Improving Paratransit Scheduling using Ruin and Recreate Methods

Carl H. Häll[*][†] Anders Peterson[*]

Abstract

We study the effects of using ruin and recreate methods in a replanning phase of a dynamic dial-a-ride problem. Several such methods are proposed, and a modeling system is used to evaluate how they improve the quality of the solutions. We show that simple changes to existing planning methods can increase the efficiency of the service. Two cases, with different forms of costs inflicted on the vehicles, are evaluated and significant improvements are found in both cases. The best results of our study are found with ruin methods based on removal of sequences of requests.

1 Introduction

Traditional public transport systems are most often insufficient to provide good public transport for everyone. It is not always possible for everybody, especially the elderly and the handicapped, to use a regular system consisting of fixed-route, timetabled, services. Typically there is therefore some type of paratransit offered to these groups of customers. This type of transportation is often quite expensive. The net cost for producing nearly 11,000,000 one-way paratransit trips in Sweden during 2008 was approximately € 270,000,000 (SIKA, 2009b). This gives us an average net cost per trip of some € 25, which can be compared to the corresponding net cost per trip in the regular public transport system, which is € 1.10 (SIKA, 2009a). It is believed that there is great potential for improved efficiency in the paratransit system, with remained quality and level of service.

One important form of paratransit is the dial-a-ride service, often operated as a door-to-door service, where customers are served on demand and rides are coordinated via a call-center. In some systems, the requests must be made a certain time in advance, often 24 hours. In other systems, customers can request almost immediate service. In the first case, known as the static case, all requests are known in advance, and therefore the planning can also be made in advance. In the second

[*]Linköping University, Dept of Science and Technology (ITN), SE-60174 Norrköping, SWEDEN
[†]E-MAIL: carha@itn.liu.se

case, known as the dynamic case, requests are received and planned in real-time, and hence, the planning must start before all requests are known. In this paper, we address the planning of dynamic dial-a-ride services.

The problem of deciding which vehicle shall serve what request, and at what time, is known as the dial-a-ride problem (DARP). This problem is well described in the literature, see e.g. Cordeau (2006), Cordeau and Laporte (2007), and Parragh et al. (2008). During the recent decades, the development of optimization methods for transportation problems such as the DARP, has been substantial. Still, very little of these research advances have been implemented in real-world dynamic dial-a-ride systems. Therefore, the practical planning of dynamic dial-a-ride services is often performed in approximately the same way today as it was done several years ago. For this reason, we believe that there is a large potential for improving the practical planning of dynamic dial-a-ride services.

Some systems for practical planning of dynamic dial-a-ride services merely consist of an insertion heuristic to insert new requests into the current schedule. Other systems also apply some reassignment procedure to improve an existing schedule. Such a reassignment procedure can be run as a background process, while waiting for new requests to insert. This paper is devoted to the development of efficient reassignment procedures that can be used in real-world systems for the planning of a dynamic dial-a-ride service.

A reassignment procedure can be interrupted by incoming requests at any time. When this happens the program must return to the last known complete schedule before inserting the new request. The more complicated the reassignment calculations are, the more likely is an interruption leading to no improvement. This is why simple local search procedures, based on greedy moves, are often used as reassignment procedures for practical planning of dynamic dial-a-ride services.

One problem with such local search methods, is that they tend to get trapped in local minima (see e.g. Voudouris et al., 2010). To escape from such minima, the local search method can be combined with some other method, in this way creating a metaheuristic. Metaheuristic methods, as for example simulated annealing or tabu search, have been successfully used for several types of vehicle routing problems and are well described in the literature (see e.g. Gendreau and Potvin, 2010). In an application of the dynamic DARP however, a metaheuristic approach is computationally demanding and not always the best option. We also note that the recurrent insertion of new requests continuously will change the set of feasible vehicle itineraries, and hence helps in avoiding to get trapped in local minima.

Balancing computational efficiency against best possible results, our attention has been drawn to so-called ruin and recreate methods. Ruin and recreate is a class of methods based on the extraction and reinsertion of subsets of all scheduled activities. For an overview of the concept, we refer to e.g. Schneider and Kirkpatrick (2006).

Ruin and recreate methods have previously been used for various routing problems. A pioneer work was presented by Shaw (1998) who suggested a "large neighborhood search" to solve the vehicle routing problem. His ruin method is inspired by the shuffling techniques used in job scheduling problems and he uses a branch-and-bound tree for the reinsertion. The term "ruin and recreate" was however suggested first two years later by Schrimpf et al. (2000), who, independent from Shaw, developed an all-purpose-heuristic for complex optimization problems, which recorded breaking numerical results for both the Travelling Salesman and the Vehicle Routing problems. We will return to Schrimpf et al. (2000) when discussing our reassignment procedure in Section 3.

Yepes and Medina (2006) use a ruin and recreate method as part of a heuristic for the vehicle routing problem with a heterogeneous fleet of vehicles and soft time windows. The ruin and recreate part of the heuristic is used to minimize the number of used vehicles, and focuses on complete removal of the shortest routes.

In Pisinger and Ropke (2007), a set of alternative ruin and recreate methods are described, and applied to different variants of the vehicle routing problem with time windows. Goel (2009) uses a randomized ruin and recreate method for a vehicle scheduling and routing problem that considers regulations for the drivers' working hours, and Wen et al. (2010) use a ruin and recreate method to restart a tabu search heuristic for solving the dynamic multi-period vehicle routing problem. To the authors' knowledge, there is no previous work done on how to use ruin and recreate methods to improve the planning of dynamic dial-a-ride services operating under real-world conditions.

In this paper, we study how various ruin and recreate methods perform in planning dynamic dial-a-ride systems. The aim is to find methods that improve the planning and can be used in existing real-world planning systems. We use the modeling system DARS, described in Häll et al. (2011), to evaluate the methods. This system is built on two planning algorithms: one insertion algorithm that is used to insert each new request into the current schedule, and one reassignment procedure that is continuously run to improve the solution. The insertion procedure simply evaluates all possible insertions and chooses the one that gives the lowest incremental cost. The reassignment procedure is the one we implement a number of versions of, in order to evaluate the various ruin and recreate methods.

The rest of this paper is organized as follows. Section 2 describes the type of dial-a-ride service that we consider. In Section 3, we present a number of different reassignment methods. Section 4 presents the input and settings to our tests. The results are then given in Section 5, and Section 6 presents conclusions and some ideas for future research.

2 A dial-a-ride service and its characteristics

Dynamic dial-a-ride services can be designed and operated in various ways. Every real-world instance of a dynamic dial-a-ride service has a number of specifications that define the service and the corresponding degree of freedom for planning and for reassignment of requests. Here, we will describe a representative case, taken from a real-world application. This case is later used in the numerical experiments. Similar setups (with similar parameters) exist in other real-world applications.

Customers can call in their requests to a call center, at any time, far in advance or almost at the moment when they whish to be served. Each customer specifies a pick-up time, a pick-up location, a drop-off location, the number of persons to be transported and if wheelchair spaces or seats with extra legroom are required. The customer and the operator agree on a planned pick-up time (PPT). This time restricts the replanning possibilities, as will be described in Section 2.1. The allowed computation time for creating a feasible schedule is limited by the fact that the customer must be given a PPT while still on the phone. The schedule specifies how each known request shall be served, i.e. it includes the itineraries of all vehicles. This means that at any given time, there must always be a valid schedule available.

The number of vehicles available is assumed to be infinite, but the total cost depends on the number of vehicles used. The vehicle fleet consists of two different types of vehicles, ordinary taxis and special vehicles with the capability to transport passengers in wheelchairs.

Vehicles can only be given new assignments when they are at a standstill, i.e. picking up or dropping off passengers. This means that a vehicle can never be redirected when it is on its way between two addresses.

The required service time, i.e. the time for picking up and dropping off passengers, is assumed to be equal for all requests, but a certain extra time is added if the customer uses a wheelchair.

All drive times are assumed to be predetermined, i.e. delays due to e.g. rush hour traffic or accidents are not modeled. Other occurrences and stochastic events such as customers who do not show up, cancellations etc. are also ignored in the model.

2.1 Service Constraints

The quality level of a dial-a-ride service is controlled by a number of specifications, which we refer to as service constraints. In this Section, we describe how time window constraints, maximum ride time constraints and vehicle capacity constraints are handled in the specific case we consider. For an extensive overview and comparison of level-of-service measurements, we refer to the review by Paquette et al. (2009).

Time Windows

In real-world applications, customers are rarely guaranteed to be picked up at the exact time they request. Instead, a time window, within which the customer must be picked up, is applied. These time windows expand the solution space, compared to what it would be if customers were to be guaranteed to be picked up at a specific time.

Mimicking real-world conditions, we use two different types of time windows. The first window (here 30 minutes around the desired pick-up time) is used to decide a PPT which is given to the customer, while on the phone. The actual pick-up, is then allowed to be delayed a certain number of minutes from the PPT. This defines the second time window, which is often quite small; in our application fixed to 10 minutes. It is only within this second window that a request can be reassigned.

An example clarifies how the time windows are used. Assume that a customer wants to be picked up at 12.20. The operator searches for a suitable pick up time in the interval $12.05 - 12.35$, and might return 12.30 as the PPT. The vehicle then has to pick the customer up in the interval $12.30 - 12.40$. All vehicles' itineraries may be reconstructed arbitrarily as long as one of them can fulfill the requirement of picking up the customer within this interval.

Maximum Ride Time

The maximum ride time for a customer (the total time spent in a vehicle) is limited to the direct drive time multiplied with a factor α. For short journeys, with a direct drive time less than γ minutes, however, an additional β minutes are allowed. We have used $\alpha = 200\%$ and $\beta = \gamma = 15$ minutes.

Vehicle Capacity

Vehicle capacities must not be exceeded. Three different types of capacities are available for each vehicle; the number of wheelchair places, the number of seats with extra legroom and the number of standard seats. The capacities for the two vehicle types are described in Table 1.

Table 1: Capacity of vehicles

Vehicle type	Number of standard seats	Number of seats with extra legroom	Number of wheelchair places
Taxi	3	1	0
Special vehicle	0	3	3

A customer, who can be placed in a standard seat, can also be placed in a seat with extra legroom.

2.2 Objective Function

The objective for our instance of the DARP, and hence also for the reassignment procedure, is to minimize the total cost. The total cost is built up by costs related both to vehicles and customers. The vehicle-related costs include fixed costs, running costs and waiting time costs. Customer-related costs include excess ride time costs and waiting time costs.

3 Reassignment Methods

We have studied a number of ways of selecting what requests shall be removed from the current solution (in the ruin part), and a number of ways to reinsert them again (in the recreate part). The two parts are independent and will be executed in sequence.

In all our methods, we only consider complete requests, that is, both the pick-up node and the drop-off node are either reassigned or kept. In a generic method, however, this does not have to be the case. Since each pick-upped customer must be subsequently dropped-off from the same vehicle, however, only ruining one trip end strongly bounds the reinsertion possibilities. Further, from a practical point of view, by keeping the requests together, we can use the same reinsertion procedure as is used when the requests are initially called in. For a straight-forward presentation we also let the word "request" (rather than "node") refer to the smallest unit to be removed and reinserted.

The methods are implemented in such a way that only greedy moves are accepted, which in our case means that if the new schedule is better than the old one, the new schedule replaces the old; otherwise the old schedule is resumed.

3.1 Ruin Methods

Schrimpf et al. (2000) have categorized ruin methods to remove requests, for generic routing problems into three categories: Random ruin methods, Sequential ruin methods and Radial ruin methods. In a Random ruin method one or several requests are randomly chosen to be removed from the schedule, whereas the requests to be removed using a Sequential ruin method must be a sequence of consecutive requests planned to be served by one vehicle. Radial ruin methods in contrast, require closeness in some metric space between the requests to be removed.

Hence, the Sequential and Radial ruin methods are local (as long as only a limited neighborhood is considered), whereas Random ruin methods are global, i.e. applies to the complete schedule, as soon as two or more requests are removed.

Removing more than one single request means that the number of requests to remove must be decided. This can be determined explicitly as a given number, or implicitly as the number of requests that happen to be presently assigned in a temporal or spatial neighborhood. In the latter case, a certain fluctuation in the number of requests to be removed is introduced between consecutive calls of the procedure.

Single Random Request

This method ruins the solution by simply lifting one randomly chosen request at a time from the schedule. This is a simple method that is used today in many tools for operational planning, and in the computational tests presented later it is considered as the default method, to which all other methods are compared. We note that this method can be interpreted as a minimum extreme case of all Random, Sequential and Radial ruin methods.

Multi Random Requests

The simplest modification to the original method is to lift multiple requests from the schedule at the same time. The requests are chosen randomly, and the procedure creates voids in all the vehicle itineraries where a request has been lifted. This enlarges the level of freedom for the recreation part.

Single Vehicle Segment

In this method, one random request is first chosen to be lifted from the current schedule. Thereafter, a given number of requests subsequently scheduled to that vehicle are also lifted. The intention is to enable greater improvements by creating a longer continuous void in the itinerary for one vehicle. The lifted requests may be kept in sequence and reinserted at the beginning or end of another vehicle's route, but can also be split up on various vehicles.

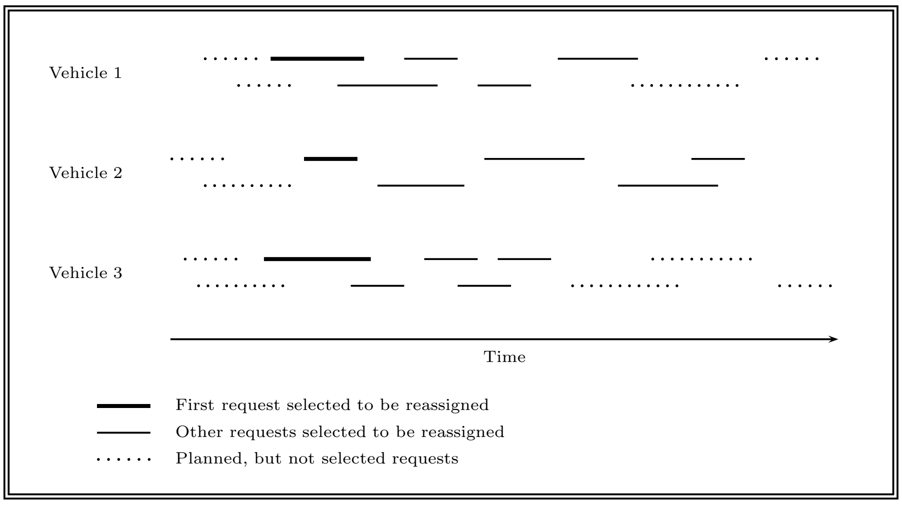
Multi Vehicle Segments

In a Sequential ruin method, as described in Schrimpf et al. (2000), all requests to be removed must belong to one single segment. A natural extension is to simultaneously consider multiple segments. The Multi vehicle segments method starts by randomly choosing one request to be lifted from the current schedule. One additional request each from a specified number of vehicles should then be chosen. The additional requests must be sufficiently close in time to the first chosen request. After this, a number of subsequent requests are lifted from each of the chosen vehicles, as is done in the Single vehicle segment method described above. An example of this method where five requests are lifted from each of three vehicles is described in Figure 1.

The advantage of using multiple vehicles is that voids may be created in the schedule so that requests lifted from one vehicle can be fitted into the itinerary of another vehicle.

Early Segments

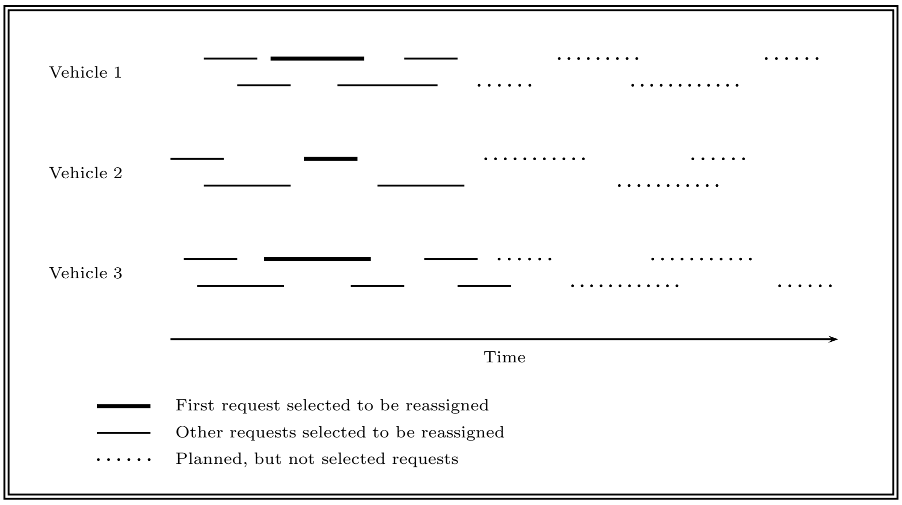
This method is similar to the previous one. A number of requests that have been scheduled on different vehicles, with some temporal overlap, are chosen. All requests that are scheduled on these vehicles, but have not yet started to be served, will then



Vehicle 1
Vehicle 2
Vehicle 3
Time
First request selected to be reassigned
Other requests selected to be reassigned
Planned, but not selected requests

Figure 1: Description of the "Multi vehicle segments" ruin method

be added to a list and ordered according to the PPT. A specified number of requests taken from the beginning of this list are lifted from the schedule. Figure 2 describes how requests are chosen according to this method, and emphasizes its difference to the Multi vehicle segments method described in Figure 1.



Vehicle 1
Vehicle 2
Vehicle 3
Time
First request selected to be reassigned
Other requests selected to be reassigned
Planned, but not selected requests

Figure 2: Description of the "Early segments" ruin method

The difference between the two methods is that in the Early segments method, it is always the first scheduled requests that are chosen, whereas in the Multi vehicle segments method it is always a given number of requests lifted from each of the chosen vehicles. This difference can be seen from a comparison of Figure 1 and Figure 2.

Spatial Distance

One random request is first chosen. Other requests whose pick-up nodes are within a specified distance in space from the first request's drop-off node are then selected. The argument for using this method is that it can simultaneously remove two requests that are potentially suitable for scheduling directly after each other on the same vehicle. The same argument also holds for the following three methods, which are based on different distance measures.

Temporal and Spatial Distance

This method uses a combination of temporal and spatial distances. The spatial distance is defined as in the previous method, whereas the temporal distance is defined as the difference in PPT between the requests. One unit of time is weighted as the distance a vehicle can travel during that time. One random request is first chosen. Then, other requests with the smallest combined distances from the random request are also chosen.

Temporal Distance 1

One random request is first chosen. Then other requests whose PPT is within a certain time interval from that of the chosen request are also chosen. If there are too many requests in the given time interval, this is shortened until a specified maximum number of requests remains.

Temporal Distance 2

All requests whose PPT is within a certain time interval ahead of the current time are chosen. As above, the time interval is shortened if necessary.

All planned Requests

All planned requests are lifted from the schedule. The idea of replanning the entire solution can be good, but since only a short computation time is available, the recreation part is likely to be interrupted (and the old schedule resumed) more often than for the other methods. Analogously to the Single random request ruin method, we note that this method can be interpreted as a maximum extreme case of all Random, Sequential and Radial ruin methods.

3.2 Recreate Methods

For the reinsertion of requests that have been lifted from the schedule, we use a simple insertion heuristic, identical to the one which is used for the initial assignment of requests as they are called in. The recreate methods merely differ in the way the requests are sorted before being reinserted. Three recreate methods are considered.

Sorting by Direct Drive Time

The requests are sorted by the time it takes to drive directly from the pick-up node to the drop-off node. The request with the longest direct drive time is inserted first. The motivation for this is that if long trips are inserted last, it is unlikely to find feasible insertions without the use of additional vehicles.

Sorting by Geography

In this method, the requests are sorted according to the pick-up node coordinates. Requests are ordered with pick-up nodes from west to east. The idea is that this should enable more coordinated pick-ups.

Sorting by PPT

This method sorts the requests according to their PPT. The request with the earliest PPT is inserted first. In this way, the vehicle schedules are most likely filled in order of time, which can reduce the slack time between two consecutive requests on a vehicle.

4 Test Scenario and Settings

The computational tests are performed in two steps, preliminary tests and full-scale tests. The preliminary tests are made to estimate the potential of the various ruin methods and the various recreate methods, and also to trim parameters in the methods. Accurate full-scale tests are computationally demanding and results are therefore computed only for selected methods. The input to the test scenarios is described in Section 4.1, and the results from the preliminary tests are described in Section 4.2. The full-scale tests are reported in Section 5.

4.1 Test Scenario

The tests have been made using real world data from the Swedish of city Norrköping, with 130,000 inhabitants. This problem instance represents a typical Swedish dial-a-ride service. Our test case contains 606 requests, corresponding to one day's passenger transportation. Some calls are made several weeks in advance whereas others are requesting service as soon as possible, possibly to be scheduled on vehicles that are already in operation. The distribution of requests over the day is presented in Figure 3. This is a typical distribution of paratransit requests, with peak-periods in the morning and afternoon that are closer together and not as significant as for regular public transport.

The travel times between two locations are pre-calculated values, based on aggregated geographical zones. The travel time between two zones is thereby assumed

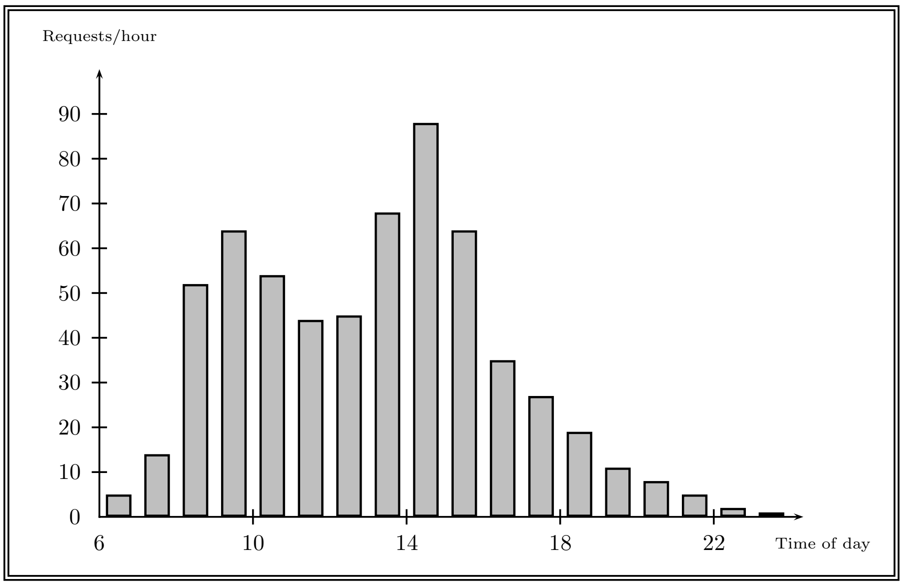


Figure 3: The distribution of requests in time

to be independent of where in the zones the actual pick-up and drop-off points are located. The city of Norrköping is divided into 164 such zones.

It would be ideal to run each simulation in real-time, but due to the time it takes to do so, this is not possible. When performing simulations in DARS, there is a time factor stating how many times faster than real-time the simulations should run. A problem with using a great value for this time factor is that a replanning method that performs well but needs at lot of computation time may perform worse in the tests than it would do in a real world planning situation. This is because that such a method is more likely to get interrupted, and with a greater value of the time factor it will get interrupted even more often. A fast method however, might be able to finish without getting interrupted as often and consequently it will be evaluated as a better method in the tests. The choice of time factor is a trade-off between performing many experiments and more accurate experiments. In the preliminary tests we have set the time factor to run ten times faster than real-time, and in the full-scale tests it runs four times faster than real-time.

Tests are made for two different scenarios. In the first scenario, called *internal vehicles*, it is assumed that all vehicles belong to one single fleet of vehicles. In this case, it is further assumed that a vehicle that is used at any time during the day (even for serving only one request), must be paid for the entire day. In the second scenario, called *external vehicles*, it is assumed that the vehicles are procured resources, charged for only when actually used. The cost of a vehicle is then based on the time from when a vehicle leaves the depot until it is back in the depot again. The reason for studying the effects of these two scenarios is that some real-world services use internal vehicles and some use external (and quite often a combination of internal and external vehicles). Hence, it is important that a computation procedure for dynamic dial-a-ride services is efficient for both these scenarios. The preliminary tests that follows in the next section are based on the internal vehicles scenario, while the results of the full-scale tests presented in Section 5 are based on both scenarios.

4.2 Preliminary Tests

The preliminary tests have been performed in three steps. In the first step it was tested how the Multi random requests method works with different number of requests to be lifted. In the second step it was evaluated how the various recreate methods worked in combination with the Multi random requests ruin method (using the number of requests that performed best in step one). In the third step, tests were made to see how the other ruin methods performed in combination with the recreate method that performed best in step two.

The preliminary tests have shown that the Multi random requests ruin method performs best when five requests are picked out each time the reassignment method is run, and that the recreate method based on sorting according to PPT performs better than the other methods. To sort by direct travel time performs almost as good as sorting by PPT, while sorting with the geographical method performs significantly worse.

The results from the third step are given in Table 2, i.e. it gives the objective function value for each ruin method. From these results it can be seen that already the simple method of lifting multiple requests (the Multi random requests method) has outperformed the Single random request method which is often used in operational planning today. However, the two ruin methods that have performed best are the Multi vehicle segments and Early segments. Further, the preliminary tests have shown that these methods perform well when Multi vehicle segments lifts up to eleven requests from three vehicles each (if that many requests are planned on each of the vehicles), and Early segments lifts a total maximum of 33 requests from up to three vehicles.

Based on the results from these preliminary tests, three ruin methods have been selected for a more careful study and comparison to the default method (the Single random request method). These are: Multi random requests, Multi vehicle segments and Early segments. The Multi random requests method has been chosen since it is very easy to implement in existing planning systems. The Multi vehicle segments and Early segments methods have been chosen since these are the two methods that have performed best in the preliminary tests. The numerical results in Section 5 are based on these four ruin methods in combination with the recreate sorting according to PPT. The simulations have been performed on a standard office computer with an Intel Core 2 Duo Processor of 1.83 GHz and 2.5 GB RAM.

Table 2: Results of preliminary tests

Ruin method	Objective function value
Multi vehicle segments	218 242
Early segments	218 864
Temporal and spatial distance	220 825
Multi random requests	226 060
Single vehicle segment	232 964
Spatial distance	233 692
Temporal distance 1	233 866
Single random request	239 847
Temporal distance 2	247 742
All planned requests	255 660
Without reassignment	315 120

5 Numerical Results

The proposed methods have been evaluated on the basis of one single historical day's requests. To avoid any artefactual results, five simulations are performed for each method. The results of the simulations are described in Section 5.1–5.2.

When evaluating the results of the reassignment methods, the objective function value is of course the main evaluation criterion. This function includes costs related both to vehicles and to customers. To assess the quality of a solution, it is also interesting to study what has caused the result. These quality measurements are included in the objective function, but as individual measurements the practical impacts of them become clearer. The criteria used are the following:

- Objective function value
- Number of vehicles needed
- Total drive time
- Total drive time with passengers
- Total drive time without passengers

5.1 Results for internal Vehicles

In Figure 4 it is shown how the objective function value is affected by the chosen reassignment method. For each method, the horizontal line gives the average value of the objective function values from the five simulations, and each mark corresponds to the objective function value for a specific simulation. We can see that the simple method to remove multiple random requests (in this case five random requests) already reduces the mean objective function value by 2.2%. Our best approach, Multi vehicle segments, lowers the mean objective function value by 8.5%.

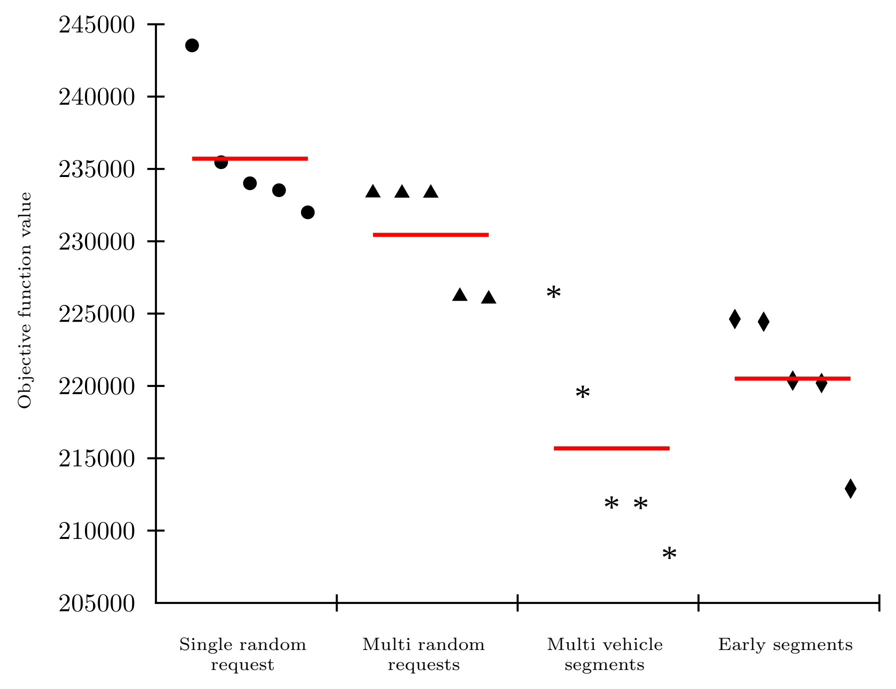


Figure 4: Objective function values for simulations of internal vehicles. Each mark is the objective function value for one simulation and the horizontal lines are the average values of the simulations made with each method

The distinct steps in the objective function values of the various solutions in Figure 4 is due to the initial cost of taking one additional vehicle into operation, which is much higher than the other cost-components of the objective function. Consequently, in the case with internal vehicles, the major improvements in the objective function value are due to the fact that the best approach here is more successful in minimizing the number of vehicles that need to be in operation simultaneously (at peak hour). The simulation that caused the highest objective function value (based on the Single random request method) needed 35 vehicles, while the simulation that gave the lowest objective function value (based on the Multi vehicle segments method) only needed 30 vehicles.

All methods have a spread on objective function values from the five respective simulations. Still, however, the worst outcome from Multi vehicle segments and Early segments are both better than the best outcome for the Single random request method. The average objective function value for the Multi random requests method is also significantly better than that of the Single random request method, even though the result of every single simulation of the Multi random requests method is not better than the best repetition of the Single random request method.

Table 3 describes how the evaluation criteria have changed with the various methods. Note that the values in the table are averages of the five simulations of each method.

Table 3: Values of the evaluation criteria for internal vehicles

	Single random request	Multi random requests		Multi vehicle segments		Early segments	
Objective function value	235702	230406	-2.2%	215679	-8.5%	220500	-6.4%
No of vehicles needed	34.0	33.6	-1.2%	31.4	-7.6%	32.2	-5.3%
Total drive time (s)	783384	773201	-1.3%	745166	-4.9%	748439	-4.5%
Drive time with pas. (s)	384989	383787	-0.3%	382867	-0.6%	384281	-0.2%
Drive time without pas. (s)	398394	389414	-2.3%	362298	-9.1%	364158	-8.6%

As can be seen in Table 3, it is the reduction in the number of vehicles and the drive time without passengers that makes the major contribution to the increased efficiency. The Multi vehicle segments method reduces the number of vehicles needed by 7.6% and the total vehicle drive time by 4.9%. The reduction in drive time is almost exclusively due to the fact that the drive time without passengers is reduced, while the drive time with passengers in the vehicles remains almost unchanged.

5.2 Results for external Vehicles

In Figure 5 we show the results for the scenario with external vehicles, and in Table 4 we describe how the evaluation criteria have changed with the various methods. The presented values in Table 4 are averages of the five simulations of each method.

Table 4: Values of the evaluation criteria for external vehicles

	Single random request	Multi random requests		Multi vehicle segments		Early segments	
Objective function value	75797	73955	-2.4%	72833	-3.9%	72596	-4.2%
No of vehicles needed	32.8	31.8	-3.0%	31.4	-4.3%	32.2	-1.8%
Total drive time (s)	723062	699529	-3.3%	683887	-5.4%	681988	-5.7%
Drive time with pas. (s)	385712	379664	-1.6%	377628	-2.1%	375689	-2.6%
Drive time without pas. (s)	337350	319865	-5.2%	306259	-9.2%	306298	-9.2%

As can be seen in Table 4, the Multi random requests method reduces the average objective function value by 2.4%, which is about the same as for the case with internal vehicles. The Multi vehicle segments and the Early segments methods lower the average objective function value by 3.9% and 4.2% respectively, which is not as much as for the case with internal vehicles. This is due to the fact that it is no longer enough to reduce the number of vehicles being used. As shown in Table 4,

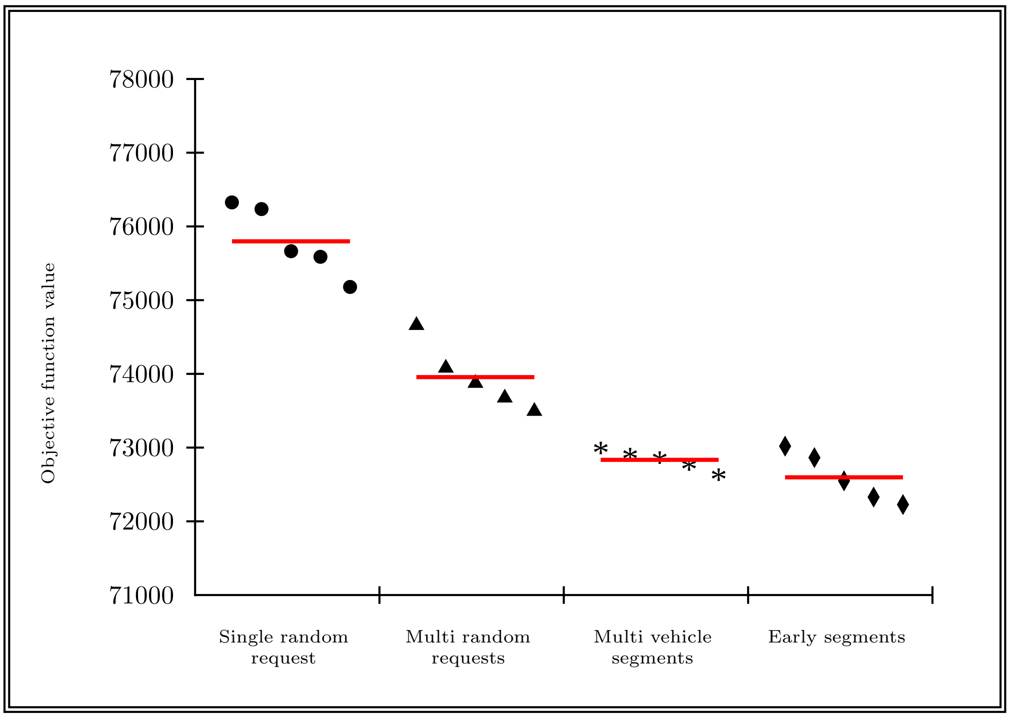


Figure 5: Objective function values for simulations of external vehicles. Each mark is the objective function value of one simulation and the horizontal lines are the average values of the simulations made with each method

the Multi vehicles segments method reduces the average number of vehicles needed by 4.3% and the Early segments method reduces it by only 1.8%. Still, the Early segments method gives a slightly better average of the objective function value. In the scenario with external vehicles, it is mainly through the reduction of drive time without passengers that the increased efficiency arises.

5.3 Successful Reassignments

Another way of measuring how the methods perform is to study the number of successful reassignments, i.e. how many times the reassignment procedure succeeds in finding better solutions. In combination with the change in objective function value, this also gives information about whether the method improves the objective function value based on many small or a few large steps. For a planning system based on one single reassignment procedure this is not essential information, since the final objective function value is what is important, but for future developments based on combinations of multiple reassignment methods, this information can be interesting.

Table 5 presents the average numbers of successful reassignments, based on the five repetitions of each method for the simulations described in Section 5.1–5.2. The values given in Table 5 can be related to those about 3 million reassignment attempts that are made in each simulation.

Table 5: Number of successful reassignments

	Single Random request	Multi random requests	Multi vehicle segments	Early segments
Internal vehicles	297	561	1545	1429
External vehicles	150	30383	29444	13096

For the internal vehicles scenario, the number of successful reassignments is more than five times higher with Multi vehicle segments than it is with the original method. It can also be seen that the increase in the number of successful reassignments corresponds to the improvement in objective function values, (compare to Figure 4 or Table 3).

For the case with external vehicles however, the table shows that the method Early segments renders about half as many successful insertions as the Multi vehicle segments. Still Early segments gives a slightly (0.3 %) better objective function value.

Figure 6 illustrates how the number of reassignments, and the total improvements in objective function value, varies during the day, hour by hour. The figure is based on one of the simulations from the case with external vehicles where the Early segments method is used. The continuous line, for which the values are read on the left axis, describes the total change in the objective function value that the reassignments have generated each hour. A reassignment is here considered to belong to the time period (hour) within which the first chosen request (the randomly chosen one) has its PPT. The dashed line, for which the values are read on the right axis, illustrates how many reassignments that have given significant changes to the objective function value.

As can be seen from the figure, both the number of significant reassignments and the total change in objective function value resemble the distribution of requests presented in Figure 3.

The ratio of the change in objective function value and the number of significant reassignments (i.e. the average improvement per reassignment) is largest just before the peak-hours (compare to Figure 3). This is the time when the highest number of requests are planned (but not yet served). It is reasonable to believe that the Early segments method can give good solutions for instances with larger number of requests.

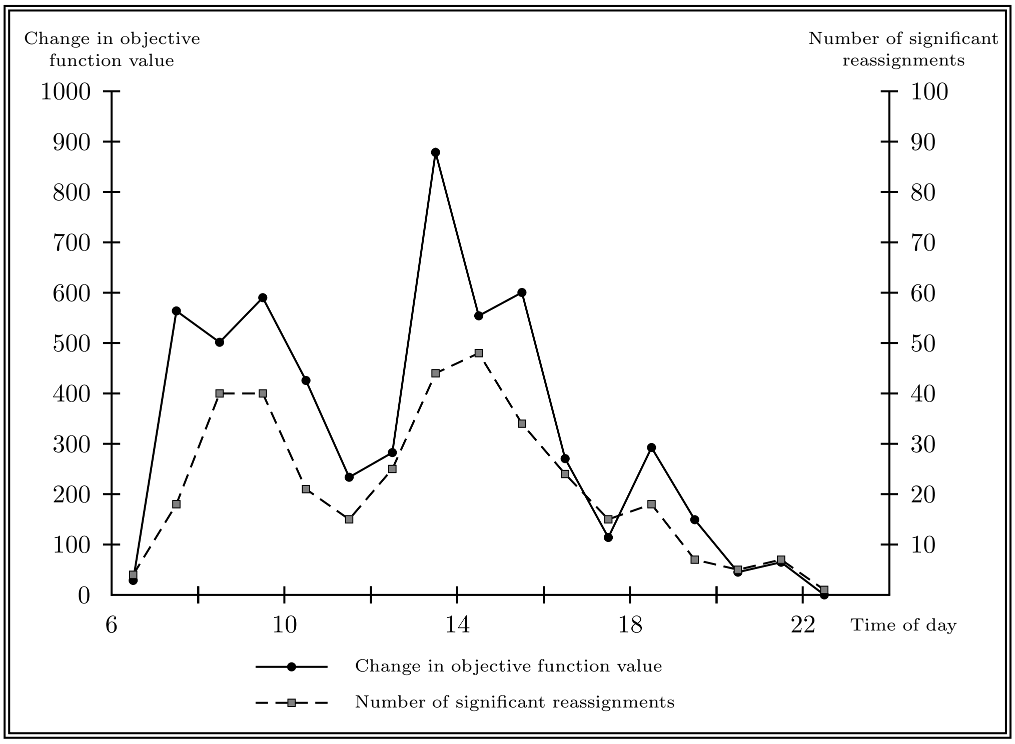


Figure 6: Effects of reassignments over time

6 Conclusions and future Research

We have shown that simple changes to existing planning algorithms can improve the efficiency of practical paratransit planning. Ruin and recreate methods can be successfully used for the reassignment part of planning of a dynamic dial-a-ride service. We have shown that ruin methods based on a simultaneous removal of segments of requests from several vehicles perform best. Compared to the method used today, the objective function value was significantly reduced for a scenario with internal vehicles, as well as for a scenario with external vehicles.

The total drive time is the evaluation criterion that in the best way corresponds to the monetary cost of providing the service. Both for internal vehicles and external vehicles, the best reduction in total drive time found is about 5%. Therefore we believe that changing the reassignment procedure can decrease the operational cost for providing the service by 5%, which in a country-wide implementation would save about € 13,500,000 per year only in Sweden. It is also likely that the algorithm can be further trimmed before being implemented in full-scale.

Future research within this area should be focused on possibilities to combine more advanced metaheuristics with the ruin and recreate methods proposed in this paper. A crucial question is how to keep an operational schedule available at any instance of time. One suggestion is to use the metaheuristic for long term replanning and ruin and recreate methods for short term replanning. In this way, two reassignment procedures are run in parallel, enabling more computation time for the metaheuristic, but without the possibility to consider requests to be served in a near future in the metaheuristic.

References

Cordeau, J., 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54 (3), 573–586.

Cordeau, J. and Laporte, G., 2007. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153 (1), 29–46.

Gendreau, M. and Potvin, J., eds., 2010. *Handbook of Metaheuristics.* 2nd ed. Springer.

Goel, A., 2009. Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43 (1), 17–26.

Häll, C., Högberg, M., and Lundgren, J., 2011. A modeling system for simulation of dial-a-ride services. *Submitted to: Public Transport.*

Paquette, J., Cordeau, J., and Laporte, G., 2009. Quality of service in dial-a-ride operations. *Computers and Industrial Engineering*, 56 (4), 1721–1734.

Parragh, S., Doerner, K., and Hartl, R., 2008. A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58 (2), 81–117.

Pisinger, D. and Ropke, S., 2007. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34 (8), 2403–2435.

Schneider, J. and Kirkpatrick, S., 2006. *Stochastic Optimization.* Berlin Heidelberg: Springer-Verlag.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G., 2000. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159 (2), 139–171.

SIKA 2009a. Kollektivtrafik och samhllsbetalda resor 2008. SIKA statistik 2009:29, SIKA, Sweden.

SIKA 2009b. Special transport service and inter-municipal transport service 2008. SIKA statistik 2009:16, SIKA, Sweden.

Voudouris, C., Tsang, E., and Alsheddy, A., 2010. Guided local search. *In*: M. Gendreau and J. Potvin, eds. *Handbook of Metaheuristics*, 2nd ed. Springer.

Wen, M., Cordeau, J., Laporte, G., and Larsen, J., 2010. The dynamic multi-period vehicle routing problem. *Computers and Operations Research*, 37 (9), 1615–1623.

Yepes, V. and Medina, J., 2006. Economic heuristic optimization for heterogeneous fleet VRPHESTW. *Journal of Transportation Engineering*, 132 (4), 303–311.