers is maximized (by maximizing the total gain, assuming a gain of 1 for every customer); second, for the same number of customers, the total distance traveled by the vehicles is minimized. A complete mathematical description of this problem can be found in [2].

3 Problem-solving methodology

ALNS extends the large neighborhood search framework of Shaw [17], a problem-solving approach which can also be related to the ruin-and-recreate principle [16]. 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 typically obtained by first removing a number of customers and then by reinserting these customers into the solution. In general, a number of destruction and reconstruction operators are available and a destruction-reconstruction pair is randomly chosen at each iteration. In the adaptive extension, a weight is associated with each operator and the selection probability of an operator is related to its weight, which is adjusted during the search based on its past successes.

The problem structure, where a vehicle workday is made of a sequence of trips and where each trip is made of a sequence of customers, is exploited by applying destruction operators at the workday, route and customer levels, in this order. This approach is described in pseudo-code in Algorithm 1. The idea is to go from gross (high-level) to fine (low-level) refinements. In this algorithm, an initial feasible solution s is first constructed. Then, a destruction and a reconstruction operator are probabilistically chosen based on their current weights. The destruction operators are first selected at the workday level, then at the route level and finally at the customer level, where each level is explored for a number of consecutive iterations. At each iteration, a new solution is obtained by applying the destruction operator followed by the reconstruction operator on the current solution s. The new solution s' is then submitted to an acceptance rule. If accepted, the new solution becomes the current solution, otherwise the current solution does not change. After exploring a given level, the weights associated with the applied operators are adjusted. This is repeated until a termination criterion is met and the best solution found $s*$ is returned. In the following, each component of this algorithmic framework will be explained in detail.

3.1 Construction of the initial solution

An insertion heuristic, where all routes and workdays grow in parallel, is used to obtain an initial solution, see Algorithm 2. Based on a given ordering, each customer is considered in turn and inserted at its best feasible insertion place over every route in every workday, including a new (empty) workday if one is still available. In this work, the best insertion place corresponds to the smallest detour in distance, where the detour is $d_{ji} + d_{il} - d_{jl}$ for the insertion of customer i between j and l. If there is no feasible insertion place for customer i, then each route is considered in turn and split into two subroutes, with an additional copy of the depot between the two subroutes. This is illustrated in Figure 1, where the square stands for the depot and the black circle for the customer to be inserted. Once the original route is split, the insertion of the customer can take place in any of the two new routes. Each route is split in every possible way (i.e., at every customer location along the route) to find the best insertion place. If there is still no feasible insertion place, then customer i

Algorithm 1 ALNS

1. construct a feasible solution s;
2. $s* \leftarrow s$;
3. initialize weights;
4. **while** the stopping criterion is not met **do**
  4.1 for L = workday, route, customer **do**
    4.1.1 for I iterations **do**
      a. probabilistically select a destruction operator at level L and a reconstruction operator based on their current weights;
      b. apply the destruction and reconstruction operators to s to obtain s';
      c. if s' satisfies the acceptance criterion **then**
        - $s \leftarrow s'$;
        - **if** s' is better than $s*$ **then** $s* \leftarrow s'$;
    4.1.2 adjust weights;
5. return $s*$.

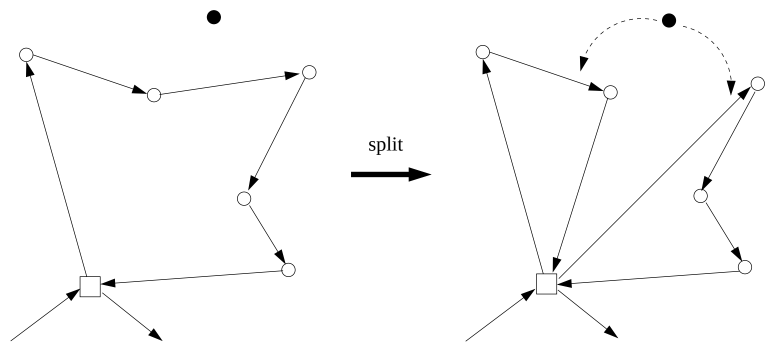

Figure 1: Split

is put in a list of (temporarily) unserved customers. The split procedure proved to be particularly useful when the infeasibility is due to the t_{max} constraint.

During this insertion procedure, the vehicle schedule based on the latest feasible departure times from the depot is considered to reduce as much as possible route durations. Let us assume that route r is the last route in a vehicle workday. This route is described by the sequence $(0^r_d = i^r_0, i^r_1, i^r_2, ..., i^r_{n_r}, i^r_{n_r+1} = 0^r_e)$, where 0^r_d and 0^r_e stand for the departure from and return to the depot, respectively, and n_r is the number of customers in the route.

To determine the latest schedule of that route, denoted by the latest feasible time to begin service $\bar{t}_{i^r_j}$ at each customer i^r_j, $j = 1, ..., n_r$, a backward sweep of the route is first applied from $i^r_{n_r+1} = 0^r_e$ to $i^r_0 = 0^r_d$ as follows:

$$\bar{t}_{i^r_{n_r+1}} \leftarrow b_{i^r_{n_r+1}},$$

Algorithm 2 Construction heuristic

1. $L \leftarrow \{1, 2, ..., n\}$;
2. **while** $(L \neq \emptyset)$ **do**
  2.1 randomly select customer $i \in L$;
  2.2 insert customer i at its best feasible insertion place in the solution;
  2.3 **if** there is no feasible insertion place in the solution **then**
    2.3.1 **for** each route k in the solution apply *split(i, k)*;
  2.4 $L \leftarrow L - \{i\}$.

$$\bar{t}_{i_j^r} \leftarrow min\{\bar{t}_{i_{j+1}^r} - t_{i_j^r i_{j+1}^r} - s_{i_j^r},\ b_{i_j^r}\},\ j = i_{n_r}^r, ..., i_0^r.$$

Once the latest departure time from the depot $\bar{t}_{i_0^r}$ is obtained, a forward sweep is applied from $i_0^r = 0_d^r$ to $i_{n_r+1}^r = 0_e^r$ to reset the $\bar{t}_{i_j^r}$ values and get the latest feasible schedule:

$$\bar{t}_{i_j^r} \leftarrow max\{\bar{t}_{i_{j-1}^r} + t_{i_{j-1}^r i_j^r} + s_{i_{j-1}^r},\ a_{i_j^r}\},\ j = i_1^r, ..., i_{n_r+1}^r.$$

The total route duration is $\bar{t}_{i_{n_r+1}^r} - \bar{t}_{i_0^r}$, which is also the minimum duration, because the waiting time is minimized by serving the route at the latest feasible time. The whole procedure is then applied to the second-to-last route $r - 1$ by setting

$$\bar{t}_{i_{n_{r-1}+1}^{r-1}} \leftarrow \bar{t}_{i_0^r} - \sigma_r.$$

This is repeated until the first route is done.

3.2 Destruction operators

Different destruction operators are defined at the customer, route and workday level. They are described in the following.

3.2.1 Customer level

These operators remove a number n_1 of individual customers from the current solution.

Random customer removal. This is a very simple approach where the customers are chosen at random.

Related customer removal. This operator is inspired from the one described by Shaw in [17]. The main difference is that the proximity measure between two customers is based on both spatial and temporal characteristics. This operator can be described as follows (see Algorithm 3).

Starting with a randomly chosen request, which starts the whole procedure, the removal of the next request is (probabilistically) biased toward those that are close

Algorithm 3 Related customer removal

1. randomly select a customer j and remove it from the solution;
2. $L \leftarrow \{j\}$;
3. **while** $\mid L \mid < n_1$ **do**
  3.1 $i \leftarrow$ randomly select a customer in L;
  3.2 **for** each customer j in the solution **do**
    3.2.1 $z_{ij} \leftarrow \alpha \cdot |t_i - t_j| + \beta \cdot d_{ij}$;
  3.3 sort the z_{ij}'s in non decreasing order and put them in list B;
  3.4 choose a random number x between 0 and 1;
  3.5 $pos \leftarrow \lceil \mid B \mid \cdot x^d \rceil$;
  3.6 select customer j associated with the z_{ij} value at position pos in B and remove it from the solution;
  3.7 $L \leftarrow L \cup \{j\}$;
4. return L

to one of the previously removed requests. The proximity between two customers is based on a metric that accounts for both the geographical distance and the absolute difference between the time of beginning of service at both customer locations, weighted by parameters α and β, respectively. Parameter d in step 3.5 controls the intensity of the bias. Namely, a high value for parameter d strongly favors the removal of requests that are close to previously removed requests (and conversely). Based on preliminary experiments, this parameter was set to 6.

3.2.2 Route level

These operators remove a number n_2 of individual routes from the current solution.

Random route removal. This is a very simple approach where the routes are chosen at random.

Related route removal. This is an adaptation of the corresponding customer-based operator which is applied at the route level. Accordingly, the algorithmic framework is very similar to the one found in Algorithm 3. First, a route is randomly chosen. Then, the removal of the next route is (probabilistically) biased toward those that are close to one of the previously removed routes.

Two different proximity measures between route r_j and some previously removed route r_i have been considered, where it is assumed that a route is defined by its set of customers.

1. *GC*: $z_{ij} \leftarrow d_{(g_{r_i} g_{r_j})}$, where g_{r_i} amd g_{r_j} are the gravity centers of routes r_i and r_j, respectively.
2. *MinD*: $z_{ij} \leftarrow \min_{k \in r_i, l \in r_j} d_{kl}$.

In the first case, the proximity is measured by the distance between the gravity centers of routes r_i and r_j, where the latter is defined as the average location over

Algorithm 4 Related route removal

1. randomly select a route r_j and remove it from the solution;
2. $L \leftarrow \{r_j\}$;
3. **while** $| L |< n_2$ **do**
  3.1 $r_i \leftarrow$ randomly select a route in L;
  3.2 **for** each route r_j in the solution **do**
    3.2.1 $z_{ij} \leftarrow$ compute the proximity measure between r_i and r_j;
  3.3 sort the z_{ij}'s in non decreasing order and put them in list B;
  3.4 choose a random number x between 0 and 1;
  3.5 $pos \leftarrow \lceil | B | \cdot x^d \rceil$;
  3.6 select route r_j associated with the z_{ij} value at position pos in B and remove it from the solution;
  3.7 $L \leftarrow L \cup \{r_j\}$;
4. return L.

all customer locations in the route. In the second case, the proximity is measured by the smallest distance between any pair of customers taken from routes r_i and r_j.

3.2.3 Workday level

A single operator is defined at the workday level. It simply removes n_3 randomly chosen workdays from the solution.

3.3 Insertion operators

After the application of a destruction operator, the customers that are not part of the solution, either because they have just been removed or because there was no feasible insertion place for them, are considered for reinsertion. Two different insertion heuristics have been defined for this purpose.

3.3.1 Least-cost heuristic

This is the insertion heuristic used for constructing an initial solution (see Algorithm 2), except that it is applied only to customers that are not part of the solution. Accordingly, the selection of the next customer is random and its insertion takes place at the feasible location with minimum detour.

3.3.2 Regret-based heuristic

A second insertion heuristic has been devised to alleviate the myopic behavior of the least-cost heuristic. This is done by defining a reinsertion order based on a regret measure. For a given customer, the 2-regret heuristic computes the minimum feasible detour in each workday. Then, it considers the difference between the detour in the second best and best workday. If this difference is large, the corresponding customer gets high priority for reinsertion because a large cost is incurred if its best workday becomes infeasible (due to the insertion of other customers). A generalized variant considers the minimum detour in each workday and sums up the differences, over all workdays, between the minimum detour in the workday and the overall minimum detour.

More formally, let us assume that the minimum detour when customer i is inserted in workday k is $detour_{ik}$ and that the overall minimum detour is obtained when customer i is inserted in workday k^*. Then, the generalized regret measure of customer i is:

$$gen\_regret_i = \sum_{k=1,\dots,m} (detour_{ik} - detour_{ik^*}) \, .$$

At each iteration, the customer with the maximum regret measure is selected for reinsertion at the feasible insertion place with minimum detour.

3.4 Acceptance criterion

The criterion for accepting or rejecting a new solution is the one used in simulated annealing [9]. That is, the new solution s' is accepted over the current solution s if s' is better than s. Otherwise, it is accepted with probability

$$e^{-\frac{f(s')-f(s)}{T}}$$

where T is the temperature parameter and f is the objective function. Starting from some initial value, the temperature is lowered from one iteration to the next by setting $T \leftarrow c \cdot T$. Clearly, the probability of accepting a non-improving solution diminishes with the value of T, as the algorithm unfolds. This behavior allows the algorithm to progressively settle in a (hopefully) good local optimum. In our experiments, the starting temperature was set to $1.05 \cdot f(s_0)$, where s_0 is the initial solution, and c to 0.99975, as suggested in [15].

3.5 Adaptive mechanism

The ALNS applies a removal and an insertion operator at each iteration on the current solution. The adaptive mechanism is aimed at choosing the removal and insertion operators in a way that accounts for their previous outcomes. A weight is associated with each operator for this purpose. Let us assume that we have h insertion operators, each with a weight w_j, $j = 1, ..., h$. The insertion 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 insertion operator, the weights are updated after a number of consecutive iterations (200 iterations in our implementation), called a segment. The weights at the start of a given segment sg are based on those used in the previous segment $sg-1$ and are computed as follows:

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

where γ has a value between 0 and 1 and π_i^{sg-1} is the score of operator i at the end of segment $sg-1$. This score, reset to zero at the beginning of each segment, is incremented when insertion operator i is used at a given iteration t to produce a new solution. More precisely, the new score at iteration $t+1$ becomes

$$\pi_i^{t+1} = \pi_i^t + \begin{cases} \sigma_1 & \text{if a new best solution has been produced,} \\ \sigma_2 & \text{if the solution produced is better than the current solution,} \\ \sigma_3 & \text{if the solution produced is accepted,} \\ & \text{but is worse than the current solution,} \end{cases}$$

where σ_1, σ_2 and σ_3 are parameters.

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 same approach is also used to update the weights of the removal operators at each level.

3.6 Termination criterion

The termination criterion is based on a fixed number of 24,000 iterations. This number has been chosen to allow convergence even on the largest test instances.

4 Computational Results

Solomon's 100-customer VRPTW instances [18], as well as the 200-, 400-, 600-, 800- and 1,000-customer instances of Gehring and Homberger [8], were used to test our algorithm. In these Euclidean instances, the travel time between two customer locations is the same as the Euclidean distance. There are six different classes of instances depending on the location of the customers (R: random; C: clustered; RC: mixed) and width of the scheduling horizon (1: short horizon; 2: long horizon). In this study, instances of type 1 have been discarded due to the short horizon that does not allow a significant number of routes to be sequenced to form a workday. Results are thus reported for R2 (11 instances), C2 (8 instances) and RC2 (8 instances). All tests were run on an AMD Opteron 3.1 GHz with 16 GB of RAM.

Solomon's instances, as well as Gehring and Homberger's VRPTW instances had to be modified to fit our problem. In particular, t_{max} was set to 100 to generate multiple routes for each vehicle. The customer coordinates of Gehring and Homberger's instances were also normalized to fit within a 100 X 100 Euclidean

% dstr.	size	100 iterations % unsv.	dist.	CPU (s)	200 iterations % unsv.	dist.	CPU (s)	400 iterations % unsv.	dist.	CPU (s)
05-35	100	28.1	1938.0	42.8	27.8	1923.5	42.4	28.1	1924.9	44.7
	400	28.0	10940.6	671.6	28.1	10938.0	678.5	28.3	10938.2	671.6
	800	30.2	22713.3	2822.9	30.6	22698.9	2898.5	30.3	22830.2	2867.9
35-65	100	31.4	1950.9	44.6	32.2	1949.2	44.8	31.5	1960.7	46.8
	400	29.9	11157.0	745.0	30.0	11185.9	740.3	29.8	11258.1	755.9
	800	30.9	22915.8	2966.8	30.9	22976.3	2990.6	31.0	22850.0	2938.1
65-95	100	34.0	1966.7	48.5	33.7	1963.7	45.7	35.6	1989.3	52.3
	400	30.3	11277.1	823.6	30.0	11199.6	807.9	30.1	11230.7	806.7
	800	30.5	22896.3	3228.2	30.4	22774.4	3285.2	30.4	22677.3	3224.2

Table 1: Impact of % of destruction and number of iterations at each level

square, as in Solomon's instances. Furthermore, the service or dwell time at each customer was set to 10 in all instances. The number of vehicles was set to 3 for the 100-customer instances, and this number was increased to 6, 12, 18, 24 and 30 for the 200-, 400-, 600-, 800- and 1,000-customer instances, respectively, to obtain instances with approximately the same degree of difficulty.

In the following, some parameter sensitivity results are presented. Then, the final results on the whole test set are reported. A comparison with known optimal solutions on small instances with no more than 40 customers are also reported at the end.

4.1 Parameter sensitivity

We have studied the impact of the number of consecutive iterations at each level and percentage of destruction (*% dstr.*) on the algorithmic performance. To this end, we have used a test set made of the RC2 instances with 100, 400 and 800 customers. The results are shown in Table 1. We have considered 100, 200 and 400 consecutive iterations at each level (i.e., 300, 600 and 1200 iterations for the whole workday-route-customer level sequence). Three different intervals for the percentage of destruction of the current solution were also tested, namely [5%, 35%], [35%, 65%] and [65%, 95%]. When a removal operator is chosen, a percentage value is uniformly randomly selected in the corresponding interval and applied at the appropriate level. For each possible combination of the two parameters and for each problem size, Table 1 shows the average percentage of unserved customers (*% unsv.*), total distance (*dist.*) and computation time in seconds (CPU).

Not surprisingly, the computation time increases with the percentage of destruction because it is more costly to reinsert a larger number of customers into the current solution. Furthermore, solution quality tends to degrade. Based on the results shown in Table 1, the best combination is a percentage of destruction in the interval [5%, 35%] and 200 consecutive iterations at each level (i.e., 600 iterations for the whole workday-route-customer sequence and 40 such sequences over 24,000 iterations). This setting has been used in the following sections.

4.2 Results

Based on the best parameter setting identified in section 4.1, different variants of our ALNS have been applied to the whole set of test instances. These results are reported in Table 2. As indicated in this table, five different variants have been considered: *Cb* only uses the customer-based removal operators, *Cb/Rb* uses both the customer- and route-based removal operators while *Cb/Rb/W* uses all removal operators. Two variants of *Cb/Rb/W* have also been tested: *Cb/Rb1/W*, where the related route removal operator using the *MinD* proximity measure is discarded (thus, only the random route removal and the related route removal using the *GC* measure are considered) and *Cb/Rb2/W*, where the related route removal using the GC measure is discarded (thus, only the random route removal and the related route removal using the *MinD* measure are considered). In each entry, we show the percentage of unserved customers (%), the total distance and the computation time in seconds (s), in this order. These results are averaged over all sizes for each problem class in Table 3.

The first observation is that the exploitation of a multi-level scheme is very beneficial when compared to the classical customer-based approach. By introducing a route level, the percentage of unserved customers is reduced by 7.81%, 8.22% and 6.92% on problem classes *RC*, *R* and *C*, respectively. An additional improvement, although less important, is also obtained by introducing the workday level.

The differences observed between *Cb/Rb1/W* and *Cb/Rb2/W* also indicate that the related route removal operator using the *MinD* proximity measure is superior to the one using the gravity center-based measure. Also, by comparing *Cb/Rb2/W* and *Cb/Rb/W*, a single related route removal operator based on the *MinD* measure provides better results than the use of the two operators concurrently.

4.3 Comparison with optimal solutions

The best ALNS variant *Cb/Rb2/W* has been applied to small instances for which the optimum is known and reported in [2]. These instances have been created by considering only subsets of 25 and 40 customers in Solomon's original instances [18]. The reported optima have been obtained with $t_{max} = 75$ and a fleet of 2 vehicles. Table 4 reports average results obtained over all instances of given class and size. On the 25-customer instances, ALNS is quasi-optimal. All customers are served and the gap in total distance does not exceed 1%. On the 40-customer instances of type *RC* and *C*, ALNS is also close to the optimum. On the three instances of type R, a difference of 2.5% is observed with regard to the percentage of unserved customers and a gap of 16% with regard to the total distance. But, these average gaps are largely due to only one particular instance, for which 3 customers are left unserved by ALNS (no unserved customer in the optimal solution) and a gap close to 30% is observed in total distance.

size	class	Cb	Cb/Rb	Cb/Rb/W	Cb/Rb1/W	Cb/Rb2/W
		27.0%	25.5%	24.8%	25.6%	25.7%
	RC	1909.6	1892.9	1894.2	1900.4	1899.2
		27.8s	29.0s	27.9s	27.5s	28.1s
		13.5%	11.5%	10.9%	12.2%	10.9%
100	R	1866.1	1828.8	1828.1	1838.8	1828.6
		29.9s	35.5s	32.7s	34.3s	33.5s
		0.0%	0.0%	0.0%	0.0%	0.0%
	C	2393.0	2239.6	2269.3	2221.3	2232.9
		41.2s	36.0s	42.8s	46.7s	43.2s
		20.3%	14.6%	12.3%	12.8%	12.4%
	RC	8690.8	9202.2	9205.9	9262.9	9218.3
		130.1s	148.5s	135.3s	143.3s	140.3s
		13.5%	7.4%	6.4%	6.4%	6.2%
200	R	10360.2	10577.7	11126.6	11239.8	11103.7
		125.6s	131.9s	129.4s	157.3s	126.9s
		3.1%	0.0%	0.0%	0.0%	0.0%
	C	9994.1	9750.4	9818.9	9806.7	9730.3
		103.0s	110.0s	138.1s	125.0s	126.2s
		33.8%	25.6%	23.2%	23.2%	22.4%
	RC	9633.5	10310.4	10279.3	10311.8	10128.0
		500.0s	522.8s	475.1s	488.9s	460.4s
		18.4%	8.5%	6.3%	6.5%	6.2%
400	R	11784.2	12102.1	12766.0	12903.3	12657.7
		511.9s	526.5s	438.4s	496.5s	427.7s
		7.6%	0.2%	0.1%	0.1%	0.0%
	C	11425.1	11949.6	11256.8	11921.0	10937.2
		471.4s	466.0s	349.3s	385.7s	340.0s
		32.7%	21.8%	21.2%	21.1%	20.4%
	RC	14170.0	14384.5	15860.2	15831.1	15577.9
		1170.8s	1211.5s	1127.4s	1099.3s	1114.2s
		20.4%	10.4%	6.4%	6.5%	6.4%
600	R	17023.7	18903.3	19270.8	19400.4	19089.2
		1281.5s	1291.4s	1025.0s	1139.1s	1009.1s
		23.3%	13.0%	10.1%	10.5%	9.9%
	C	14187.8	14225.4	14667.9	14803.7	14626.0
		1184.2s	1360.4s	1046.9s	1129.8s	1028.5s
		36.5%	26.1%	25.0%	25.2%	24.3%
	RC	18353.3	18878.8	20772.6	20921.4	20858.5
		1938.4s	1993.2s	1818.2s	1955.7s	1842.8s
		22.2%	11.3%	8.0%	8.3%	7.9%
800	R	22270.1	23890.9	26192.8	26402.5	26136.9
		2088.4s	2125.2s	1691.2s	1834.0s	1678.3s
		37.9%	26.9%	25.0%	24.8%	24.9%
	C	14333.8	14271.9	14427.7	14454.6	14441.1
		1861.7s	2071.3s	1857.1s	1810.6s	1747.4s
		37.5%	30.3%	27.3%	27.5%	26.3%
	RC	22063.0	25157.9	25635.0	25584.2	25368.3
		3008.5s	3331.5s	3050.5s	2899.5s	2855.2s
		24.7%	14.2%	10.6%	10.8%	10.3%
1000	R	26862.1	28876.2	30700.5	30845.8	30732.2
		3377.0s	3358.1s	2730.8s	2943.5s	2718.8s
		49.6%	39.8%	36.9%	36.9%	36.5%
	C	14994.9	14670.6	14700.3	14637.2	14587.6
		2508.9s	2804.8s	2819.6s	2807.1s	2602.4s

Table 2: Average results by problem classes and sizes

class	Cb	Cb/Rb	Cb/Rb/W	Cb/Rb1/W	Cb/Rb2/W
RC	31.8% 12470.0 1129.3s	24.0% 13304.5 1206.1s	22.3% 13941.2 1105.7s	22.6% 13968.6 1102.4s	21.9% 13841.7 1073.5s
R	18.8% 15027.7 1235.7s	10.6% 16029.8 1244.7s	8.1% 16980.8 1007.9s	8.5% 17115.4 1100.8s	8.0% 16924.7 999.0s
C	20.2% 11221.5 1028.4s	13.3% 11184.6 1141.4s	12.0% 11190.2 1042.3s	12.0% 11307.4 1050.8s	11.9% 11092.5 981.3s

Table 3: Average results over all sizes for each problem class

			Cb/Rb2/W		Optimal	
size	class	# instances	% unsv.	distance	% unsv.	distance
25	RC	5	0.0	844.1	0.0	844.0
	R	11	0.0	624.1	0.0	620.1
	C	7	0.0	635.6	0.0	629.1
40	RC	3	13.9	1386.4	13.3	1377.5
	R	3	3.3	1289.0	0.8	1071.6
	C	3	0.0	1105.0	0.0	1093.0

Table 4: Comparison with optimal solutions

5 Conclusion

This paper has described an adaptation of the ALNS framework based on the hierarchical structure of the vehicle routing problem with multiple trips. Empirical results demonstrate that it is very beneficial to apply operators at the customer, route and workday levels, as opposed to the classical approach where only customer-based operators are used.

References

[1] Alonso F., Alvarez M.J., Beasley J.E., 2008. "A Tabu Search Algorithm for the Periodic Vehicle Routing Problem with Multiple Vehicle Trips and Accessibility Restrictions", *Journal of the Operational Research Society 59*, 963-976.

[2] Azi N., Gendreau M., Potvin J.-Y., 2010. "An Exact Algorithm for a Vehicle Routing Problem with Time Windows and Multiple Use of Vehicles", *European Journal of Operational Research 202*, 756-763.

[3] Azi N., Gendreau M., Potvin J.-Y., 2007. "An Exact Algorithm for a Single Vehicle Routing Problem with Time Windows and Multiple Routes", *European Journal of Operational Research 178*, 755–766.

[4] Battarra M., Monaci M., Vigo D., 2008. "An adaptive Guidance Approach for the Heuristic Solution of a Minimum Multiple Trip Vehicle Routing Problem", *Computers & Operation Research*, 316–329.

[5] Brandão J.C.S., Mercer A., 1998. "The Multi-Trip Vehicle Routing Problem", *Journal of the Operational Research Society 49*, 799–805.

[6] Brandão J.C.S., Mercer A., 1997. "A Tabu Search Algorithm for the Multi-Trip Vehicle Routing and Scheduling Problem", *European Journal of Operational Research 100*, 180–191.

[7] Fleischmann B., 1990. "The Vehicle Routing Problem with Multiple Use of Vehicles", Working Paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Germany.

[8] Gehring H., Homberger J., 1999. "A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows". In: K. Miettinen, M.M. Mäkelä, J. Toivanen (Eds.), Proceedings of EUROGEN99 - Short Course on Evolutionary Algorithms in Engineering and Computer Science, Reports of the Department of Mathematical Information Technology, No. A 2/1999, University of Jyväskylä, 57–64.

[9] Kirkpatrick S., Gelatt Jr. C.D., Vecchi M.P., 1990. "Optimization by Simulated Annealing", *Science 220*, 671–680.

[10] Olivera A., Viera O., 2007. "Adaptive Memory Programming for the Vehicle Routing Problem with Multiple Trips", *Computers & Operations Research 34*, 28–47.

[11] Petch R.J., Salhi S., 2003. "A Multi-Phase Constructive Heuristic for the Vehicle Routing Problem with Multiple Trips", *Discrete Applied Mathematics 133*, 69–92.

[12] Petch R.J., Salhi S., 2007. "A GA Based Heuristic for the Vehicle Routing Problem with Multiple Trips", *Journal of Mathematical Modelling and Algorithms 6*, 591–613.

[13] Pisinger D., Ropke S., 2007. "A General Heuristic for Vehicle Routing Problems", *Computers & Operations Research 34*, 2403–2435.

[14] Rochat Y., Taillard E., 1995. "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", *Journal of Heuristics 1*, 147–167.

[15] Ropke S., Pisinger D., 2006. " An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows", *Transportation Science 40*, 455–472.

[16] Schrimpf J., Schneider H., Dueck G., 2000. "Record breaking optimization results using the ruin and recreate principle", *Journal of Computational Physics 159*, 139–171.

[17] Shaw P., 1998. "Using constraint programming and local search methods to solve vehicle routing problems", In: G. Goos, J. Hartmanis and J. van Leeuwen (Eds.), Principles and Practice of Constraint Programming - CP98, *Lecture Notes in Computer Science 1520*, 417–431.

[18] Solomon M.M., 1987. "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints", *Operations Research 35*, 254–265.

[19] Taillard E. D., Laporte G., Gendreau M., 1996. "Vehicle Routeing with Multiple Use of Vehicles", *Journal of the Operational Research Society 47*, 1065–1070.

[20] Toth P., Vigo D., 2002. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA.