Fast heuristics for a dynamic paratransit problem

Cremers, M.L.A.G.; Klein Haneveld, W.K.; van der Vlerk, M.H.

Fast heuristics for a dynamic paratransit problem

Maria L.A.G. Cremers* Willem K. Klein Haneveld
Maarten H. van der Vlerk

Department of Operations
University of Groningen
P.O. Box 800, 9700 AV Groningen
The Netherlands

January 9, 2009

Abstract

In a previous paper we developed a non-standard two-stage recourse model for the dynamic day-ahead paratransit planning problem. Two heuristics, which are frequently applied in the recourse model, contain many details which leads to large CPU times to solve instances of relatively small size. In this paper we simplify both heuristics to decrease CPU time considerably while maintaining the quality of the obtained solutions as much as possible. Numerical experiments on (semi-)realistic instances, inspired by practice, show that our recourse model with fast heuristics provides acceptable solutions within reasonable time.

1 Introduction

The Dynamic Day-ahead Paratransit Planning (DDaPP) problem concerns the planning of transportation requests of elderly and disabled people for the next day. Already today, one day before the day of operation, many requests are known at the transportation company. Moreover, it is already known that there are not enough own vehicles to serve all requests, especially during peak hours. In this phase the decision is at stake which of these early requests to serve with own vehicles and which ones to assign to subcontractors, under uncertainty of the late requests arriving tomorrow, during the day of operation. Subcontracting day-ahead can be profitable since it is less expensive than during the day of operation. Besides the uncertainty with respect to the late requests, clustering (combining) of requests is an important aspect of the DDaPP problem. It ensures that less vehicles are necessary to serve all requests. Since subcontracting

*Email: m.l.a.g.cremers@rug.nl

is more expensive than using an own vehicle, clustering actually ensures that less requests will need to be subcontracted.

In a previous paper [1] we studied the DDaPP problem and developed a non-standard two-stage recourse model in which both important problem aspects are included in detail. In the first stage, modeling today, all early requests are clustered and subsequently assigned to own vehicles or those of subcontractors. The second-stage problem, modeling tomorrow, is a dynamic problem in which the late requests arrive one-by-one. A discrete event simulation is applied in which two heuristics are repeatedly used, each generating a type of decision: clustering of requests and assignment to vehicles. Both heuristics contain many details which results in rather large CPU times for instances of relatively small size.

In this paper, our goal is to apply our model to (semi-)realistic instances of large size. However, using the recourse model CPU times to solve one large instance are up to 90 hours. Therefore, simplifications are needed in the time-consuming clustering and assignment heuristics, without deteriorating the quality of the solutions too much. As we will see, a small adjustment in the assignment heuristic leads to a large decrease in the number of calls, and hence in CPU time. For the clustering heuristic various simplifying, fast alternatives will be developed since it is less obvious how to decrease CPU time while maintaining the quality of the obtained solutions as much as possible. To choose the alternative which is the best compromise between CPU time and quality, experiments have been performed with small instances. For these instances the model with accurate, elaborate heuristics can be used to find solutions of high quality and serve as a benchmark for the quality of the various alternatives.

In Section 2 we briefly describe the DDaPP problem, recourse model and original heuristics to generate decisions in the discrete event simulation. Section 3 contains a description of the various alternative heuristics and the results of the experiments to determine which alternative is the best compromise between quality of the obtained solutions and CPU time. In Section 4 we present the various practical settings and the results of our experiments. A summary and conclusions can be found in Section 5.

2 Summary preceding paper

A description of the DDaPP problem can be found in Section 2.1. Section 2.2 contains the non-standard two-stage recourse model. In the second stage, a discrete event simulation is applied in which two heuristics are used to generate the clustering and assignment decisions. Both heuristics are described in Section 2.3.

For the motivations and detailed descriptions we refer the interested reader to [1].

2.1 Problem description

In the DDaPP problem today a (pre)planning decision is made for tomorrow, the day of operation. Two sets of requests are considered: early and late. Early requests are known today and late requests are gradually revealed during the day of operation, but at least a certain time before they start. All requests need to be served, either by own vehicles or by those of subcontractors.

There are two kinds of passengers: ambulatory and those in a wheelchair, and three types of vehicles: cars, vans, and taxis. Cars and vans belong to the own vehicle fleet, while taxis are vehicles of subcontractors. Ambulatory passengers can be served by all vehicle types, and wheelchair passengers only by vans and taxis which have convertible seats. For cars and vans the available number of vehicles is given, which can vary during the day. Furthermore, the capacity of all vehicle types is given, as well as the costs per distance unit.

For each request the number and the kind of passengers is given, as well as the pickup and delivery location (origin and destination), and a desired pickup or delivery time from which time windows for pickup and delivery are derived.

Requests can be clustered and served with one vehicle in order to reduce the number of vehicles required to serve all requests. Since the number of own vehicles is fixed and subcontracting is more expensive than using an own vehicle, clustering ensures that less requests will need to be assigned to subcontractors. For requests to be combined, the vehicle capacity, time windows, and maximum excess ride time (due to the detour) need to be respected. The excess ride time is defined as the actual minus the direct ride time. Clustered requests are called a *route*.

We neglect the time to drive empty from the destination of a request to the origin of the next one. This assumption reduces the complexity of the problem considerably and is acceptable in our view since we are concentrating on a day-ahead problem which does not require the actual vehicle routes to be made.

The objective of the DDaPP problem is to minimize the expected costs of serving all requests, both early and late. For the own vehicles only the operational costs are taken into account, which are proportional to the distance driven. For subcontracting, besides the operational costs, a fixed fee needs to be paid for each request. Furthermore, subcontracting during the day of operation is more expensive than before, as discounted rates have been agreed with the subcontractors for timely knowing which requests to serve.

2.2 Two-stage recourse model

Figure 1 contains a schematic overview of the two-stage recourse model in which ideas from stochastic programming and online optimization are combined. The first stage, modeling today, consists of two consecutive optimization problems. First, all early requests are clustered into routes. For this purpose, a clustering heuristic, based on certain problem characteristics, is developed and described in [2]. After clustering, the obtained routes are assigned to own vehicles and taxis. Subcontracting is a permanent decision, and hence, subcontracted routes

do not appear in the second-stage problem. In contrast, the assignment of routes to own vehicles is tentative and can be reconsidered during the day of operation.

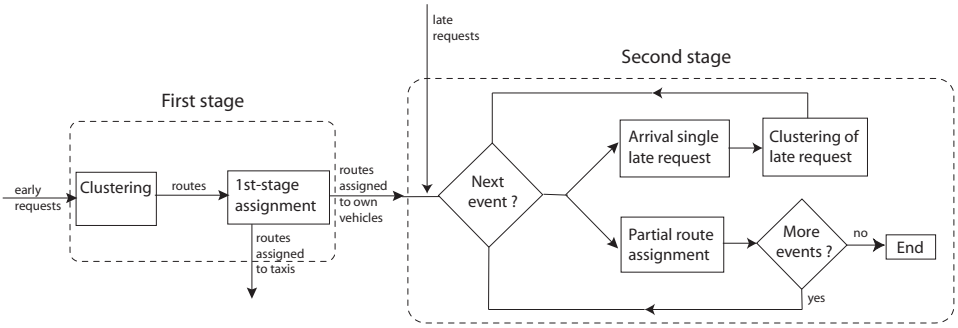

Figure 1: Schematic overview of the two-stage recourse model.

The objective of the DDaPP problem is to find the first-stage assignment with minimal total (expected) costs. The costs of an assignment consists of two parts: the costs of assigning routes to taxis today and the expected costs of assigning the remaining routes and all late requests to vehicles tomorrow. Thus, in both stages only the costs of the permanent assignments are taken into account.

In the second stage, modeling tomorrow, for a given first-stage assignment and observing a sequence of realized late requests a discrete event simulation is carried out. In this simulation, two types of decisions are made. First, every time a late request arrives, immediately an attempt is made to cluster the late request with a tentatively assigned route. Second, routes are given a permanent assignment. This occurs as late as possible so that more late requests may arrive and better decisions can be made. In Section 2.3 the heuristics generating each type of decision are described.

2.3 Heuristics of the discrete event simulation

For both the clustering and assignment heuristics the CPU time per call is small. However, both heuristics are called many times which gives rise to large total CPU times: the clustering heuristic is called for every late request of every sequence of realizations and the assignment heuristic for every time a route needs to be given a permanent assignment. Hence, adjustments to be made have to reduce CPU time per call (even further) or the number of calls.

The **clustering heuristic** attempts to combine one late request with a tentatively assigned route. Routes with a permanent assignment are disregarded since they are not allowed to be changed anymore.

The set of routes is divided into subgroups based on common locations since in paratransit transport some locations like hospitals and elderly homes occur frequently. The late request is attempted to be combined with the routes of

only one subgroup. For simplicity reasons the other subgroups are ignored. If no combination satisfying the clustering conditions, specified in Section 2.1, can be found, the late request will not be clustered, but forms a new route instead.

In the **assignment heuristic** a number of routes is permanently assigned to own vehicles or taxis. Routes which already have a permanent assignment are considered implicitly since they diminish the number of available own vehicles. Moreover, also tentatively assigned routes starting within a certain look-ahead period, but not yet receiving a permanent assignment, are considered.

All routes under consideration are assigned one-by-one to own vehicles or – whenever this is not possible – to taxis. To determine whether a route can be assigned to an own vehicle, time is discretized in periods of equal length and a relaxed feasibility test is used: for each time period the route needs to be served, it is checked whether an own vehicle (of certain type) is still available. Thus, the assignment decision neglects scheduling aspects, but is suitable for the preplanning decision we want to make.

3 Alternatives for the heuristics

CPU time to solve the model with the elaborate clustering and assignment heuristics is rather large. For small instances with in total 55 – 75 requests CPU time is 25 – 70 minutes and for large instances with approximately 700 requests it can be up to 90 hours[1]. The cause of these large CPU times is the numerous calls of the clustering and assignment heuristics in the discrete event simulation. Hence, to decrease CPU time, simplifying adjustments to one or both heuristics have to be made, which decrease the number of calls, CPU time per call, or both. The challenge is to find adjustments which not only reduce CPU time considerably, but also maintain the quality of the obtained solutions on an acceptable level.

In the elaborate version of the assignment heuristic it is obvious how to realize a large decrease in the number of calls: just reduce the look-ahead period to zero. Initially, we expected to improve the assignment decision by not only considering the routes that need to be assigned permanently, but also routes starting during the look-ahead period. However, experiments showed that the improvement is only minor. If the look-ahead period is equal to zero, for every observed sequence of realizations of late requests only one call of the heuristic is needed instead of numerous calls. In this one call all routes are assigned permanently, while in the other case every time a subset of routes is assigned, some tentatively, others permanently. The one call can be achieved by ordering the routes in increasing start time.

For the clustering heuristic it is less obvious how to decrease CPU time without risking solutions of poor quality. Hence, various alternatives have been developed. Experiments with small problem instances are used to select the alternative which is the best compromise between CPU time and quality of the

[1] On a 2.33 Ghz Intel Core 2 Duo.

obtained solutions. The latter is determined by comparison to the (supposedly) high quality solutions of the model with the accurate, elaborate heuristics.

In Section 3.1 we describe the various alternatives for the clustering heuristic. In fact, two categories are considered: alternatives which reduce the number of calls and alternatives which decrease CPU time per call. To select one alternative, to be used in experiments with (semi-)realistic instances, experiments with small problem instances have been performed. The data of these instances can be found in Section 3.2, while Section 3.3 contains the parameter setting of the assignment heuristic and the genetic algorithm which is used to solve the entire recourse model. In Section 3.4 we present the results of the experiments with the small instances and select the alternative which appears to be the best compromise between CPU time and quality.

3.1 Description of alternatives for the clustering heuristic

We consider two categories of alternatives for the clustering heuristic. In the first category the late requests are not combined at all and hence the number of calls of the heuristic is reduced to zero. By not clustering the late requests, however, the number of routes which need to be outsourced increases. In an attempt to compensate for this disadvantage, virtual vehicles are added to the own fleet as used in the assignment heuristic. The number is chosen carefully in order to subcontract approximately the same number of routes. The cost structure remains the same.

In the second category, CPU time per call of the heuristic is decreased by simplifying the clustering decision. Instead of applying the clustering heuristic described in Section 2.3, we simply assume that the proportion of combinations is the same in both stages. To this end, we estimate the clustering probability based on information of the first-stage clustering, in several ways. Since the early requests are still combined with the accurate, elaborate heuristic and the only difference in the characteristics of both types of requests is the arrival time, we expect the information from the first-stage combinations to be valuable when clustering the late requests. Obviously, this approach is rather crude, and invalid combinations might result, but we expect that a reasonable estimate of the proportion of combinations to be made is more important than knowing the details of the actual combinations.

Below, the alternatives of both categories are explained in detail.

The **first category** comprises alternatives in which the late requests are not combined and virtual vehicles are used for compensation. The alternatives in this category differ in how the number of virtual vehicles is determined.

1a. No virtual vehicles. We do not expect this alternative to perform very well since on the day of operation not enough own vehicles are available to serve the additional routes caused by not clustering the late requests. Consequently, too many early requests will be assigned to taxis as subcontracting today is less expensive than tomorrow.

1b. The number of virtual vehicles is fixed. Several numbers have been tested and only the best result, obtained with 2 virtual cars and no extra vans, will be presented. If this alternative will be chosen for the experiments with the realistic instances, it might be hard to determine the number of virtual own vehicles since we did not find a relation between the required number of virtual cars and vans and the parameters of the problem.

1c. The number of virtual vehicles varies during the day of operation and it also depends on the observed sequence of realized late requests. Ideally, the number of virtual cars and vans should be equal to the shortage originating from not clustering the late requests. This way the late requests which would have been combined can be served with virtual vehicles and the first-stage decision does not need to be changed.

To estimate the number of virtual cars and vans for every realization and time period, first the probability of combining a late request is estimated. Then it is multiplied by the number of late car and van requests, respectively. In time period t and realization r, let ac_{tr} and av_{tr} be the additional number of own cars and vans and nc_{tr} and nv_{tr} the number of late car and van requests. Furthermore, let $\hat{p}_2$ be the estimated probability of combining a late request. Then

$$ac_{tr} = \lceil \hat{p_2} \cdot nc_{tr} \rceil ,$$

$$av_{tr} = \lceil \hat{p_2} \cdot nv_{tr} \rceil .$$

In order to estimate the probability of combining a late request, linear regression is used with two independent variables: the observed clustering probability of the early requests and the ratio of the expected number of early requests to the total expected number of requests, fraction of early requests for short. As argued above, we expect the probability of early requests that are combined to be a reasonable estimator for the clustering probability of late requests. Moreover, also the (relative) numbers of requests in both stages are relevant, since combining becomes harder in case there are only few requests. Thus, if there are few early and many late requests the probability of combined early requests might be a too low estimate; if there are many early and few late requests the estimate might be too high. Hence, we have included the fraction of early requests in the regression. This fraction has a negative effect on the probability of combining a late request, whereas the clustering probability of the early requests has a positive effect. Let p_1 be the observed clustering probability of the early requests and f the fraction of early requests. Then, using the data set of Section 4.1, we find

$$\hat{p_2} = 0.294 + 0.965p_1 - 0.173f \qquad \text{with } R^2 = 0.925. \tag{1}$$

In the **second category** of clustering alternatives information from the combined early requests, in the form of probabilities, is used to cluster the late

requests. Using this information is faster than checking the clustering conditions. We expect good results since the early requests are still combined by the accurate, elaborate heuristic and the data of both types of requests are drawn from the same distributions which is also reasonable from a practical viewpoint. In our implementation, three different (groups of) probabilities need to be specified. First, the probability that a late request is clustered. Second, if a combination occurs, probabilities are needed for all tentatively assigned routes to determine with which route the late request is clustered. Third, the data of the chosen route (e.g. time windows and costs) needs to be updated according to probabilities. Since the clustering conditions are not checked, invalid combinations might be formed. Hence, instead of calculating the changes in the data of the chosen route, probabilities are used for this purpose. The alternatives in this category differ in the way the three (groups of) probabilities are determined.

Clustering probability

1. The probability is the same as in alternative 1c, see (1). The probability only depends on information from the first stage. It is the same for all first-stage assignments and all observed sequences of realized late requests.

2. For part of the late requests – in our case the first third of each sequence – the elaborate clustering heuristic is used, from which we calculate a probability to cluster the remainder. Hence, the clustering probability might be different for every first-stage assignment and every sequence of realizations of late requests.

3. The advantage of 2 is the learning effect, but CPU times will probably remain too high. To mitigate this, we use the elaborate clustering heuristic only for some of the sequences of realized late requests, namely the first quarter of sequences. For the remaining ones, the average clustering probability over the first quarter of sequences is calculated and used. Hence, the probability might be different for every first-stage assignment, but only for the first quarter of sequences of realized late requests.

The numerical implementation of alternatives 2 and 3 above (i.e., 1/3 and 1/4, respectively) is obtained by performing in-depth analyses of our earlier experiments with small instances.

Route probabilities

a. Every tentatively assigned route obtains the same probability to be combined with the late request under consideration. Since the number of tentatively assigned routes varies during the discrete event simulation, also the route probabilities vary.

b. The route probability depends on the number of requests in the route: the higher the number of requests, the higher the probability of the route, as routes consisting of many requests are more likely to combine with.

c. In b, the capacity of vehicles is not taken into account. In this alternative, as soon as a route uses full capacity of the vehicle, it is excluded from further combining by setting its route probability to zero. Thus, contrary to b, every resulting route is feasible with respect to vehicle capacity.

Data probabilities

i. The data probabilities, used to update data of a route, are determined by ourselves using our experience of the DDaPP problem derived from numerous experiments.

ii. The data probabilities are calculated from an in-depth analysis of some small instances for which the elaborate clustering heuristic is used to combine the late requests.

In total, there are 18 alternatives in the second category, labeled according to their clustering, route and data probabilities. For example, 2_1bii is the alternative of the 2^{nd} category with clustering probability 1, route probabilities b and data probabilities ii.

3.2 Data of the small instances

All 48 data instances are equal to those used in [1]. They are randomly generated, contain 50 early requests and the number of late requests varies between 5 and 25.

Pickup time	7 – 10 AM
Delivery time	until noon
Capacity car	3
Capacity van	6
Number of passengers per request	1
Probability wheelchair passenger	0.125
Operational costs car	0.8 per grid unit per route
Operational costs van	1 per grid unit per route
Operational costs taxi	1 per grid unit per request
Fixed fee subcontracting	3 per request
Surcharge subcontracting tomorrow	20%

Table 1: Part of the data of the small instances.

Table 1 contains part of the data of the instances. Furthermore, the capacity of the own vehicles does not depend on the customer type. Until 10 AM, 10 cars are available, thereafter 4, and the entire day 3 vans are available. All vehicles have a speed of 8 grid units per hour. The locations lie on a 7×7-grid and have an equal probability to be chosen, except for two special locations (e.g. hospitals) which have a higher probability: 0.05, 0.1, or 0.3. Moreover, requests need to be subcontracted at least one hour in advance.

The 48 data instances can be divided into four classes of 12 instances each. The classes differ in the number of late requests and the probability on a special location. Table 2 contains the specifications and names of the four classes.

	FewLate	ManyLate	LowClust	HighClust
Number of late requests	5 – 10	20 – 25	8 – 14	8 – 14
Prob. on special location	0.1	0.1	0.05	0.3

Table 2: Specifications of the four small instance classes.

3.3 Parameter setting of the heuristics

In the assignment heuristic of the discrete event simulation, the time periods have a length of 15 minutes and the look-ahead period is equal to zero.

To find a good first-stage assignment of routes to vehicles a genetic algorithm is used. For a description of the algorithm, we refer the reader (again) to [1]. Here, we will only discuss briefly the parameters of the genetic algorithm, with which the algorithm consistently finds solutions of similar quality, as tests have indicated.

The population size is set equal to 30. The initial members are constructed by reserving some capacity today to be able to serve the late requests arriving tomorrow. Tests have indicated that the algorithm converges faster to a good solution compared to starting with a random initial population.

The genetic algorithm stops after the generation of 500 feasible, non-duplicate children, since beyond this number the best solution found appears to improve only marginally.

To estimate the total costs of a solution, a sample of 200 sequences of realizations for the late requests is used. Then, the half-length of the 95% confidence interval for the estimated costs of the best solution found is approximately 1% (of the estimated costs), which we think is reasonable.

3.4 Choice of alternative

We want to compare the various alternatives for the clustering heuristic to each other and to the elaborate heuristic. To be able to make a fair comparison, for each alternative the cheapest solution found by the genetic algorithm is *evaluated* in the model with the elaborate heuristics, from now on referred to as the *elaborate model*. These 'true' estimated costs are compared to those of the cheapest solution of the elaborate model by calculating the relative difference with respect to the latter. To determine the quality of each alternative, we use three indicators:

- the average relative difference in estimated costs over all instances,
- the average relative difference in estimated costs per instance class, and

- the minimum and maximum relative difference in estimated costs over all instances.

Table 3 shows for each alternative the three quality indicators and CPU time in seconds per instance. The order of the instance classes is FewLate, ManyLate, LowClust, and HighClust. CPU time to solve the elaborate model is 1555 – 4138 seconds per instance, all on a 2.33 Ghz Intel Core 2 Duo.

Alt.	Av. diff.	Per instance class	Min. – Max.	CPU time
1a	1.48	0.93 – 2.10 – 0.82 – 2.05	-0.81 – 5.91	80 – 156
1b	0.80	0.77 – 0.95 – 1.02 – 0.49	-0.68 – 3.30	83 – 152
1c	0.55	0.46 – 0.94 – 0.33 – 0.49	-0.76 – 3.37	92 – 157
2_1ai	0.87	0.50 – 1.19 – 0.49 – 1.28	-0.80 – 3.92	197 – 525
2_1aii	0.76	0.32 – 0.95 – 0.52 – 1.24	-0.76 – 2.97	207 – 521
2_1bi	0.70	0.40 – 0.92 – 0.37 – 1.13	-0.42 – 4.17	209 – 523
2_1bii	0.91	0.66 – 0.91 – 0.53 – 1.55	-0.35 – 3.94	214 – 522
2_1ci	0.62	0.46 – 0.87 – 0.37 – 0.80	-0.55 – 3.82	211 – 528
2_1cii	0.79	0.49 – 0.95 – 0.53 – 1.18	-0.69 – 2.97	217 – 539
2_2ai	0.52	0.28 – 0.77 – 0.30 – 0.74	-0.23 – 1.87	722 – 2145
2_2aii	0.61	0.37 – 0.79 – 0.21 – 1.06	-0.58 – 3.68	731 – 2153
2_2bi	0.70	0.36 – 1.21 – 0.37 – 0.87	-0.87 – 2.60	725 – 2159
2_2bii	0.61	0.27 – 0.94 – 0.45 – 0.80	-0.41 – 3.03	726 – 2162
2_2ci	0.66	0.21 – 0.99 – 0.46 – 0.96	-0.58 – 2.72	741 – 2205
2_2cii	0.67	0.38 – 1.27 – 0.02 – 1.01	-0.78 – 2.97	737 – 2207
2_3ai	0.67	0.61 – 1.07 – 0.33 – 0.68	-0.31 – 2.52	355 – 936
2_3aii	0.83	0.73 – 1.08 – 0.48 – 1.04	-0.45 – 2.35	353 – 943
2_3bi	0.76	0.35 – 1.17 – 0.45 – 1.08	-0.68 – 2.64	360 – 958
2_3bii	0.73	0.31 – 1.26 – 0.47 – 0.88	-1.00 – 2.62	362 – 968
2_3ci	0.69	0.52 – 1.23 – 0.23 – 0.77	-0.87 – 2.58	371 – 977
2_3cii	0.65	0.38 – 0.95 – 0.49 – 0.80	-0.47 – 2.27	374 – 973

Table 3: The quality indicators and CPU time of the various alternatives for the clustering heuristic.

We need to choose one alternative which is the best compromise between quality of the obtained solutions and CPU time. Since the quality indicators, especially the average relative difference over all instances and per instance class, do not differ much and CPU time does, the focus by choosing an alternative is on the latter. That is, an alternative from the first category (using virtual vehicles) will be chosen. Since alternative 1c has the best quality indicators in this category, we select alternative 1c to apply to (semi-)realistic instances.

A more detailed discussion of the results in Table 3 supporting our choice follows.

First of all, all average relative differences over all instances are less than 2%, twice the half-length of the 95% confidence interval for the estimated costs of a solution. Hence, on average all alternatives provide solutions which are not

significantly more expensive. Moreover, all alternatives are on average much better than other fast models like the myopic model[2]. This naive model, which can be solved in seconds, does not reserve capacity today for the late requests arriving tomorrow. In Section 4.3 the myopic model is discussed in more detail. We conclude that the quality of almost all alternatives is reasonably good.

Furthermore, there is not one alternative which is clearly better than the others on all three quality indicators and CPU time. The variation in the average relative difference, both over all instances and per instance class, is in general low. Differences in CPU time are considerable, but for all alternatives CPU time is less than for the elaborate model, which is 1555 – 4138 seconds per instance. Notice that the minimum relative difference is always negative, indicating that each alternative finds, for at least one instance, a solution with lower estimated costs, compared to the solution of the elaborate model. This is caused by a small number of instances for which the solution of the elaborate model is probably not very good.

In the first category of alternatives the most sophisticated alternative (1c), in which the number of additional own vehicles varies during the day of operation and per sequence of realizations, is clearly the best. The average relative difference over all instances and per instance class is smallest for alternative 1c, whereas the minimum and maximum relative difference and CPU time are almost equal to the best of the alternatives in this category. As we expected, the quality of alternative 1a, in which no additional own vehicles are available, is low compared to all other alternatives. In this alternative, the results show that too many early requests are subcontracted today, compared to the solution of the elaborate model. Subsequently, during the day of operation own vehicles remain unused while a day earlier requests have been subcontracted, which is more expensive than using an own vehicle.

In the second category of alternatives differences in the three indicators are in general quite small, while those in CPU time are large. The variation in CPU time is caused by the elaborate clustering heuristic, which is either not applied, for approximately one in twelve late requests, or for a third of all late requests. We expected the alternatives to provide solutions of higher quality if more late requests are combined by the elaborate clustering heuristic. This is barely visible in the results since the variation in the average relative difference is only small. However, we conclude that the use of a more sophisticated clustering probability is not worth the increase in CPU time. The alternatives with the elementary clustering probability, which is the same for all solutions and all observed sequences of realized late requests, perform slightly worse than the other alternatives in this category, but with (much) lower CPU time.

[2]The average relative difference in 'true' estimated costs over all instances between the solution of the myopic model and the cheapest solution of the elaborate model is 2.15%.

4 Numerical experiments

The model with the simplified, fast clustering and assignment heuristics is now applied to (semi-)realistic instances. In Section 4.1 the data of the realistic instances of large size is presented. Section 4.2 contains the parameter setting of the genetic algorithm since the values of some parameters have been changed to fine-tune the algorithm for the realistic instances. In Section 4.3 we present the results for the realistic instances. Generally speaking, the results of the recourse model with fast heuristics are better than those of the naive, myopic model, but the differences in 'true' estimated costs are rather small. This could be explained by the intuition that for these problem instances, differences between the estimated costs of the myopic model and the elaborate model may be small too, leaving little room for improvement. Of course, we are interested to see how our model with fast heuristics performs on more challenging problem instances. To this end, we constructed semi-realistic instances by including artificial penalty costs in the second stage. Section 4.4 contains the results of the experiments with these instances.

4.1 Data of the realistic instances

All data instances are randomly generated, but inspired by practice. Contrary to the small instances, the total expected number of requests is fixed since in practice the total number of requests is known approximately from history and experience. The total expected number of requests is approximately equal to 700, with either few early and many late requests, as many early as late requests, or many early and few late requests.

The number of requests per hour is non-homogeneously Poisson distributed. Thus, peaks in the number of requests to serve may occur. Table 4 shows the expected number of requests per hour for the three partitions of the total number of requests. Notice that late requests are not allowed to start before 7 AM since requests need to be subcontracted at least one hour in advance. The begin times of the pickup time windows are uniformly distributed per hour.

	a. Many early, few late		b. As many early as late		c. Few early, many late	
Time period	Early	Late	Early	Late	Early	Late
6.00 – 7.00	20	–	15	–	5	–
7.00 – 10.00	70	12	40	40	12	70
10.00 – 15.00	30	6	18	20	5	35
15.00 – 18.00	60	10	35	35	10	60
18.00 – 22.00	10	1	5	5	1	10
Total	600	100	350	345	100	605

Table 4: The expected number of early and late requests per hour.

Table 5 contains part of the data of the realistic instances. All locations lie on a 10×10-grid and have an equal probability to be chosen, except for two

special locations (e.g. hospitals) which have a higher probability of either 0.05, 0.15, or 0.25. The capacity of the own vehicles does not depend on the customer type. All vehicles have a speed of 20 grid units per hour so that the ride time between any origin-destination pair is at most one hour. Furthermore, there is a delay in the availability of the number of own vehicles (not changing at the hour, but 15 minutes later) since the beginning of the pickup time window is used to determine the time windows. The actual pickup time will (probably) be later, implying a delay in the number of vehicles needed, to which we have adapted the availability of own vehicles. Parameter choices are inspired by practice to some extent, and partly chosen to obtain interesting instances (based on trial and error).

Pickup time	6 AM – 10 PM
Delivery time	until midnight
Capacity car	3
Capacity van	6
Number of passengers per request	1 with prob. 0.8
	2 with prob. 0.18
	3 with prob. 0.02
Probability wheelchair passenger	0.125
Operational costs car	0.8 per grid unit per route
Operational costs van	1 per grid unit per route
Operational costs taxi	1 per grid unit per request
Fixed fee subcontracting	3 per request
Surcharge subcontracting tomorrow	50%

Time period	Number of cars	Number of vans
6.00 – 7.15	7	1
7.15 – 10.15	35	4
10.15 – 15.15	15	2
15.15 – 18.15	30	3
18.15 – 22.15	7	1
22.15 – 24.00	1	1

Table 5: Part of the data of the realistic instances.

Since there are three partitions of the total number of requests and three probabilities on a special location, nine classes of instances result. These classes will be indicated by names reflecting the amount of early requests and the probability on a special location, e.g., FewLow for few early (and hence many late) requests and a low probability on a special location; analogously, we define FewMedium, FewHigh, EqualLow, EqualMedium, EqualHigh, ManyLow, ManyMedium, and ManyHigh.

4.2 Parameter setting of the heuristics

In the assignment heuristic of the discrete event simulation, the time periods have a length of 15 minutes.

As for the small instances, a genetic algorithm is used to find a good first-stage assignment of routes to vehicles; only the values of some parameters are adjusted.

The genetic algorithm stops if for 750 feasible, non-duplicate children no improvement in the estimated costs of the cheapest solution has been found, where no improvement is defined as a difference in estimated costs smaller than 0.1%. Tests have shown that the number of generated children varies considerably among the classes, but also that the estimated costs of the cheapest solution found decrease only marginally if more children are generated.

A sample of 100 sequences of realizations for the late requests is drawn to estimate the costs of a solution. With this sample size it will appear that our model provides for most classes and instances significantly better solutions than the myopic model, on the 95%-level. Here, significancy is determined by constructing a confidence interval for the relative improvement in estimated costs.

Besides the sample size also the randomness of the genetic algorithm itself influences the accuracy of the results. Since the genetic algorithm is a random heuristic, equally good solutions might not always be found. To test this effect, we ran the genetic algorithm 10 times on the same instance, for a small number of instances from different classes. We found that the difference in estimated costs was at most 0.63%, which is small enough to conclude that the genetic algorithm consistently finds solutions of similar quality.

4.3 Results realistic instances

To determine the quality of the alternative clustering heuristic for realistic instances, we introduce two solutions: the heuristic solution and the myopic solution. The first is obtained by solving our non-standard two-stage recourse model with fast heuristics with the genetic algorithm. The solution with the lowest estimated costs is called the heuristic solution. The myopic solution is the simple or naive approach which we want to beat. It is obtained by solving the myopic model in which the late requests are ignored. That is, the early routes are greedily assigned to own vehicles, or – whenever this is not possible – to subcontractors. Consequently, the myopic solution does not reserve capacity today to be able to serve the late requests, arriving tomorrow, with own vehicles. In this respect, the myopic solution can also be called the optimistic solution, contrary to the pessimistic solution in which all early requests are subcontracted for fear of future costs.

For all nine classes 12 instances have been generated and solved. For each instance, the estimated costs of the heuristic and myopic solution are calculated by evaluating them in the most realistic model we have developed: the model with the accurate heuristics. Moreover, the relative improvement with respect

to the estimated costs of the myopic solution is determined. For both solutions the number of subcontracted early routes is reported, as well as the total number of early routes. For each class the averages over the 12 instances of the results mentioned above are presented in Table 6. Furthermore, the average CPU time in seconds per instance and the average number of generated children in the genetic algorithm is listed. Both refer to the heuristic solution; the myopic solution is obtained in a couple of seconds.

	Estimated costs				nSub early			
Class	Heuristic	Myopic	Improve (%)	nRoutes	Heuristic	Myopic	time	nChild
FewLow	4682	4738	1.18	84	21	1	702	1522
FewMedium	4157	4197	0.94	79	13	1	619	1385
FewHigh	3775	3798	0.58	74	9	0	523	1126
EqualLow	4318	4506	4.19	313	77	19	1377	3129
EqualMedium	3852	3929	1.94	270	43	14	936	2271
EqualHigh	3589	3618	0.80	228	22	11	762	1759
ManyLow	4073	4165	2.20	513	101	73	1133	2603
ManyMedium	3625	3647	0.58	438	67	56	709	1902
ManyHigh	3380	3382	0.05	361	47	44	456	1216

Table 6: The average results over 12 instances per class, for realistic instances.

For all classes, the estimated costs of the heuristic solution are on average lower than those of the myopic solution. The relative improvement in estimated costs of both solutions is on average between 0.05% and 4.19%. Thus, by solving our recourse model with the modified clustering and assignment heuristics instead of the myopic model, the estimated costs decrease on average up to 4.19%. For class ManyHigh the relative improvement of most instances is not significant at the 95%-level, for the remaining classes almost all improvements are significant. Hence, we recommend to solve all classes, except ManyHigh, with our recourse model instead of the myopic model. In ManyHigh there are only few late requests which can be relatively easily combined. Hence, taking into account the late requests is not very important here, and the quality of the myopic solution is relatively good. CPU time varies between 9 and 23 minutes per instance, which we think is reasonable to solve a day-ahead decision problem for a large instance.

The estimated costs of both solutions, as well as their relative improvement, decrease if the probability on a special location increases. A higher probability implies that more combinations of requests are made. Hence, less routes need to be assigned to vehicles in the second stage, and also the expensive subcontractors are used less. This leads to a decrease in the estimated costs. The relative improvement decreases since it is less important to take the late requests into account when more combinations are made. Furthermore, not only the number of early routes decreases, but also the number of subcontracted early routes, both in the heuristic and myopic solution. In the heuristic solution this is caused by the lower number of routes that need to be assigned to vehicles in the second stage. In the myopic solution less early routes need to be assigned to the

same number of own vehicles implying less early subcontracted routes. Also the number of generated children decreases if the probability on a special location increases. Since the number of early routes decreases, less possible solutions exist allowing the genetic algorithm to find a good solution faster. CPU time is approximately linear in the number of generated children and hence, also CPU time decreases if more combinations can be made.

The estimated costs of the heuristic and myopic solution decrease if more early requests and less late requests arise, since less requests will need to be subcontracted during the day of operation, which is 50% more expensive than before. The relative improvement in estimated costs of both solutions is largest when there are approximately the same number of early and late requests. This is the outcome of two opposite effects. First, the more late requests arrive, the more important it is to take them into account, as only the heuristic solution does, and hence the relative improvement becomes larger. On the other hand, many late requests implies few early requests, and hence also few early routes. Since some routes might not be profitable to subcontract due to e.g. the large number of combinations made, relatively few routes are suitable for subcontracting today, causing the heuristic solution not to perform very well. When there are approximately the same number of early and late requests, there are enough early requests that are profitable to subcontract today and enough late requests to make it worthwhile to take them into account. In short, for such instances choices do make a difference. Consequently, the relative improvement in estimated costs between both solutions is largest. For the same reason, for these classes the largest number of children is generated and CPU time is highest.

To conclude, for realistic instances the gain in estimated costs by solving our recourse model with fast heuristics instead of the myopic model is on average (over all instances of classes) 1.38%, and can be as high as on average 4.19%. For most instances of most classes the relative improvement is significant at the 95%-level. Only when there are few early (and hence many late) requests and a high probability on a special location the myopic solution is competitive. CPU times of up to 23 minutes to solve a day-ahead planning problem for a large instance are acceptable.

4.4 Results semi-realistic instances

Although we have concluded that it is profitable to solve most classes of realistic problem instances with our recourse model with fast heuristics instead of the myopic model, the relative improvements in estimated costs are in general not very large. However, we are interested to see how our model with fast heuristics performs on instances for which we expect that a larger relative improvement can be obtained. To this end, semi-realistic instances have been constructed by introducing penalty costs for subcontracting during the day of operation, instead of the earlier surcharge. By varying the penalty costs the degree of aversion to subcontracting tomorrow can be reflected.

Besides the heuristic and myopic or optimistic solution, we make a comparison to the pessimistic solution in which all early routes are subcontracted today

for fear of future costs for the late requests arriving tomorrow. We expect the pessimistic solution to perform better as the value of the penalty costs becomes higher. Indeed, if subcontracting tomorrow is undesirable due to high costs, it can be profitable to subcontract today all early routes to have more own vehicles available for the late requests arriving tomorrow.

In our experiments, the same data instances are used as in Section 4.3, except for the penalty costs for subcontracting tomorrow, for which we consider four different values: 1.5 (the original one), 15, 150, and 1500. We analyzed all classes, but results are only presented for the classes FewMedium, EqualMedium, and ManyMedium since the number of early and late requests has much more influence on the results than the probability on a special location. Table 7 contains per class (averaged over 12 instances): the average number of subcontracted early routes for the four heuristic solutions with different penalty costs, the myopic solution, and the pessimistic solution. Notice that for the myopic and pessimistic solution the number of subcontracted early routes is independent of the value of the penalty costs.

Class	H_1.5	H_15	H_150	H_1500	Myopic	Pessimistic
FewMedium	13	34	48	56	1	79
EqualMedium	43	76	89	96	14	270
ManyMedium	67	88	99	103	56	438

Table 7: Number of subcontracted early routes for various solutions.

The number of subcontracted early routes is lowest for the myopic solution and highest for the pessimistic solution, as these solutions are obtained by assigning to subcontractors as few early routes as possible and all early routes, respectively. For the heuristic solutions the number of subcontracted early routes increases with the value of the penalty costs: the more expensive subcontracting tomorrow is relative to today, the higher the profits will be of reserving capacity for tomorrow. For extremely high penalty costs, one would expect all early routes to be subcontracted. However, it appears that the chosen number of virtual vehicles to compensate for not clustering the late requests is too large, which results in too few subcontracted early routes. This adverse effect becomes only noticeable in this extreme setting, to the extent that the pessimistic solution outperforms the heuristic solution.

Figure 2 shows for the same three classes the increase in the estimated average costs of the heuristic, myopic, and pessimistic solution, as a function of the penalty costs. In general, for instances with realistic penalty costs, the recourse model with fast heuristics provides the best solution, followed by the myopic model and the pessimistic model. If the aversion to subcontracting tomorrow as measured by the penalty costs increases, the pessimistic model becomes better and better. Contrary to our expectations and as explained above, for the two largest values of the penalty costs the pessimistic model (clearly) outperforms the recourse model, which in turn outperforms the myopic model (by far).

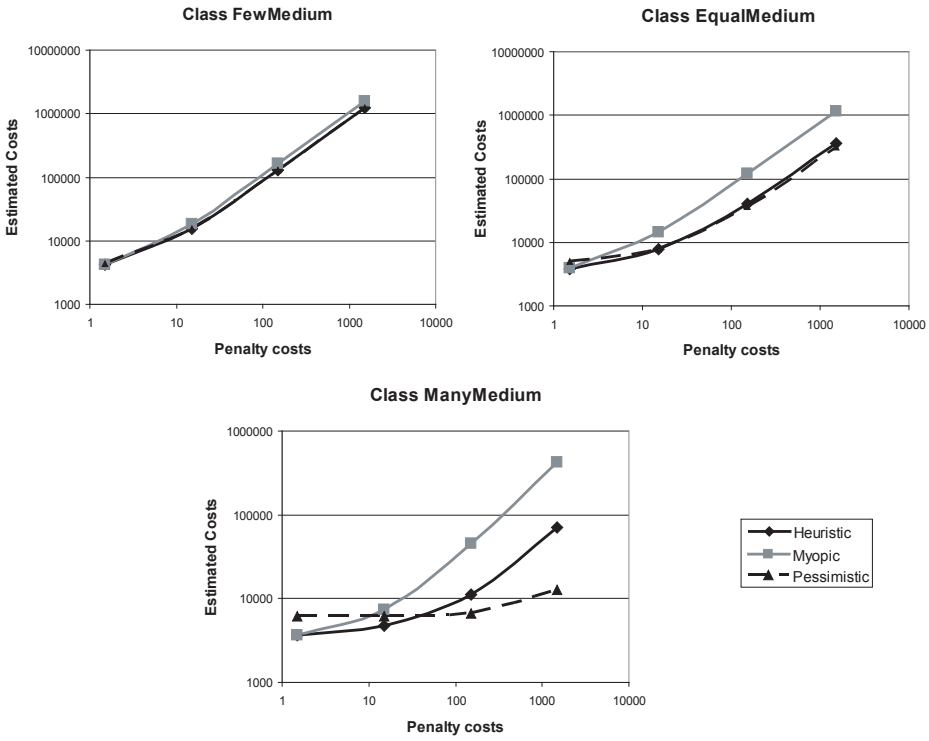

Figure 2: Influence of penalty costs on estimated costs of various solutions.

For the class FewMedium the estimated costs of all three solutions are almost equal, independent of the value of the penalty costs. Since in this class there are only few early and many late requests, subcontracting tomorrow is inevitable and the variation in the number of routes subcontracted during the day of operation is low. Hence, also the variation in the estimated costs is low.

For the class EqualMedium subcontracting tomorrow is also inevitable, but since there are approximately as many early as late requests, the variation in the number of routes subcontracted tomorrow can be larger. Thus, for higher penalty costs, in the heuristic solution much more capacity is reserved compared to the myopic solution. In the pessimistic solution by definition all capacity is reserved for tomorrow, which for high penalty costs yields better results than the myopic solution, comparable to the heuristic solution. This effect is even stronger for the class ManyMedium.

To conclude, for semi-realistic instances, in which subcontracting tomorrow is (very) undesirable, it is very profitable to solve our recourse model with fast heuristics instead of the myopic model. Our second benchmark, the pessimistic model, can be a fast competitor of the recourse model, especially if the value of

the penalty costs is high.

5 Summary and conclusion

In a previous paper we developed a non-standard two-stage recourse model for the dynamic day-ahead paratransit planning problem. In the second stage a discrete event simulation is applied in which two heuristics are used to generate decisions. Both heuristics contain many details which results in large CPU times for instances of relatively small size. In this paper we have simplified both heuristics to decrease CPU time without sacrificing the quality of the results too much and solve instances of (semi-)realistic size within reasonable time.

In the assignment heuristic the reduction of the look-ahead period to zero leads to a large decrease in the number of calls of the heuristic, and hence in CPU time to solve the entire model. To simplify the clustering heuristic without deteriorating the quality too much, various alternatives have been developed which either reduce CPU time per call or the number of calls. The alternative which results in the best compromise between CPU time and quality of the solutions is used to solve (semi-)realistic instances. In this alternative late requests are not clustered; to compensate, virtual vehicles are added in the assignment heuristic so that approximately the same number of routes needs to be subcontracted.

The results are promising. For realistic instances, inspired by practice, our recourse model with fast heuristics almost always outperforms the myopic model in which the late requests are ignored. The improvement in estimated costs of solving our recourse model instead of the myopic model is up to 4.19%. Only for instances with many early and few late requests and a high probability on a special location the quality of the myopic solution, which is obtained in seconds, is relatively good. CPU time of solving the recourse model is 9 – 23 minutes per instance, which we think is acceptable.

For semi-realistic instances, in which the costs of subcontracting during the day of operation is increased, the improvement in estimated costs becomes (very) high. Thus, when subcontracting tomorrow is not desirable, our recourse model outperforms the myopic model by far. If subcontracting is very undesirable the pessimistic model, in which all early routes are subcontracted, is a fast competitor of our recourse model.

References

[1] M.L.A.G. Cremers, W.K. Klein Haneveld, and M.H. van der Vlerk. A dynamic day-ahead paratransit planning problem. To appear in IMA Journal of Management Mathematics. `http://mally.eco.rug.nl/papers/DDaPP.htm`.

[2] M.L.A.G. Cremers, W.K. Klein Haneveld, and M.H. van der Vlerk. A two-stage model for a day-ahead paratransit planning problem. To appear in Mathematical Methods of Operations Research, 2009.