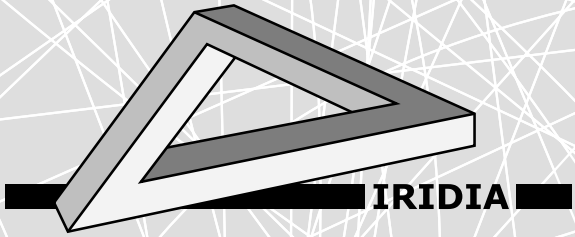


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Application of Two Nearest Neighbor Approaches to a Rich Vehicle Routing Problem

Paola Pellegrini

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2005-015

October 2005

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2005-015

Application of Two Nearest Neighbor Approaches to a Rich Vehicle Routing Problem

Paola PELLEGRINI paolap@pellegrini.it
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

December 1, 2005

Abstract

In recent years, rich vehicle routing problems have been considered with particular attention by many researchers. In this paper a specific case is analyzed, which includes multiple time windows constraints and heterogeneous fleet of vehicles. A measure of difficulty of the instances is proposed and some elements that have an impact on it are analyzed. Furthermore, two heuristic algorithms are proposed, based on the well known *nearest neighbo*r heuristic procedure. Computational experiments are performed on instances with different size and different level of difficulty. The results obtained are quite encouraging.

Keywords: Vehicle routing problem, multiple time windows, heterogeneous fleet, difficulty of instances, randomized nearest neighbor.

1 Introduction

The vehicle routing problem (*VRP*) is very well known in the literature. Many techniques have been proposed for solving it, either optimally or approximately. The increasing efficiency of these methods and the availability of a larger computing power allow more and more researchers to focus their attention on what are called rich vehicle routing problems. Under this denomination many variants of the classical *VRP* are grouped. These variants have in common the fact that they consider some aspects that are essential to the routing of vehicles in real problems. Some examples of these features are: the use of a heterogeneous fleet [11, 14], the limitation due to time windows [5, 6], periodic vehicle routing [1], backhauling [7], pickup and delivery [8], the fleet size and mix problem [16], the use of vehicles with multiple compartments, and the consideration of multiple objectives in routing schedules [13].

In this paper we address one of these variants, considering mainly two kinds of constraints that are often found in reality: multiple time windows constraints and heterogeneous fleet of vehicles. Although it is an important issue for practical applications, to the best of our knowledge no reference on this topic can be found in literature. Taking into account these two kinds of constraints simultaneously, each considered very challenging when tackled alone, implies a significant increase in the complexity of the instances. For this reason, as first approach to this rich *VRP*, we propose an algorithm that brings together a classical heuristic used for the vehicle routing problem with single time windows, the *nearest neighbor* proposed by Solomon in 1987 [12], and some randomness that nowadays is proving to be very effective when dealing with complex combinatorial problems. We will refer to this new algorithm as the *randomized nearest neighbor*.

When studying the performance of an algorithm with respect to a problem, it is important to identify the class (or classes) of instances to which the results are referred. In a similar way, it is in general very interesting to study the changes in the behavior when dealing with different classes of instances. As a consequence, an important issue in this framework is the classification of different classes of instances. Analyzing the feasible region for particular instance, or a class of instances, is clearly very difficult even when considering only partial aspects of it. This implies that it is not easy to identify *a priori* which among two instances (or, again, two classes of instances) is the most difficult to solve. In this contest we consider a measure of difficulty represented by the probability of obtaining a feasible solution for an instance, when this solution is randomly generated. Of course this measure cannot represent all the characteristics of the search space, especially if we take into account that the same configuration may result more favorable to some approaches than to others, since the procedures for exploring the space can vary significantly. Nonetheless, we consider the measure defined above as a first step in the direction of a deeper understanding of some relevant elements. To this aim, we propose an analysis of how some elements can impact this difficulty.

Finally, a computational experience is presented, comparing the performance of the *randomized nearest neighbor* with the one of the deterministic version on four classes of instances with different degrees of difficulty.

The rest of the paper is organized as follows. In Section 2 the vehicle routing problem with multiple time windows and heterogeneous fleet is defined. In Section 3 the *deterministic nearest neighbor* and the *randomized nearest neighbor* are presented. In Section 4 we describe the logic behind the generation of the instances we use for the experiments and in Section 5 the characteristics of the classes of instances considered are investigated. Finally, in Section 6 the computational results are reported.

2 Vehicle routing problem with multiple time windows and heterogeneous fleet

The vehicle routing problem in its general formulation concerns the distribution of goods to a defined set of customers, using a fleet of vehicles with fixed capacity located in a depot. The optimal solution consists in the set of tours that allow to visit all the customers at the minimum total cost satisfying a set of constraints: the quantity demanded by each customer must be delivered; every customer must be visited exactly once; the capacity of the vehicles cannot be exceeded; all the vehicles must leave and go back to the depot. Many references can be found about this classical version and about its variants; for a deep review and analysis we refer the reader to [15].

It is worth making an observation concerning the logic according to which the solutions are considered. As just said, the objective is to find the set of tours implying the minimum total cost. In the literature, the concept of total cost has been defined in different ways, and two main elements can be detected to have an impact on it. The first corresponds to the number of vehicles required to serve all the customers, and the second to the traveling time (or distance) implied. In the current study we consider these two objectives according to a hierarchical order, the first being the number of vehicles used and the second the total time of the tours.

Moreover, the particular problem we are tackling considers some additional constraints to this general framework. In particular, we allow the customers to define a set of time windows in which they are available to receive the visit of a vehicle. These time windows are intervals out of which the service cannot be offered: if the vehicle reaches the customer before the opening time it has to wait until that moment, and if it reaches the customer after the moment in which the time window closes it cannot deliver the required good[1]. A peculiarity of the problem studied here is the fact that multiple time windows are considered, while the literature is focused on the single time window case. The reason why we are introducing this additional source of complexity is that

[1] We are considering here strong time windows, which are counterposed to soft ones. This second case is realized when the service can be offered out of the time window, incurring in a penalty cost.

it appears very often in reality, for example in most of the cases in which the tours concerning a whole week have to be planned at the same moment.

Another constraint considered here is the one imposed by the availability of different vehicles with different characteristics. The first diversity that comes to one's mind is related to the capacity. This is of course the case in some situations, but in many others we can have vehicles with the same capacity but able, for example, to transport different types of goods. In both cases, the specific demand of a customer can impose the utilization of a specific type of vehicle. This is exactly the kind of constraint we are considering, supposing that for serving a customer it may be necessary to use a precise type. As in many real situations, the cost of the different vehicles is considered equivalent (in many cases the cost of the driver is the one that impacts most the accounting of the costs, and it is constant for the different vehicles), so the consideration of the heterogeneous fleet is not related to the computation of the total cost of a solution, and then to the objective function, but to the additional constraints necessary for not including in the same tour customers requiring different vehicles. This hypothesis is often met in reality.

Finally, since the main objective is considering problems of practical interest, a constraint has been included concerning the time needed for completing a single tour: the maximum time for each has been fixed to 8 hours, corresponding to a standard working day.

3 Algorithms

As anticipated in the introduction, we analyze here two strictly connected algorithms. The first one is an adaptation of the *nearest neighbor* heuristic proposed by Solomon in 1987 [12] for tackling the *VRP* with single time windows, which consists of the successive insertion of the nearest customer. Of course the concept of closeness when dealing with time windows is not easy to define, since the waiting time implied by the choice of the spatially closest feasible customer may have a great impact on the quality of the solution. In order to take this element into account, Solomon proposed the following measure of *distance* between the generic customers i and j:

$$distance(i,j) = \alpha \cdot t(i,j) + \beta \cdot waiting(j) + \gamma \cdot urgency(j). \tag{1}$$

The relevant elements in this formula are $waiting(j)$ representing the waiting time that would be implied by moving to customer j with time window $[startTW(j), endTW(j)]$ after customer i (which imply a traveling time of $t(i,j)$), with the service of $serviceTime(i)$ time units starting at $startService(i)$ (2), and $urgency(j)$ representing the urgency of offering the service implied by the remaining part of the time window of customer j (3). The *distance* between two customers is the weighted average of these two quantities and the effective traveling distance considered in terms of time ($t(i,j)$).

$$waiting(j) = \max\{startTW(j) - startService(i) - serviceTime(i) - t(i,j), 0\}; \tag{2}$$

$$urgency(j) = endTW(j) - t(i,j) - startService(i) - serviceTime(i). \tag{3}$$

The tours, then, are constructed choosing one after the other the customer that minimizes (1) until feasible moves are available, and returning to the depot when this is not the case. This procedure is iterated until all the customers have been served. As it appears clear, this heuristic belongs to the class of the greedy algorithms.

In order to apply this procedure to the problem we are considering here, two main changes are necessary: first of all the definition of feasible customer has to be adapted, taking into account the constraints concerning the heterogeneous fleet and the maximum time available for completing the subtours; then a way for dealing with the multiple time windows has to be defined.

The constraint related to the maximum time available for completing each subtour can be easily inserted, forcing the vehicle to go back to the depot when the available time has elapsed.

For taking into account the differences among vehicles, the following procedure has been applied. First of all, one type of vehicle $\hat{k}$ is fixed as default. Then the insertion of the customers begins considering the capacity constraint imposed by type $\hat{k}$, and without taking into account

the type of vehicle possibly required by the customers. When the first customer requiring a specific type of vehicle is inserted, this type is considered for the subtour. Of course the following customers will be feasible if either they do not require a specific class of vehicles or they require the class already established. When considering customers for the insertion while the vehicle is still the default one, if the cumulated demand is bigger than the capacity of some type of vehicle, all the customers requiring such types are set unfeasible.

As far as the multiple time windows are concerned, only one of them is considered, and it is chosen dynamically at each step: the selected time window is the first one that closes after the moment in which the customer can be reached, if such time window exists.

After the construction, a *swap* local search is applied to the solution obtained. It consists in the inversion of the position in the sequence of the couples of consecutive customers until an improvement is found. In this case the procedure is restarted one step before the improvement. This local search is well known in the literature and it was first proposed by Lin in 1962 [9]. It offers the advantage of not being very expensive in terms of computational time although it offers satisfactory results, which in our case is crucial since the addition of many constraints could make most of the other typical procedures very time consuming, preventing any solution method from being useful in real applications.

Combining the approach proposed by Solomon, the explained adaptation for dealing with the new constraints and the local search, we get the algorithm that will be referred to as *deterministic nearest neighbor* (*DNN*).

One of the aims of this paper is the presentation of another approximate solution method, which we will call *randomized nearest neighbor* (*RNN*). The main idea is joining the approach of the algorithm just described with the stochasticity, that is having a great success in the literature related to complex combinatorial problems. This is possible by repeating the procedure presented for the *DNN* with a simple variation: when the vehicle is leaving the depot for starting a new tour, the choice of the customer is random, according to a uniform distribution in which all the customers not yet inserted have equal probability of being chosen; after this first step, the selection goes on following the *distance* relation, and when the solution is complete, the *swap* local search is applied. This procedure is repeated until the computational time available has elapsed. A great advantage of this algorithm is its simplicity, that allows to tackle, in a reasonable amount of time, hard problems as the one considered here.

4 Instances used for the computational analysis

As far as we know, standard benchmark instances for the vehicle routing problem with multiple time windows and heterogeneous fleet are not publicly available. In order to test the behavior of the algorithms, we implemented an instances generator specific for this problem.

The main peculiarity of this procedure is the attempt to reflect as much as possible what can be a real situation.

The created instances consist in nodes grouped in one cluster representing an Italian city randomly selected in the available database. Following [10], a city is represented as a set of concentric circles. The number of circles depends on the population of the city considered (the higher the population, the higher the number of circles). We will refer to the areas obtained in this way as to *zones*. The nodes are uniformly distributed in each *zone* and the depot is located in the most external circle. For a detailed description of this procedure we refer the reader to [10], in which an instances generator for the vehicle routing problem with stochastic demand using the same procedure for the localization of the nodes is described.

For the current experiments, we consider instances with 50, 100, 150 and 200 customers, using the parameters relative to the computation of the distances between the nodes, as for example the speed of the vehicles, presented in [10]. In particular, the distances are expressed in travel time, associating to each *zone* a coefficient that represents the speed that can be used to go from one point to another, and for which the Euclidean distance must be multiplied. To reproduce the situation of real cities, where this speed decreases while one approaches the center, the coefficient

Table 1. Procedure for determining the time windows of customer J.

```
/* S(J_v) and E(J_v) represent the starting and the ending minute, respectively, of time
window v of customer J, while du(J_v) represents its duration. n(J) is the number of
time windows assigned to customer J */
Set v = 1
Choose n(J) such that minN ≤ n(J) ≤ maxN;
While v < n(J)
        Choose the day in which the time window has to be located (d_{J_v});
        Choose du(J_v) such that minDu ≤ du(J_v) ≤ maxDu;
        Choose S(J_v) such that minStart_{d_v} ≤ S(J_v) < maxEnd_{d_{J_v}};
        If (S(J_v) + du(J_v) > maxEnd_{d_{J_v}}) {
                E(J_v) = maxEnd_{d_{J_v}}
                If (E(J_v) − S(J_v) < minDu)
                        continue;
        }
        Else {
                E(J_v) = S(J_v) + du(J_v)
        }
        If (∃w < v : TW(J_w) and TW(J_v) either overlap or are distant less than di)
                continue;
        v++
```

becomes smaller while moving from a *zone* to a more internal one. The speed considered in order to compute these coefficients are 15 km/h for the most internal circle of the city, 40 km/h for the most external one, with proportional values associated to those being in between.

As far as the definition of the time windows is concerned, the procedure reported in Table 1 has been applied. The aim is to fix the number of time windows per customer (n) and their duration (du) randomly according to a uniform distribution with parameters previously set ($minN$ and $maxN$, $minDu$ and $maxDu$ respectively). Moreover, some choices have been kept stable: the time horizon has been set to 7200 hours, representing five working days, and the time windows has been fixed between 7:00 am ($minStart$) and 7:00 pm ($maxEnd$) of each day. Finally, a minimum distance between the time windows belonging to the same customer (di) has been imposed.

Finally the demand of each customer (dem) has been randomly selected between *min-dem* and *max-dem*, while the following elements are fixed: two types of vehicles have been considered with a capacity equal to 200 and 300, respectively. The number of customers requiring a specific type of vehicle varies between 15 and 18 (the specific customers have been randomly chosen according to a uniform distribution).

Modifying the parameters $minN$, $maxN$, $minDu$[2], $maxDu$, di, *min-dem* and *max-dem* four classes of instances have been generated. In particular:

1. $dem \in \{[10, 15], [10, 20], [10, 25]\}$, $n \in [2, 5]$, $du \in [240, 420]$, $di = 60$;
2. $dem \in \{[5, 45], [5, 50], [5, 55]\}$, $n \in [2, 5]$, $du \in [240, 420]$, $di = 60$;
3. $dem \in \{[10, 15], [10, 20], [10, 25]\}$, $n \in [4, 10]$, $du \in [120, 180]$, $di = 80$;
4. $dem \in \{[5, 45], [5, 50], [5, 55]\}$, $n \in [4, 10]$, $du \in [120, 180]$, $di = 80$.

The computational time used as a stopping criterion for tackling each instance has been fixed to 10 seconds for all the instances.

5 Difficulty of the instances

The parameters for the generation of the four classes of instances have been fixed taking into account that some characteristics have a greater impact than others on the difficulty of an instance.

[2]The duration and the minimum distance of the time windows are expressed in minutes.

In particular, the first consideration is related to the demand. Let's consider a very elementary example in which we want to count the feasible solutions using two vehicles of capacity equal to 100 for an instance with four nodes A, B, C and D with demand $d_A = 80$, $d_B = 90$, $d_C = 15$, $d_D = 5$. In this case, there are three possible solutions using two vehicles[3]: serving A and B in one tour and C and D in the second; serving A and C in one tour and B and D in the second; serving A and D in one tour and B and C in the second. It is evident that only the second option is feasible, being the load assigned to the vehicles in the first case equal to 170 and 20, and in the third case equal to 85 and 105, respectively, i.e. the capacity constraint for one of the subtours is violated. On the other hand, if the demands of the four nodes are respectively 40, 45, 50 and 55, despite the total quantity is bigger than before, the feasible options are two: the only inadmissible is serving A and B in one tour and C and D in the second, since the latter would require a capacity of the vehicle at least equal to 105. After these considerations, the first distinction between the settings of the parameters used for generating the instances has been related to the variance associated to the distribution from which the customers' demands are drawn: in classes 1 and 3 the variance is quite low, while it is much higher in classes 2 and 4.

The second observation is related to the time windows. Intuitively, the more the time windows of different customers are overlapping, the easier it is to find a solution, since the nodes at each step that are feasible according to this type of constraints are more numerous.

In this sense, let us forget for a moment all the constraints not related to the time windows, and let us analyze the probability $p(T)$ of having an overlap of at least T minutes between a selected time window of customer A (A_i) and the time windows of customer B ($TW(B)$). The reason why we consider this particular case of overlap is that we want to investigate the probability of being allowed to visit a generic customer (in our case B) at any point in the construction of the solution, after completing the service to any other customer (A). In this sense we are not interested in the time windows of customer A, a part from the one we are in, but what we want to investigate is the probability of being allowed to visit customer B immediately after A. Of course moving to B even if there are no overlapping time windows is possible but, since this study considers also constraints related to the duration of the subtours, in general it is not advantageous. In any case, it is possible to increase the duration of the time windows opportunely and investigating the overlap between these new intervals which are considered in some sense acceptable. Even if the reasoning presented is clearly not exhaustive, it appears nonetheless interesting and helpful for a deeper understanding of different situations. Another limitation of the following approach is that we consider all time windows of the same duration (du), which allows to consider as random variable only the distribution of the starting points. Again, even if limiting the investigation to these case, by considering different combinations of durations and number of time windows it is possible to have an idea of the general trends.

First of all it must be observed that given the configurations of the time windows used for the instances considered, at most two time windows of customer B (B_j and B_k) can overlap A_i, due to the duration and the minimum distance imposed. Therefore, there are two possible cases for having an overlap of at least $T = T_j + T_k$ minutes, with $T \leq du$: either B_j overlaps A_i of T_j and B_k of T_k, or just one of them, suppose B_j, overlaps A_i of $T = T_j + T_k$ minutes. For the first possibility to be true, the following inequalities must hold, omitting for the moment the choice of the day in which the time windows are inserted:

$$\left\{\begin{array}{lcl} S_{A_i} & \geq & S_{B_j} \\ S_{A_i} & \leq & S_{B_j} + du - T_j \\ S_{A_i} & \geq & S_{B_k} + T_k - du \\ S_{B_j} & \geq & MinStart \\ S_{B_j} & \leq & MaxEnd - du - x \end{array}\right. \Rightarrow \left\{\begin{array}{lcl} S_{A_i} & \geq & S_{B_j} \\ S_{A_i} & \leq & S_{B_j} + du - T_j \\ S_{A_i} & \geq & S_{B_j} + x - du + T_k \\ S_{B_j} & \geq & MinStart \\ S_{B_j} & \leq & MaxEnd - du - x \end{array}\right.$$

with x equal to the distance between the starting point of the two time windows of customer B, $du + di \leq x \leq 2du - T_j - T_k$. Moreover the three time windows must be chosen all in the same day.

[3]We are not interested in the order followed, since we are considering just the capacity constraint.

Normalizing all the quantities with respect to the interval of one day in which the time windows can be located ($MaxEnd - MinStart$) and setting $\alpha = \frac{di}{MaxEnd-MinStart}$, $\beta = \frac{T_j+T_k}{MaxEnd-MinStart}$ and $\omega = \frac{du}{MaxEnd-MinStart}$, it is easy to see how the listed conditions can be reduced to the location of S_{B_j} in the interval $[0, 1-x-\omega]$, and S_{A_i} in an interval of length $2\omega-x-\beta$, with $\omega+\alpha \leq x \leq 2\omega-\beta$. As a consequence, the event considered is characterized by the following probability:

$$p_1(T) = \Big(\frac{1}{nDays}\Big)^2 \int_{\omega+\alpha}^{2\omega-\beta} (1-x-\omega)(2\omega-x-\beta)dx. \tag{4}$$

If we consider the second case, in which only one time window of customer B overlaps A_i of at least T minutes, we must distinguish two possibilities: either two consecutive time windows B_j and B_k are fixed in the same day and are separated by an interval x such that $2du - T \leq x \leq MaxEnd - MinStart - du$, or they are placed in different days. Using the logic and the symbols introduced for the case of two time windows, the first possibility is represented in (5) and the second in (6):

$$p_2(T) = \Big(\frac{1}{nDays}\Big)^2 \int_{2\omega-\beta}^{1-\omega} (1-x-\omega)(2\omega-2\beta)dx; \tag{5}$$

$$p_3(T) = \frac{nDays-1}{nDays}\frac{1}{nDays}(1-\omega)(2\omega-2\beta). \tag{6}$$

In (5) the situation for S_{B_j} is similar to the one observed in (4), and so it is constrained to the interval $[0, 1-x-\omega]$, while in (6) it is no more limited by the presence of another time window B_k and so the reference interval becomes $[0, 1-\omega]$. On the other hand, S_{A_i} is in both cases free to vary between $S_{B_j} - \omega + \beta$ and $S_{B_j} + \omega - \beta$.

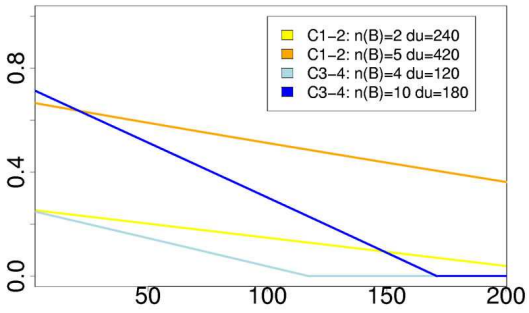
Being these three events disjoint, the probability of having an intersection of T minutes between two time windows of customer B and time window A_i of customer A is equal to the sum of $p_1(T)$, $p_2(T)$ and $p_3(T)$, multiplied by the number of possible couple of consecutive time windows B_j, B_k which is equal to $n(B) - 1$. In this way we consider all the possibilities except the overlap between the last time window of customer B and A_i, which is realized with probability $\frac{1}{nDays}(1-\omega)(\omega-\beta)$. The total probability, then, is:

$$p(T) = (n(B)-1)(p_1(T) + p_2(T) + p_3(T)) + \frac{1}{nDays}(1-\omega)(\omega-\beta). \tag{7}$$

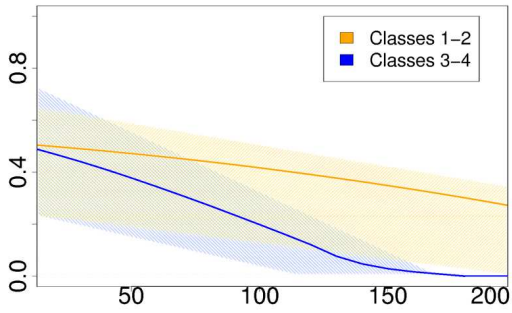
In order to analyze the differences between the classes of instances used for the computational experience, we computed this probability in the worst and in the best case scenario - minimum number of time windows and minimum duration, and maximum number of time windows and maximum duration, respectively - for the two couples of classes 1-2 and 3-4, for $0 \leq T \leq 200$. The results are reported in Figure 1(a).

Figure 1(b), on the other hand, represents the same probability empirically computed for the instances generated: for each time window of each customer we counted the number of customers having time windows overlapping of at least T minutes, and we averaged these values over the time windows, the customers and finally the instances. As it can be seen the average probabilities found following this procedure fits well the trend suggested by Figure 1(a). Both these graphics tell that for very low overlaps classes 3-4 are characterized by a higher probability than classes 1-2, while the opposite is true for greater values of T. In other words, even if the instances belonging to the second pair of classes are characterized by more likely overlaps, the dimensions of these overlaps are generally very small. When requiring a greater amount of minutes of correspondence among time windows, the probability decreases rapidly. In our eyes this characteristic is more relevant than the presence of more frequent negligible overlaps, since it is more likely to allow different combinations in the sequence of customers representing a tour. In this sense, then, the first two classes of instances appear less difficult in terms of time constraints.

As somehow showed by the previous reasoning, considering the impact of the difficulty of instances of one type of constraint is not an easy task in itself. Clearly, analyzing the characteristics



(a) Probability of overlap of T minutes (x axis) between one time window of customer A and all those of customer B in correspondence to the worst and the best case scenarios for the different classes, calculated according to formula (7).



(b) Average probability of overlap of T minutes (x axis) between one time window of customer A and all those of customer B calculated for the instances generated.

of instances in which these types of constraints are combined is even harder. In order to have an idea of what happens when we consider together the peculiarities previously analyzed, we propose an empirical study on the different classes of instances. In particular, we implemented a random solutions generator, which at each step inserts in the tour a node randomly chosen among the feasible ones. If there are no feasible insertions, a depot is selected until all the possible ones have been used. When there are neither more feasible customers nor more depots, or when all the customers have been inserted, the (possibly partial) solution built is recorded. If all the customers have been inserted the solution is feasible, otherwise it is not.

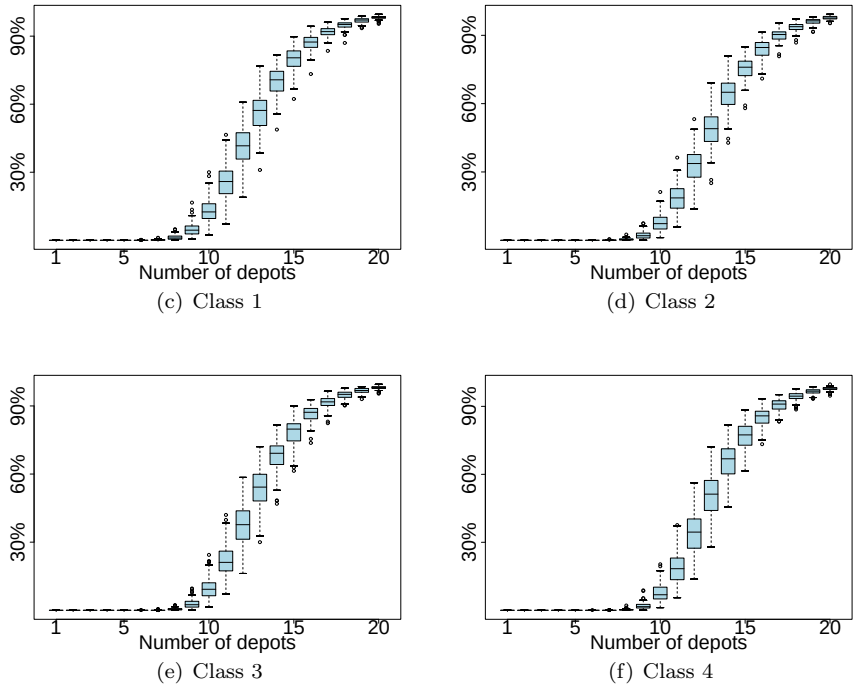
The percentage of feasibility is then computed dividing the number of feasible solutions found by the total number of those generated. For our experiments we used 1000 trials per instance, and 200 instances with 50 customers[4] for each class. All the possibilities between 1 and 20 depots were considered.

In Figure 1 the boxplots of the results of these experiments are reported. It is quite easy to see how the difficulty increases following the order 1-3-2-4. Nonetheless, for a more precise comparison we can consider Table 5 in the Appendix, in which for each number of vehicles we report the value of the lower end of the whisker, the first quartile (25th percentile), the second quartile (median=50th percentile), the third quartile (75th percentile), and the upper end of the whisker relative to each type. These results confirm the expectation previously exposed. Moreover, it is possible to observe that the increase in the difficulty follows the schema just defined, with an almost equivalent incidence of the change of the time windows and of the demands configurations, with a slight advantage in terms of easiness for class 3. When the two factors are combined (class 4), the result is a clear increase of difficulty.

6 Results

The algorithms *DNN* and *RNN* have been implemented in C++ and tested on 200 instances with 50, 100, 150 and 200 customers for each class (3200 instances in total) on an AMD Opteron^{TM} 244. The computational time allowed for the two procedures is 10 seconds for each instance. By keeping a constant computational time, it is possible to observe the relative efficiency of the *randomized nearest neighbor* when increasing the dimension and the difficulty of the instances tackled. The *deterministic nearest neighbor* in general does not use all the computational time

[4]Parallel experiments have been made for instances with different number of customers and the results appear qualitatively equivalent.



(c) Class 1 (d) Class 2

(e) Class 3 (f) Class 4

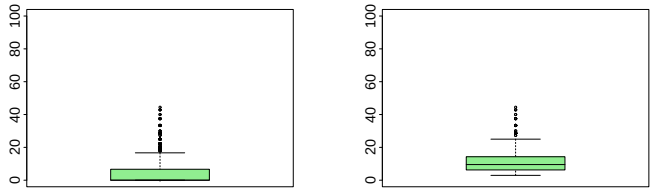
Figure 1. Representation of the percentage of feasibility obtained by randomly generated solutions for the different classes of instances varying the number of depots used.

Table 2. Values chosen for the parameters.

class \ parameter	α	β	γ
1	0.85	0.7	0.2
2	0.85	0.55	0.2
3	0.85	0.35	0.2
4	0.8	0.6	0.2

available, but it is hard to state how long it runs, since the local search procedure needs an amount of time that varies noticeably.

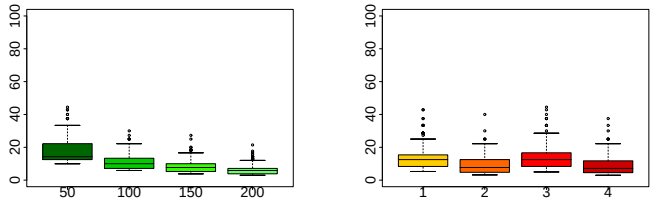
The three parameters that characterize the algorithms have been tuned using the F-Race procedure proposed by Birattari et al. [2] for each class of instances, considering a range of values for each one given by all the multiples of 0.5 in $[0.2, 0.85]$. The tests were performed considering only the one step *nearest neighbor* procedure, and the selected combinations have been used for both *DNN* and *RNN* and are reported in Table 2. Following Birattari [3, 4], we perform the experiments on a large set of instances, running the algorithms once on each of them. The first analysis of the data focuses on the number of vehicles required to complete the tours, being this the first objective. In 34.34% of the cases, the *RNN* improved the solution found by the *DNN*, and the distribution of this improvement is reported in Figure 2. The average difference is 3.7% (the distribution is reported in Figure 2(a)) and it it statistically significant according to the Wilcoxon Test with a 95% confidence level; moreover, if we consider only the instances for which an improvement is detectable, this average becomes 11.13% (the distribution is reported in Figure 2(b)). Figure 3, then, reports the same distribution, considering only the improved solutions, referred to each number of nodes belonging to the instances (Figure 3(a)) and to the different classes (Figure 3(b)). It is clear that the improvement decreases when the number of nodes



(a) Considering all the instances.

(b) Considering the instances in which the difference is not null.

Figure 2. Percentage improvement in the number of vehicles needed in the solutions found with *RNN* compared to those found with *DNN*.



(a) Instances grouped by dimension.

(b) Instances grouped by class.

Figure 3. Improvement in the number of vehicles needed in the solutions found with *RNN* compared to those found with *DNN* grouping the instances according to different criteria. The x axis represents either the number of customers of the instances (a), or the class to which the instances belong; the y axis represents the percentage improvement in the number of depots obtained by *RNN* over *DNN*.

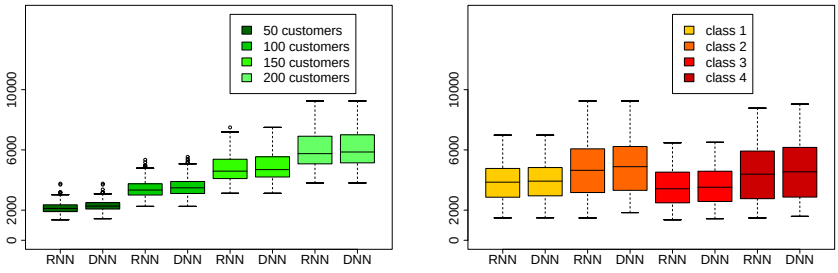
increases, as it was expected, since the computational time available is constant. The average improvement is nonetheless relevant, since for instances with 200 nodes it is equal to 6.26% in average. Finally, observing the trend obtained by grouping the results according to the classes of the instances, it is easy to detect a loss of efficiency of the *RNN* as the instances become more difficult.

The average number of vehicles needed to complete the solutions proposed by the algorithms for the instances grouped in terms of dimension and class is reported in Table 3. It is remarkable that when randomly generating solutions, as explained in Section 5, with a number of vehicles equal to the values found by any one of the algorithms (or the smallest integer that is greater than or equal to it) the percentage of feasible solutions found is null or almost null.

The second objective of the problem considered is the minimization of the total traveling time. Since hierarchically the number of vehicles is the first element to minimize, only if the two

Table 3. Average and percentage difference of the number of vehicles needed to construct the solutions proposed for the instances grouped in terms of dimension and class.

dim	*DNN*	*RNN*	Δ%	class	*DNN*	*RNN*	Δ%
50	6.99	6.50	7.01	1	10.60	10.17	4.06
100	11.20	10.76	3.93	2	17.18	16.66	3.03
150	15.86	15.44	2.65	3	10.32	9.92	3.88
200	20.81	20.41	1.92	4	16.76	16.36	2.39



(a) Instances grouped by dimension.

(b) Instances grouped by class.

Figure 4. Distribution of the total traveling time required by the solutions found with *RNN* and *DNN* grouping the instances according to different criteria, and considering only those for which the same number of vehicles is found from both algorithm.

Table 4. Average traveling time needed to complete the solutions proposed for the instances for which *RNN* and *DNN* obtain a final solution composed by the same number of subtours, grouped in terms of dimension and class.

dim	*DNN*	*RNN*	class	*DNN*	*RNN*
50	2293.57	2139.88	1	3914.89	3807.50
100	3552.24	3422.33	2	4863.80	4695.48
150	4868.61	4748.50	3	3621.76	3516.69
200	6108.81	5976.74	4	4655.06	4496.81

algorithms obtain a final solution composed by the same number of subtours, the traveling time becomes relevant. In this sense, the results presented here concern only the instances for which the number of vehicles used is the same in the solution found by *RNN* and *DNN*. The distributions of the results are presented in Figure 4, where as before the instances are grouped first by dimension (Figure 4(a)) and then by class (Figure 4(b)). As it can be seen considering both criteria the results of the *randomized nearest neighbor* are better than those of the *deterministic nearest neighbor*, following a quite stable proportion. The detail of the average of these values is presented in Table 4. The differences found are statistically significant according to the Wilcoxon Test with a 95% confidence level.

7 Conclusion

In this paper two algorithms for solving the rich vehicle routing problem characterized by multiple time windows and heterogeneous fleet of vehicles are proposed. The two procedures are based on the *nearest neighbor* heuristic introduced by Solomon [12], and one of them brings together this classical approach and the stochastic one.

The computational experiments has been performed on four sets of instances with different size and level of difficulty. A short analysis of the difficulty of the different classes of instances is also presented.

The results appear quite encouraging, since in a very short computational time the algorithm including the stochastic approach reaches much better results than its deterministic counterpart for all the classes and the dimensions of instances, obtaining improvements in 59% of the cases. The behavior of the *deterministic nearest neighbor* is, nonetheless, significantly better than the one of the random solutions generator.

This is a good starting point for investigating different approaches for this rich vehicle routing problem, that allows to think to many practical applications.

References

[1] Baptista S., Oliveira R.C., Zquete E., "A perio vehicle routing case study", *European Journal of Operational Research*, vol. 139, 2002, pp. 220–229.

[2] Birattari M., Stützle T., Paquete L., Varrentrapp K., "A racing algorithm for configuring metaheuristics", in *GECCO 2002: Genetic and Evolutionary Computation Conference*, New York, USA, 2002.

[3] Birattari M., "On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs?", Technical Report TR/IRIDIA/2004-01.2, Université Libre de Bruxelles, Belgium, 2004.

[4] Birattari M., "The problem of tuning metaheuristics, as seem from a machine learning perspective.", PhD thesis, Université Libre de Bruxelles, Belgium, 2004.

[5] Cordeau J.F., Laporte G., Mercier A., "A unified tabu search heuristic for vehicle routing problems with time windows", *Journal of the Operational Research Society*, vol 52, 2001, pp. 928–936.

[6] Gambardella L.M., Taillard E., Agazzi G., "MACS-VRPTW: A multiple ant colony system for vehicle routing problem with time windows", in *New idea in optimization*, edited by Corne D., Dorigo M., Glover F., McGraw-Hill, 1999, pp. 63–76.

[7] Goetschalckx M., Jacobs-Blecha C., "The vehicle routing problem with backhauls: An optimization based approach", Proceedings of the 2nd Industrial Engineering Research Conference, May 26-27, 1993, Los Angeles, California, pp. 504–509.

[8] Gronalt M., Hartl R.F., Reimann M., "New savings based algorithms for time constrained pickup and delivery of full truckloads", *European Journal of Operational Research*, vol. 151, 2003, pp. 520–535.

[9] Lin S., "Computer solutions of the traveling salesman problem", Bell System Tech. J 44, 1965, pp. 2245-2269.

[10] Pellegrini P., Birattari M., "Instances generator for the vehicle routing problem with stochastic demand", Technical Report TR/IRIDIA/2005-10, Université Libre de Bruxelles, Belgium, 2005.

[11] Prins C., "Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case", *Journal of Mathematical Modelling and Algorithms*, vol. 1, 2002, pp. 135–150.

[12] Solomon M. M., "Algorithms for the vehicle routing and scheduling problem with time window constraints", *Operation Research* 35, 1987, pp. 254–265.

[13] Tan K.C., Lee T.H., Chew Y.H., Lee L.H., "A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems", IEEE Congress on Evolutionary Computation, December 8-12, 2003, Canberra, Australia, pp. 2134–2141.

[14] Tarantilis C.D., Kiranoudis C.T., Vassiliadis V.S., "A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem", *European Journal of Operational Research*, vol. 152, 2004, pp. 148–158.

[15] Toth P., Vigo D., "The Vehicle Routing Problem", SIAM monographs on discrete matemathics and applications, 2002.

[16] Wassan N.A., Osman I.H., "Tabu search variants for the mix fleet vehicle routing problem", *Journal of the Operational Research Society*, vol. 53, no. 7, 2002, pp. 768–782.

Appendix

In Table 5 some statistics concerning the percentage of feasibility are reported. This measure is obtained using a random generator of solutions varying the class of instances considered and the number of vehicles. Each small table reports the value of the lower end of the whisker, the first quartile (25th percentile), the second quartile (median=50th percentile), the third quartile (75th percentile), and the upper end of the whisker of the distribution.

Table 5. Statistics related to the percentage of feasibility obtained by a generator of random solutions for the different classes of instances varying the number of vehicles used.

20 vehicles

class 1	class 2	class 3	class 4
97	96.1	96.8	96.4
97.9	97.2	97.7	97.45
98.2	97.6	98.1	98.0
98.65	98.3	98.5	98.3
99.6	99.2	99.5	98.9

19 vehicles

class 1	class 2	class 3	class 4
94.5	93.3	93.9	93.9
96.2	95.35	96.0	95.9
97.10	96.35	96.8	96.7
97.65	96.9	97.7	97.3
98.8	98.10	98.5	98.7

18 vehicles

class 1	class 2	class 3	class 4
91.8	89.7	90.8	90.5
94.0	92.55	93.8	93.45
95.15	93.8	95.1	94.5
95.85	94.75	96.05	95.6
97.70	97.1	97.8	97.8

17 vehicles

class 1	class 2	class 3	class 4
87.0	85.5	85.7	83.9
90.65	88.35	90.2	88.8
92.0	90.35	91.8	91.0
93.3	91.5	93.25	92.45
96.1	95.3	96.5	95.2

16 vehicles

class 1	class 2	class 3	class 4
79.5	73.0	79.1	75.1
85.05	81.25	84.2	82.6
87.4	84.65	87.2	85.8
89.45	86.85	88.9	87.85
94.3	91.3	92.8	93.4

15 vehicles

class 1	class 2	class 3	class 4
66.8	65.8	63.5	61.6
76.6	72.3	74.6	72.85
80.45	76.0	79.8	77.40
83.5	78.65	82.15	81.2
89.8	85.0	89.9	88.3

14 vehicles

class 1	class 2	class 3	class 4
55.7	49.0	52.8	45.6
65.8	59.65	64.2	60.2
70.75	65.0	69.15	66.85
74.5	69.0	72.4	71.25
81.8	80.9	81.6	81.7

13 vehicles

class 1	class 2	class 3	class 4
38.4	34.1	32.7	28.0
50.6	43.4	48.15	44.05
57.25	49.0	55.25	51.25
61.7	54.2	59.9	57.3
76.7	69.2	72.10	72.2

12 vehicles

class 1	class 2	class 3	class 4
18.9	13.8	16.2	13.7
35.8	27.7	31.3	27.4
41.65	33.7	37.7	34.5
47.40	37.65	43.8	40.3
60.8	48.7	58.5	56.3

11 vehicles

class 1	class 2	class 3	class 4
7.1	5.9	7.2	5.6
20.55	14.1	17.35	13.5
25.9	18.6	21.1	18.35
30.5	22.65	26.05	23.05
44.10	30.7	38.3	37.30

10 vehicles

class 1	class 2	class 3	class 4
2.4	1.2	1.5	1.2
9.6	4.95	6.4	5.0
12.5	7.3	9.25	6.85
16.1	10.1	12.1	10.2
25.1	17.1	20.0	17.5

9 vehicles

class 1	class 2	class 3	class 4
0.5	0.1	0.1	0.1
2.8	1.0	1.4	1.0
4.45	2.0	2.45	1.6
6.3	3.15	3.95	2.6
10.9	6.2	6.6	4.4

8 vehicles

class 1	class 2	class 3	class 4
0.0	0.0	0.0	0.0
0.5	0.1	0.15	0.1
1.0	0.3	0.4	0.2
1.85	0.6	0.75	0.3
3.6	1.3	1.6	0.6

1 to 7 vehicles

class 1	class 2	class 3	class 4
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0
0.3	0.1	0.0	0.0
0.7	0.2	0.0	0.0