A new approach to solving the Dial a Ride Problem using Iterative Local Search

M. Belvar[1], A. Gomez[1], P. Priore[1], J. Puente[1], and J. Parreño[1]
[1]Administración de Empresas, University of Oviedo, Gijon, Asturias, Spain

Abstract - *We present through this paper a simple Iterate Local Search for solving the Dial-a-Ride problem. The aims of this paper are to present the problem, its features and particularities and describe in detail our proposal. A detailed description is made for the main parts of our algorithm, especially the local search employed in each step in order to improve the solution. For this purpose, we have adopted a simple Variable Neighborhood Search using different local operators that make local movements between clients. To avoid falling in minimum locals a set of perturbations have been developed. The proposed algorithm has been tested against a set of well-known problems and a brief comparison between our results and the best results from literature is made in order to evaluate the efficiency and quality of our approach*.

Keywords: Iterative Local Search, Dial a Ride Problem, Metaheuristics, Logistics

1 Introduction

The Dial-a-Ride problem (commonly known as DARP) is an optimization combinatorial problem which goal is designing and creating routes and timetables that allow satisfy the necessities of a set of clients that require to be transported from one determinate point to another in a certain period of time. This problem has gained an increasing importance nowadays because there are more and more people that need to use this kind of services in order to move themselves between different places. For instance, is rather common that certain kind of people need to move from their house to hospitals or clinic centers during the day and it is not possible for them to use the public transport for many reasons (people that for example need adapted vehicles). Due to this fact, a large number of European cities have started to integrate these services in the last decades in order to improve the life-quality of those that require them.

Similarly, to the Vehicle Routing Problem (VRP) DARP is a NP-Hard transport problem deeply studied in the literature and they have many features in common, especially with some particular versions of the VRP like VRPTW (VRP with Time Windows) and VRPPD (VRP with Pickup and Delivery). In fact, we can see that DARP has many characteristics in common with a combination of both VRP versions (VRP with Time Windows and Pickup and Delivery). The main difference between these problems is that while VRP and its versions are focused on the delivery of goods DARP is more focused on the transportation of people (especially people in need). In the following chapters we will present the own particularities of the DARP problem.

There are different versions of the DARP. For instance, one could be according to the operation mode. In the static mode the whole set of requests are known before the journey starts. On the other hand, in the dynamic mode the requests are gradually revealed in real time throughout the day. However, this case is much less common because normally we know the timetables before the day starts. In terms of vehicles we can differentiate between versions with one or various vehicles. When just one vehicle is available it tries to serve all requests made by the users. Besides we can make a difference according to the vehicles features. Whether all vehicles are alike, we say that the fleet is homogeneous. Otherwise the fleet will be heterogeneous and each vehicle will have their own characteristics (some vehicles only have chairwheels, others only attend some kind of people, etc.). Finally, we also can establish different versions according to the number of deposits available; instances with just one deposit or instances with multiple different deposits. In our study, we will focus on the classic DARP problem with static mode using a fleet of n homogeneous vehicles and just one deposit.

Next section provides a brief review of literature and we continue presenting the problem specification as well as the mathematical model used in the resolution of the problem. Besides, we present our proposed algorithm, its features and particularities and how we have adapted it to solve the DARP. Finally, we make different experiments using a set of benchmark employed in the literature and we establish a brief comparison between the results obtained against the published in the literature.

2 State of the Art

DARP has received an increasing attention in the last decades. In the last years the resolution of this problem has gained a big relevance in the literature and numerous studies have been proposed for its resolution. Wilson et al [1][2] were the first in developing algorithms for solving real Dial-a-Ride

problems in Haddonfield (New Jersey) and Rocherster (New York) in the early 70's. In 1984 Jaw et al [3] published a heuristic based on a sequential insertion procedure so as to assign different clients to the fleet of vehicles available. Healy et al proposed in 1993 solving the DARP problem using a new extension of local search [4]. This extension, which they called sacrificing, was able to yield significant improvements over the routes without affecting the algorithm's running time. A simulated annealing (SA) was proposed by Baugh et al in 1998 [5] using clustering strategies. Cordeau and Laporte tried to solve the problem through a taboo search (TS) in 2003 [6]. In this work they presented an eight step evaluation scheme to determine the feasibility of a given route based on the forward time slack concept introduced by Savelsbergh [7] and employed it for reducing the individual ride time of each request on the tour. Three years later they proposed another similar approach using a branch-and-cut algorithm [8] that uses new valid inequalities for the DARP.

The first genetic algorithm (GA) was employed by Jorgensen et al in 2006 [9]. In this elaboration they based their work on the classical cluster-first, route-second approach where it alternates between assigning customers to vehicles using a GA method and solving independent routing problems for each vehicle through a heuristic. A summary called The dial-a-ride problem: models and algorithms was done again by Cordeau and Laporte [10] reviewing and summarizing the different DARP versions and algorithms proposed till the date and their results. Parragh et al made a great contribution in the DARP field by means of a Variable Neighborhood Search (VNS) [11]. In their paper, they aimed to solve a DARP instance using a VNS metaheuristic combining different concepts employed in the literature. A year later, they included a classic DARP variation in the service clients and the vehicles employed [12]. In that version they considered the vehicles as not homogeneous and clients with different needs.

The most recently works about DARP have been made by Kirchler [13] et al employing a granular taboo search with the objective of producing good solutions in a short amount of time (up to 3 min). Parragh et al proposed in [14] a new approach based on a hybrid column generation. It integrated VNS into column generation and combined it with large neighborhood search (called LNS). They were able to achieve good solutions in a reasonable time. We have used these results for establishing a comparison between Parragh's results and the ones returned by our algorithm.

3 Problem description

The DARP problem is represented on a direct graph G=(V,A) where V={v_0,v_1,…,v_2n} represents the set of vertices and A={(v_i,v_j): v_i,v_j ∈V,i ≠j} the set of arcs. Vertex v_0= v_2n and they represent the depot from where the vehicles start and finish their journey. There exist a total of n clients that have to be attended by m vehicles of capacity Q. Each pair (v_i,v_(i+n)) represents a request for transportation from origin v_i (pickup location) to destination v_(i+n)(delivery location). In each vertex v_i ∈V there are different information associated such as load q_i (representing the number of clients in the vehicle at the same time), a non-negative service duration d_i (representing time operations like pickup or delivery clients) and a time window [e_i,l_i] that determines the interval in which the client has to be attended by the vehicle and where e_i represents the minimum time to start the service in the vertex i and l_i determines the maximum time to start the service in i. Note that in each pair (v_i,v_(i+n)) one of the request is considered as critical (a tight time window is associated with it) and the other not (meaning that the time window is defined by [0,T] where T defines the end of the planning horizon). Always a new client is uploaded to a vehicle the load is increased by 1 (v_1,…,v_n) and decreased by -1 (v_(n+1),…,v_2n) when a client is delivered.

Each arc (v_i,v_j) has associated a certain non-negative cost ci,j representing how much cost travel from point i to point j. Furthermore, a good quality in the service is required so to achieve this goal is necessary to define a maximum travelling time L for each client. This represents the maximum time that a client can stay in a vehicle without deteriorate the quality of the service. Regarding to the vehicles, a vehicle that leaves a certain vertex i in D_i= B_i+ d_i implies that the vehicle will arrive to the vertex j in the instant A_j= D_i+ t_ij where t_ij represents the time travel from i to j. Due to the time windows associated with each vertex the service cannot start in a certain vertex i before the moment e_i and cannot exceed the moment l_i either. The start service in a certain vertex i can be defined as 〖 B〗 _i= 〖 max𝑓0(A〗 _i,e_i). Therefore, if the vehicle arrives to i before that the service can start (that is, A_i< e_i) it has to wait a certain time defined by W_i= B_i- A_i until the service can start. The sum of the travelling time of each client is established as L_i= B_(n+i)- D_i where B_(n+i) represents the start service in the associated pair n+i. The time duration of a route made by a vehicle k is equal to B_(2n+1)^(k)- B_0^(k).

3.1 Section and subsection headings

Let P the set of origin vertex P={1,….,n} D the set of destinations D ={n+1,….,2n} and N=P ∪D ∪{0,2n+1} the set of all requests plus the deposit. Let K the set of m vehicles that form the fleet. For each arc (i,j) and each vehicle k∈K, x_(ij)^k=1 represents the vehicle k travelling from the node i to the node j. B_i^k is the instant which the vehicle k starts it service in the vertex i. Q_i^k is the load of the vehicle k after visit i and L_(i)^k is the travelling time of a client inside the vehicle k.

We can summarize that the DARP goal is to minimize the sum of all travelling costs for each of the vehicles that form the problem. This is:

minimize

$$\sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ij}^{k}\, x_{ij}^{k} \qquad (1)$$

subject to

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^{k} = 1 \qquad \forall i \in P \qquad (2)$$

$$\sum_{j\in N} x_{ij}^{k} - \sum_{j\in N} x_{n+i,j}^{k} = 0 \qquad \forall i \in P, k \in K \qquad (3)$$

$$\sum_{j\in N} x_{0j}^{k} = 1 \qquad \forall k \in K \qquad (4)$$

$$\sum_{j\in N} x_{ji}^{k} - \sum_{j\in N} x_{ij}^{k} = 0 \qquad \forall i \in P \cup D, k \in K \qquad (5)$$

$$\sum_{i\in N} x_{i,2n+1}^{k} = 1 \qquad \forall k \in K \qquad (6)$$

$$B_j^k \geq (B_i^k + d_i + t_{ij}) x_{ij}^k \qquad \forall i,j \in N, k \in K \qquad (7)$$

$$Q_j^k \geq (Q_i^k + q_j) x_{ij}^k \qquad \forall i,j \in N, k \in K \qquad (8)$$

$$L_i^k \geq B_{n+i}^k - (B_i^k + d_i) \qquad \forall i \in P, k \in K \qquad (9)$$

$$B_{2n+i}^k - B_0^k \leq T \qquad \forall k \in K \qquad (10)$$

$$e_i \leq B_i^k \leq l_i \qquad \forall i \in N, k \in K \qquad (11)$$

$$t_{i,n+i} \leq L_i^k \leq L \qquad \forall i \in P, k \in K \qquad (12)$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{Q, Q + q_i\} \qquad \forall i \in N, k \in K \qquad (13)$$

Constraints (2) and (3) guarantee each request is attended just by one vehicle. The three following constraints (4, 5 and 6) force that the routes start and finish in the depot. With constraint (7) we establish that the start service in a certain request depends on the sum of the start service in the previous request plus the service time in that petition plus the travelling time between both nodes. In constraint (8) we guarantee the consistency in the vehicle load. In (9) the travelling time of each client cannot exceed the established limit L since otherwise there would be an excess in the travel. This condition is guaranteed also by constraint (12). In restriction (10) we force to the route duration to comply with the limit imposed. Finally, constraint (11) assures that each petition will be attended inside the limits of the time window associated and in constraint (13) we guarantee that the numbers of passengers inside a vehicle do not exceed the maximum capacity of it.

3.2 Solution evaluation

During the search we will not accept valid solutions, we must penalize those that do not satisfy the set of restrictions imposed in the previous paragraph. We propose the following evaluation formula according to Cordeau and Laporte in [7] in order to minimize the total routing costs.

$$f(s)=c(s)+ \alpha q(s)+ \beta d(s)+ \gamma w(s)+ \omega o(s) \qquad (14)$$

The evaluation function proposed can be interpreted as the sum of the distances travelled by each vehicle plus each of the constraints committed if they exist. These violations are expressed as the load excess $q(s) = \sum_{i=1}^{2n}(y_i - Q)^+$ the distance excess produced by the vehicle $d(s) = \sum_{k=1}^{m}(B_{2n+1}^k + B_0^k - T)^+$, the sum of the non-compliance with time windows $w(s) = \sum_{i=1}^{2n}(B_i - l_i)^+$ and the travelling time exceeded by each user $o(s) = \sum_{i=1}^{n}(L_i - L)^+$ Parameters α,β,γ,ω are the penalties coefficients used during the search and they are dynamically modified, being initially α =β= γ= ω=1. These values will be updated during the algorithm execution according to the evolution of the same. Thus, α is multiplied by a constant δ if in the last 10 iterations there were at least one solution unfeasible according to the load. If not, it is divided by δ. The same is done with β, γ and ω but with respect to the vehicle distance, the time windows violations and the user travelling time respectively.

4 Iterative Local Search

We propose solving the DARP problem using an Iterative Local Search (ILS) proposed by Lourenço et al [15]. The main idea behind the algorithm is making local search applying a perturbation in each iteration to allow the algorithm escape from local minimums.

In the following paragraphs we will detail and analyze the relevant aspects of the approach proposed and the aspects related to the ILS. These points are the generation of the initial solution, the mechanisms associated with the ILS such as perturbation mechanisms or the strategy of diversification/intensification, the graph reduction done before the algorithm starts and the operators employed for doing the local search by means of a VNS.

4.1 Initial solution and preprocessing

The generation of the initial solution follows the approach proposed by Medeiros [16], which proposal is similar to the approach used by Parragh et al in their work [11]. First, all requests are sorted in a list l by the median of the interval[e_i,l_i]. All vehicles are then initialized using the first k requests on the list (k represents the number of available vehicles). Remaining l-k requests are inserted sequentially (first origin, next destination) in each vehicle. The vehicle selected is the one that minimizes one of the four next criterions:

- Criterion 1). The distance between the origin of the last request in the vehicle and the origin of the client candidate to insert.
- Criterion 2). The distance between the origin of the last request in the vehicle with the destination of the client candidate to insert.

- Criterion 3). The distance between the destination of the last request in the vehicle with the origin of the client candidate to insert.

- Criterion 4). The distance between the destination of the last request in the vehicle with the destination of the client candidate to insert.

Using one of this criterions (randomly chosen) we select the vehicle more suitable for our current candidate (client) and its origin and destination will be inserted at the end of the vehicle. The process is repeated until all requests in the list l have been assigned to the different vehicles.

It is important to highlight that for each request (i,n+i) just one is defined as critical. That means that only this vertex defines a tight time window [e_i,l_i] and the non-critical vertex time window is defined by [0,1440]. Due to during the search will be important that all requests define a concrete time window it is necessary define a fictional time window for these vertices. A vertex with time window = [0,1440] means that the request is not forced to be attended in a concrete period of time and it can be served in any moment.

To define this fictional time window we follow the tightening techniques proposed by Cordeau [7]. Before, we have to make a difference between an inbound and an outbound request. A certain pair (i,n+i) can be inbound if it defines the tight time window in its origin vertex i but not in the destination n+i. On the other hand, a pair (i,n+i) can be outbound if it defines the tight time window in its destination n+i but not in the origin i.

4.2 Graph reduction

Due to the eight steps procedure described in [6] the DARP problem requires a big computational time, especially when the instance problem is large. We propose a graph reduction so as to reduce the number of feasible connections in the graph and the computational time required by each problem. We take into consideration the concepts of time window and origin/destination to determine what connections are possible and which are not. Summarizing, we can remove the following arcs:

Arcs from the depot to possible destinations. A connection depot $\rightarrow$X being X $\geq$n+1 is not possible because before visiting a destination the vehicle have to visit its associated origin.

In the same way a connection X $\rightarrow$depot being X $\leq$n is not possible because the last request in a journey has to be a destination.

Regarding to the time windows certain travels are not valid because they would produce impossible connections. Imagine a certain node A with an associated time window [80,120]. This means that the node A has to be attended in that interval. Now suppose another node B with an associated time window [30,60]. The travel A $\rightarrow$ B is not possible because when the service can start in A (minimum at the moment with value 80) and the vehicle arrive to B it will have overcome the limit l_b associated with B.

Another consideration we take into account is the next. We know that a maximum riding time L for each client is imposed in order to guarantee a good quality in the service.

Otherwise the travel from i to j is not possible.

We impose that the difference be less than 90 because that amount is the maximum riding time for each customer and long distances can affect other customers inside the vehicle (producing that their ride time increases). Thus, if the difference is greater than 90 we will fall into a riding time violation.

We will analyze the impact of this reduction in section 5 and we will see how this strategy improves the efficiency and results obtained by our algorithm.

4.3 Local search

We have implemented two different VNDs heuristics in the local search phase. Both heuristics exploit the neighborhood of the current solution making movements between requests. The first VND (we call it VND1) makes movements only between requests from different vehicles and it is applied first. Then, the second VND (called VND2) carries on optimizing each route applying movements between requests from the same route.

Two kinds of neighborhoods are included in the group VND1:

- Shift. A critical c and non-critical nc vertices are selected from a vehicle v_1 and removed from there. For each feasible position for the critical vertex c in v_2 we try to insert the non-critical vertex in its best possible position (according to the cost) in vehicle v_2. The positions for c and nc that reduces the fitness function the most are selected and both vertices are moved to there.

- Swap. A critical vertex c_1 and its associated request non-critical nc_1 are selected from a vehicle v_1. Similarly, a critical vertex c_2 and it associated request non-critical nc_2 are selected from a vehicle v_2. The swap is done exchanging the origins and the destinations. For this reason we have to detect for each pair c_1-nc_1and c_2-nc_2 what vertex is the origin and what is the destination.

Similarly, but inside each route we propose three neighborhoods used during the application of group VND2. These neighborhoods are:

Shift. Similarly, to the shift operator belonging to VND1 this operator makes shift movements between requests. Both pair critical and non-critical are removed from a vehicle position and inserted in the best possible position (that who reduces the fitness function) inside the same vehicle following the idea explained before.

WRI. This operator is introduced in [17]. The idea is move individual origin or destination vertices forward or backward in their respective routes. Note that a certain request just can be moved until a certain position (an origin cannot appear after a destination in a route in order to guarantee the consistency of the solution). By means of this operator the search attempts to reduce or remove existing infeasibilities and improve the objective function.

TwoOPT. This is the well-known 2-OPT operator which is applied on a sequence of points of one route and introduced in [18]. This operator is quite efficient on a single routing problem and it consists on reversing a sequence of points of one route removing a route connection and creating a new organization based on the best possible connection.

In each VND the neighborhoods are selected randomly. Each time a new best solution is founded the whole neighborhood is reset and the process starts again. Only improvement movements are accepted during the search.

To guarantee an adequate performance, here we use the concept of graph reduction introduced above. Before making a movement between two requests v_i and v_j we look if the connection is possible. Whether the travel is possible we evaluate the possible movement, otherwise the search continues without considering any movement. It allows reducing considerably the search space and the algorithm becomes faster.

4.4 Perturbations

Here we present the mechanisms employed to escape from local minimums. These perturbations play an important role in our proposal because they will make possible to explore a wider search space. In DARP we have noticed that perturbations have to be made with special care because a correct order in the requests have to be guaranteed (an origin request cannot be attended after its destination request) and we have to make feasible changes regarding to the time window of each client.

Besides, due to the problem characteristics, big changes in the current solution could lead the algorithm to lose the good characteristics achieved until the moment during the search and produce many changes in the next application of the VNS so that the time computing will be affected. On the other hand small changes could make the algorithm not able to escape from the local minimum generating cycles during the search and wasting computational time. Two different perturbations are selected randomly in each iteration during the search.

Insertion. The idea is similar to the swap neighborhood introduced by Parragh et al in their work [11]. Two random vehicles v_1 and v_2 are selected. From the first vehicle v_1 a randomly chain of ∂ requests is selected and remove from it. All requests forming the chain are removed from their original vehicle and inserted one-by-one in the vehicle v_2 first inserting the critical vertex and then the non-critical according to the best possible cost. That means that finally we will have inserted each request in the position that contributes more to reduce the vehicle cost (taking into account possible violations in the constraints imposed). Note that both critical and non-critical request must be deleted from the first vehicle v_1 even if it is not part of the sequence.

Finally, we will remove the pair (4,9) and the pair (5,10) from vehicle v_1once they are inserted in the best possible position in vehicle v_2.

Swap. Following the previous idea, we will select randomly two vehicles v_1 and v_2 again. We will repeat the previous process with a chain c_1 of length ∂ from vehicle v_1 and we will remove it from v_1. Next, the same is made on vehicle v_2 with other chain c_2 of length ∂. Finally, we will insert each request belonging to sequence c_1 in vehicle v_2 following the same idea that in the preceding perturbation (insertions critical/non-critical are made regarding to the best possible cost) and then we will insert each request belonging to sequence c_2 in vehicle v_1.

The ∂ parameter is known as the perturbation strength. Normally ∂ will be included between [1,K] being K the total number of vehicles in the current problem. A chain of size ∂=1 means that just two requests (belonging to the same client) will be selected and moved. We have noticed that the more clients there are in the problem, the more customers we have to move in order to produce good perturbations.

4.5 Intensification and Diversification

Taking the control over the intensification and diversification concepts during the search is important so as to ensure that the search produces good solutions. For this reason, once the algorithm starts its search the current solution is explored in order to exploit the good characteristics of it. While the search is advancing it is normal that after a number of iterations without improvements the good characteristics of the solution are lost. To avoid it, after a number of iterations without enhancing the current solution the best solution is perturbed and the search continues from there. The balance between intensification and diversification is controlled by the

parameter ρ and large values favor the diversification and it allows the algorithm to explore a higher search space. On the other hand, small values focus on exploiting the current solution and a more reduced space. In our proposal we take ρ=m being m the amount of vehicles in the current problem.

5 Experiments

The algorithm was implemented in C# and tested on the benchmark data set of Cordeau and Laporte [6] forming a total of 20 instances of different sizes. We compare our results with the best results obtained by the TS developed by them and the best results obtained by Parragh et al in [14] using their LNS algorithm. In this work, Parragh et al get better results than the previous obtained in their work in [11] and wasting less computational time. A total of 1000 iterations are imposed to our ILS for each instance. The algorithm could be stopped before reaching that limit if in the last 300 iterations the local search is not able to enhance the best solution founded till the moment.

5.1 Test instances

The set of instances used is formed by 20 different problems and the size is determined by the total number of clients (included between 24 and 144) to be attended. In each instance, the first n/2 instances have a time window on the destination while the remaining n/2 instances have a time window on the origin. For those n/2 instances that do not have a tight time window associated we make a preprocessing step before the algorithm starts as we described in section 4.1. For each vertex a service time d_i=10 is considered and the number of persons that is processed in each request is equal to 1. The whole set of instances consider a route duration limit equals to 480, the vehicles have the same capacity Q=6 and the maximum ride time for each customer is 90. The cost and travel time from a vertex i to a vertex j is equal to the Euclidean distance between these two vertices. The first 10 instances, which from now we will call group 'a', define wider time windows and, in general, the computing time necessary to solve them is less than the remaining 10 (that we will call group 'b').

5.2 Results

In this section we present the results obtained after running our algorithm against the benchmark explained above. We compare our results with the results obtained by Parragh et al in their work presented in [14]. Besides, we present the best solutions founded for each instance by Cordeau and Laporte using their taboo search. To establish the comparison, we take into account the best solution founded for each instance, the average fitness after 5 runs and the CPU time measured in minutes.

Table1 summarizes the performances of our proposal for each instance executed. In general terms, we have obtained acceptable solutions in a reduce time for the whole benchmark. We have noticed that our algorithm shows a good performance with small and medium instances and it is able to solve them in a few minutes.

Comparing the execution time between LNS and ILS we can conclude that our proposal is faster and it is able to solve the instances in much less time. In some cases, we have experimented that our algorithm reduces the CPU time significantly. On the other hand, we can conclude that problems of class 'b' are more difficult to solve by our ILS due to these instances need more computational time (the gain of time respect the LNS proposal is reduced).

Table 1: Main results

N	m	n	BSK	Hybrid LNS [14]		ILS	
				Best	**Avg**	**Best**	**Avg**
1a	3	24	190.02	**190.02**	**190.02**	**190.01**	**190.01**
2a	5	48	302.08	**301.34**	302.53	301.39	305.56
3a	7	72	532.08	535.28	538.21	536.04	549.28
4a	9	96	572.78	571.09	576.26	598.8	615.74
5a	11	120	636.97	629.52	637.59	670.64	682.15
6a	13	144	801.40	**788.88**	800.35	876.4	882.65
7a	4	36	291.71	**291.71**	292.56	296.51	301.88
8a	6	72	494.89	491.93	495.20	508.05	520.83
9a	8	108	672.44	661.47	676.09	722.16	746.26
10a	10	144	878.76	872.31	878.93	931.85	954.67
1b	3	24	164.46	**164.46**	166.06	**164.45**	165.51
2b	5	48	296.06	295.96	298.88	303.69	305.06
3b	7	72	493.30	**484.83**	491.29	508.11	515.15
4b	9	96	535.90	534.84	541.19	559.5	571.61
5b	11	120	589.74	587.67	590.22	612.08	625.99
6b	13	144	743.60	**738.01**	743.64	800.87	808.28
7b	4	36	248.21	**248.21**	**248.21**	**248.18**	256.56
8b	6	72	462.69	463.67	470.25	473.98	486.87
9b	8	108	601.96	**593.49**	606.25	632.97	651.55

5.3 Graph reduction impact

In order to demonstrate the improvement produced as consequence of the graph reduction presented in section 4.2, we will evaluate again a number of instances so as to establish a comparison between the algorithm without applying a previous graph reduction and the algorithm applying it. Due to the time computing needed to carry out these experiments, we just consider a reduced set of instances. For this purpose, we have selected five problems of group 'a' and five problems of group 'b'. The number total of ILS iterations is set to 500 and we use the same parameters configuration employed in the results presented in section 5.2. Next table shows the results obtained by the application of both techniques. ILS-NGR represents the algorithm without graph reduction strategy and ILS-GR is the proposal presented in this paper.

Table 1. Analysis of the graph reduction impact

N	ILS-NGR		ILS-GR		Time reduction (%)
	Best	CPU	Best	CPU	
2a	312.05	1.4	302.54	1.23	12.14
3a	567.6	2.52	571.85	2.3	8.73
5a	721.36	8.52	684.7	7.5	11.97
7a	296.86	0.57	296.51	0.45	21.05
10a	1022	23	977.06	15.83	31.17
1b	171.85	0.27	168.52	0.28	-3.7
4b	597.25	7.33	568.46	7.17	2.18
8b	498.52	4.02	479.83	4.28	-6.47
9b	683.45	13	668.07	11.22	13.69
10b	909.52	22.67	920.7	19	16.19

.

Although in some cases the non-application of the graph reduction yields better results regarding to the time the quality of the solutions is much worse in general terms (except in two cases where the no application of the graph reduction get better results). This is due to the search space reduction produced by the inclusion of the technique explained. In fact, the inclusion of the technique allows the algorithm to get a good balance between time computing and quality solution.

6 Conclusions

An Iterated Local Search based on Variable Neighborhood Search has been proposed to solving the DARP. During the search two kinds of VNDs are used in order to improve the solution. Besides we present the perturbations used as mechanisms to avoid local minimums and the concept of graph reduction. This graph reduction allows the algorithm to reduce considerably the time computing and to enhance the results obtained as well. In order to measure its impact, we analyze two different proposals where one uses this strategy and the other no. Once presented the results of this experiment we can see as our strategy allows the algorithm to get solutions in less computational time.

The proposed algorithm was tested against the benchmark created by Cordeau and Laporte and the solutions obtained were compared with the most significant previous studies done in the literature. Although the quality of our solutions is not as good as the solutions obtained by Parragh, in general our algorithms is able to get acceptable solutions employing less computational time.

7 References

[1] Wilson, N. &. (1971). Scheduling algorithms for dial-a-ride systems.

[2] Wilson, N. &. (1976). Advanced dial-a-ride algorithms research project: Final report. Technical Report 76-20, Massachusetts.

[3] Jaw, J.-J. &. (1986). A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. Transportation research, 243-257.

[4] Healy, P. &. (1993). A new extension of local search applied to the Dial-A-Ride Problem. European Journal of Operational Research, 83-104.

[5] John, W. &. (1998). Intractability of the Dial-a-Ride problem and a multiobjective solution using simulated annealing. Engineering Optimization, 91-123.

[6] Cordeau, J. &. (2003). A tabu search heuristic for the static multi-vehicle. Transportation Research Part B, 579–594.

[7] Savelsbergh, M. W. (1985). Local search in routing problems with time. Ann. Oper. Res. 4, 285–305.

[8] Cordeau, J. (2006). A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. Operations Research, 573-586.

[9] Jorgensen, R. &. (2006). Algorithms, Solving the Dial-a-Ride problem using Genetic. Journal of the Operational Research Society, 1321–1331.

[10] Cordeau, J. &. (2007). The dial-a-ride problem: models and algorithms. Ann Oper Res 153, 29–46.

[11] Parragh, S. &. (2010). Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 37, 1129-1138.

[12] Parragh, N. (2010). Introducing heterogeneous users and vehicles into models. Transportation Research Part C, 912-930.

[13] Kirchler, D. &. (2013). A Granular Tabu Search algorithm for the Dial-a-Ride Problem. Transportation Research Part B, 120-135.

[14] Parragh, N. &. (2013). Hybrid column generation and large neighborhood search. Computers &Operations Research, 490-497.

[15] Lourenço, H. R. (2002). Iterated Local Search. En F. &. Handbook of metaheuristics, 321-352.

[16] Medeiros, P. (2016). Solving the Dial-a-Ride Problem Using. Porto Alegre, Brazil.

[17] William P, N. &. (1999). Solving a Dial-a Ride Problem with Hybrid Multi-objective Evolutionary Approach: Application to Demand Responsive Transport. Transportation Research Part B 34, 107-121.

[18] Chevrier R. &. (2012). Solving the pickup and delivery problem with time windows using reactive tabu search. Applied Soft Computing 1247-1258.