Noname manuscript No.
(will be inserted by the editor)

On the call intake process in service planning

Gerhard Post · Stefan Mijsters

Received: date / Accepted: date

Abstract The call intake process for field service assigns a time window to a client's request for a service. What time window is chosen has influence on the quality of the total planning. The quality is measured by the number of planned requests, the lateness of planned requests and the total travel time.

We constructed challenging datasets, for 25 engineers and an average of 200 tasks per day. We use simulations on these datasets to study the effect of different strategies. Aspects that turn out to be beneficial are: ignore the severity of lateness, give preference to empty shifts, cluster tasks in a shift, and use intermediate optimization.

Keywords Vehicle routing, field service, clustering, call intake, scheduling

1 Introduction

There is a large body of literature on Vehicle Routing Problems (VRPs) with different types of settings and constraints. For a survey on rich VRPs, see for example [Caceres-Cruz et al (2014)]. Here we are interested in the area usually called 'field service', where in addition to the usual set-up the shifts (working hours) of the (field) engineers are fixed beforehand. Having fixed employee shifts might seem not very relevant, but it is: the main objective in VRP usually is to minimize the number of vehicles first, and secondary the total travel distance. In the case we consider here, this 'number of vehicles' is fixed: each shift of an employee is open for service tasks, and there is no gain

Gerhard Post
PCA, Klipperweg 19, 8102 HR Raalte, The Netherlands
and
Department of Applied Mathematics, University of Twente, The Netherlands
E-mail: g.f.post@utwente.nl

Stefan Mijsters
PCA, Klipperweg 19, 8102 HR Raalte, The Netherlands

in leaving a shift empty. On the contrary, spreading the work can be one of the objectives.

More in particular, we consider the situation in which a company (or 'service provider') provides services to its clients upon request. Such a request leads to a task that is executed by an engineer of the company at the address of the client. A task can be a small repair (in case the company is a housing corporation or an installation company), a damage assessment (in case of an insurance company), or some other service. We assume in our datasets that the tasks require between 30 minutes and 60 minutes of service time. Since tasks are executed at the addresses of the clients, the engineer has a daily shift, usually starting at the home address of the engineer, visiting approximately 10 clients, and driving to the end destination, which is either the home or the company address[1].

2 The routing phase

For the moment we assume that somehow all tasks are defined. In particular a task can require a skill, has an (expected) service time, and a time window defines the earliest start time and the latest start time. Usually, time windows originate from the so-called 'block times' that the company applies to all service requests. These block times usually are two to four hours long. If all tasks have time windows within one day, the routing problem essentially consists of daily problems. In particular, tomorrow is relevant; we need to finalize the schedules and inform the field engineers on the routes.

Quite similar to the Maintenance Personnel Scheduling Problem (MPSP) in [Misir et al (2015)], the tasks assigned to a shift of the engineer must obey the following constraints:

- Pre-assigned tasks and other appointments should be scheduled as given.
- The field engineer must be skilled for the task.
- A task should start within its time window.
- The expected travel times between the scheduled tasks should be respected.
- The start address and the finish address of a shift can be different; traveling from the start address to the first task, and traveling from the last task to the finish address can sometimes partly be done in private time.
- If the shift has a break, the duration of the task or travel is extended with the duration of the break. In particular, a task or travel cannot start during the break, but can start before and finish after the break.
- If a task is assigned to a region, the engineer should work in this region. This region can be a geographical region, but also an administrative region. An engineer can be assigned to different regions during the week and even during the day. Multiple regions at the same time are possible.

[1] When the routing aspect of a shift with tasks is being discussed, we might use the word *route*.

– A task can have a pre-assigned engineer, which must be respected, or a preferred engineer. Assigning a preferred engineer takes precedence over minimizing the travel time.

Since the shifts are fixed, the personnel rostering constraints mentioned in [Misir et al (2015)] are not relevant. The main objective is first: to assign as many tasks as possible, second: take preferences into account, and third: minimize the total travel time. Note that minimizing the travel time reduces the direct costs, but also might create space for an extra task in a shift. This is not relevant anymore when we optimize the planning for tomorrow, but for later days it might create space to assign additional tasks.

3 The call intake process

In the previous section we discussed the routing problem to be solved. As said there, the routing problem usually split in daily problems, because of the time windows that are attached to the requests. The assignment of the time windows is done in the phase we call the 'call intake process'. This process for the service planning differs quite of lot from home deliveries, as discussed in [Strauss et al (2021)] and [Visser (2019)]:

– Often home deliveries are for today or tomorrow. In service planning the time scale usually is in weeks.
– In home deliveries the requests do not require skills, and can be assigned to all resources.
– In home deliveries the service durations are short, maybe just 2 minutes. Hence a shift can contain over 100 tasks in an urban region.
– In home deliveries the client orders via internet, while in service planning the majority is done via a planner or the customer contact center.
– In service planning the client usually does not pay for the service, hence giving an incentive to efficient time windows is harder.

Summarizing, we might have more influence on the time window for a request, and, moreover, it might be important to use this to get a favorable time window for a request. We can give the company's planner insight in the differences in travel times for different time windows; if in a certain week there are no convenient time windows for both, the client and the service provider, the planner might switch to the next week, not mentioning (or even not having available) unfavorable time windows.

In this call intake process, there is a balance between using low travel times, and filling the shifts of service engineers for the upcoming days. How eager should we be filling these shifts? If a shift for tomorrow can accommodate a task, shouldn't we simply take it, to avoid that a part of the capacity of the shift is left unused?

Note that the call intake process can be viewed as the construction phase of a routing problem. Methods that improve this phase can be helpful in the call intake process. On the other hand, we know that the result of the construction

phase is not directly related to the result after optimization. Hence, we want to study these differences as well.

4 Explanatory example

4.1 Set-up

We discuss a small 1-dimension example, to explain the way the call intake process works, and to show the effect of clustering, which we explain in detail in Section 7. The example can be analyzed completely.

The set-up is the following:

- There is one engineer at position 0. All incoming tasks can be executed by this engineer.
- Every day two tasks appear, with equal chance for the positions -1 or 1. The deadline is three days (tomorrow and the two days after that).
- The engineer can handle two tasks per day, which are either both at position 1 (travel time is 2 units), or both or position -1 (travel time again 2 units) or one at position -1, and 1 at position 1 (travel time 4 units).
- The time windows coincide with the days.
- The planning is two tasks behind. That means the following. Today we do the planning for tomorrow and the two days after tomorrow. However, there are already two tasks planned for tomorrow and the day after. Since postponing tasks has no benefits, we have 5 (reasonable) 'states':
  1. State $(1,1) \odot (.,.)$: tomorrow the engineer has two tasks at position 1, nothing planned yet for the day after.
  2. State $(-1,-1) \odot (.,.)$: tomorrow the engineer has two tasks at position -1, nothing planned yet for the day after.
  3. State $(1,-1) \odot (.,.)$: tomorrow the engineer has one task at position 1 and one at position -1, nothing planned yet for the day after.
  4. State $(1,.) \odot (-1,.)$: tomorrow the engineer has a task at position 1, and the day after at position -1.
  5. State $(-1,.) \odot (1,.)$: tomorrow the engineer has a task at position -1, and the day after at position 1.

4.2 Strategy: first possible day

Starting from the 5 states above, we apply the strategy 'first possible day'. That means that any task that appears is planned at the first possible day i.e. the first day with at most one planned task. Note that it not allowed to look ahead! From a meta point of view, we know that 2 tasks per day will appear (since that is our set-up), but for the simulation we do not know this. For example, if we are in State 4, and a task at position -1 appears as first task, we plan it for tomorrow, even though the next task might be at position 1.

With this strategy we have the following transition matrix between the states; matrix element (i, j) is the probability that State i moves to State j on the next day.

$$\begin{pmatrix} 0.25 & 0.25 & 0.5 & 0 & 0 \\ 0.25 & 0.25 & 0.5 & 0 & 0 \\ 0.25 & 0.25 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \end{pmatrix}.$$

We see that in the steady state the States 4 and 5 disappear, because they do not satisfy the 'first possible day' strategy. Moreover, we can calculate that in the steady state, the States 1 and 2 have a probability $\frac{1}{4}$, and State 3 has probability $\frac{1}{2}$. From this we obtain the expected daily travel time:

$$\frac{1}{4} * 2 + \frac{1}{4} * 2 + \frac{1}{2} * 4 = 3.$$

4.3 Strategy: cluster

In the 'first possible day' strategy, we do not use the freedom to postpone a task. Doing this is an example of clustering. Our strategy for the next appearing task at position x is the following:

- If tomorrow has only one task planned, we plan it tomorrow.
- If tomorrow is full, plan it on a day we already visit position x.
- If after tomorrow we don't visit x yet, plan it on the first empty day.

Again we can calculate the transition matrix, which is now

$$\begin{pmatrix} 0.25 & 0.25 & 0 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0 & 0.25 & 0.25 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0.5 & 0 \end{pmatrix}.$$

In this case, State 3 disappears from the steady state, and the other states all have probability $\frac{1}{4}$. Analyzing the expected daily travel time for this asymptotic situation yields:

$$\frac{1}{4} * 2 + \frac{1}{4} * 2 + (\frac{1}{8} * 2 + \frac{1}{8} * 4) + (\frac{1}{8} * 2 + \frac{1}{8} * 4) = 2.5$$

Hence clustering yields a saving of 16.7% in travel time, while still planning all tasks within the deadline.

5 Simulation set-up

The example above is exceptional in the sense that it can be completely analyzed. It the situation we will describe below, this is impossible due to the huge number of tasks and the complicated 2-dimensional geometry, which is common in a real life. To study the effect of different strategies we will run simulations.

To have a realistic situation, we created five datasets for a company with 25 engineers. We generated data for 270 (working) days, with an average of 200 requests per day. These requests are turned into tasks during the call intake process. This means that a time window is assigned to it, and temporarily an engineer to assure feasibility of the schedule. The time and engineer of the assignment can be changed, as long as the time window is respected: the task should start in its associated time window. We assume an online process, by which we mean that the requests have to be processed in the order they appear, without knowledge of the requests later in the list. In more detail, the data is created in the following way.

5.1 Locations and travel times

All locations are picked randomly from a 100×100 grid, all grid points with equal chance $\frac{1}{10{,}000}$. The travel time is calculated by the Euclidean distance at a speed of 1 grid point per minute. The travel time is rounded to the nearest *second*. For example traveling from $(24, 67)$ to $(44, 50)$ gives a grid distance of $\sqrt{20^2 + 17^2} = 43.462...$ minutes, which gives as travel time 43 minutes and 28 seconds, or 2608 seconds.

5.2 Engineers

There are 25 engineers, generated randomly as described below.

- The shift starts and ends at the home location of the engineer. The home location is generated randomly from the grid.
- All shifts are 8 hours long, no break is considered. Private travel time is not allowed, i.e. the work starts at the shift's start time by traveling to the first task, and ends by traveling back home, where the arrival should be at or before the shift's end time.
- There are five skills, and the employees have one to four skills. The frequencies of the skills among the engineers are different; they are 10, 14, 15, 18, and 21, so in total there are 78 resource skills (an average of 3.12 skill per engineer).

5.3 Requests

The requests have the following properties.

- Each request has a intake day, the day on which it becomes available. On this day it must be planned to one of the following days (not on the intake day itself). As explained above, we consider the online situation, by which we mean this the request also has a sequence number, and requests should be assigned following the order by the sequence numbers, without knowledge of upcoming requests.
- Each request has a preferred deadline which lies between 5 and 10 days. This reflects the situation that a company has internal or external agreements on how many requests should be planned within the preferred deadline, for example 80% of the requests. If the preferred deadline is 7 days and the intake day is 102, then preferably the task should be scheduled not later than day 109. However, it is allowed to pass the preferred deadline by 50%, so by 3 days in this case. It is allowed, but not preferred, to schedule the task on day 110, 111, or 112. If this is not done, the request is registered as unplanned, and disappears from our simulation. From the company's point of view, an external resource is required to handle this request.
- Each request has a duration (in minutes), the service time needed for the task. The durations are taken between 30 minutes and 60 minutes, with an average of 45 minutes. The distribution is not taken uniformly, but triangular in the following way. Assume we have N different values. The extremes (here 30 and 60) have chance $m = \frac{1}{N^2}$ and the middle value (here 45) has chance $M = \frac{2}{N} + m$. The chance for the other values is found by linear interpolation. The obtained value is rounded to the nearest 5 minutes, so that requests have durations of $30, 35, 40, \ldots$, or 60 minutes.
- Each request has a location, which is taken randomly from the grid.
- Each request requires a skill, which is chosen uniformly from the 5 available skills.
- Per day we generate between 180 and 220 requests. These are chosen from the triangular distribution, which is generated in the same way as for the request durations.

5.4 Time windows

We turn a request to a task by assigning a time window to it. This time window prescribes the start time of a task. In our simulations we use time windows of 2 hours. Since a shift is 8 hours long, it intersects with 4 time windows.

5.5 Validation

The data we generate represents more or less one year. We use the first 50 days as warm-up period. Our validation is over the next 200 days. We also leave out the last 20 days; since we can plan at most 15 days in advance, the end of period effects are not yet visible in the validation period. The results

in validation period are uniform during the whole period; hence the warm-up period is sufficient. What we consider are the requests with intake day in the validation period. In particular we are interested in (the percentage of) unplanned requests, and (the percentage of) the requests that were planned outside the preferred deadline.

5.6 Strategies

We investigate the effect of different strategies, and the use of optimization. For the strategies we consider following ones:

- **First possible day.** We assign the request to the first possible day. If on the first possible day there are several possible shifts, we assign it to the shift and position in which the *extra* travel time is the lowest. The other stops remain in the same order. While inserting we have to consider the time window we try to assign to the request, and the time windows of the already assigned tasks in the shift.
- **Min travel time.** If there are several options for a request, we assign it to the shift in which the **extra** travel time is the lowest, in the same way as in **First possible day**. In case of ties we use the earliest possibility.
- **Add at end.** We only assign a task at the end of a route. Among all possibilities we choose the one with lowest travel time from the previous stop. The travel time back to home is **not** considered. In case of ties, we use the earliest possibility.

All strategies first consider shifts within the preferred deadline. If in this period an option is found, it is always used (though from a higher level, it might be inefficient).

If no option within the preferred deadline is found, we consider the 50% extension interval. It is not prescribed to minimize lateness, but again, if there are options, we will choose one of them.

6 Results for the basic strategies

In this section we present the results of the simulations we executed. We start with the basic strategies, as described in previous subsection. Based on the results we try to improve the simulations by different parameters.

We present the results for of all five datasets in one table. The results of the different datasets are very similar, so there is no added value in presenting five different tables. Per day there are on average 200 tasks, that means that each dataset has around 40,000 tasks to validate, in total there are 200,316 tasks. We present the following performance indicators:

- **Unplanned.** The percentage of the tasks that could not be planned in the 50% extended deadline interval.

- **Lates.** The percentage of tasks that was planned too late, i.e. not within the preferred deadline. The percentage is taken relative to all planned tasks.
- **Avg late.** The average number of days late, among all late tasks.
- **Travel.** The average travel time in minutes per task, calculated as follows: the total travel time on the validation days divided by the total number of tasks scheduled on the validation days; so it represents the travel time per executed task.

6.1 If late then minimize lateness

Here, we follow the basic strategy, but we try to minimize lateness. That means: if we cannot find an option within the preferred deadline, then first we consider the options with 1 day late, and if this returns no result, we look at 2 days late, etcetera. This leads to the following results.

Strategy	Unplanned (%)	Lates (%)	Avg late	Travel
First possible day	15.16	82.21	2.03	23.6
Min travel time	15.11	82.18	2.03	23.5
Add at end	14.07	81.60	2.02	23.0

Results with increasing penalty on late days

We see that the 'First possible day' strategy and the 'Min travel time' perform very similar; the simplest strategy 'Add at end' performs better in all respects. Note that the number of unplanned tasks is high, and the number of tasks that is planned within the preferred deadline is very low. That probably explains that the first two strategies are very similar. That the third strategy performs better is at least partly due to being optimistic about extra travel time, especially if the shift is more or less empty. Before turning to this point, we want to investigate the effect of minimizing lateness. If the planning is so tight, it seems reasonable not to care *per task* about minimizing lateness. We simply want to avoid unnecessary travel time. That is the subject of the next subsection.

6.2 If late then don't care how much

As always, our strategies first try to assign within the preferred deadline, but if no option is found there, we consider all options in the allowed days late, and pick the best one. This leads to the table below.

Strategy	Unplanned (%)	Lates (%)	Avg late	Travel
First possible day	15.16	82.21	2.03	23.6
Min travel time	13.05	79.57	2.50	22.7
Add at end	10.24	76.63	2.85	20.1

Results with equal penalty on all late days

First we note that for the 'First possible day' strategy nothing changes (it still prefers to minimize lateness). The other 2 strategies clearly benefit from this relaxation: all performance indicators drop significantly except the average days late. Still the 'Add at end' strategy is clearly better than the others. Based on this experience we will not penalize extra lateness in further experiments. For instances in which it is hard to assign all tasks, this is probably always a good idea. Let's see what making it easier to start in an empty shift will do.

6.3 In an empty shift ignore the travel time back to home

In construction algorithms it is well known that weighing different travels in different ways can make a difference [Solomon (1987)]. Since we are still in the range of construction, we expect to see this here as well. So we calculate the extra travel time in empty shifts as only the time traveling to the task, and not the way back. The results are presented in the table below.

Strategy	Unplanned (%)	Lates (%)	Avg late	Travel
First possible day	13.69	81.11	2.01	22.5
Min travel time	11.14	77.62	2.65	20.4
Add at end	10.24	76.63	2.85	20.1

Results with travel time reduction for empty shifts

For the strategy 'Add at end' there is no difference (of course). The other two strategies clearly benefit from this change. The 'Min travel time' strategy is closing in on the 'Add at end' strategy. Before discussing a clustering strategy, we do a final check if preferring the first days has some influence on the results. We will keep the empty shift travel time reduction.

6.4 Give travel time reduction to early days

In less challenging planning problems, it might be a good idea to fill the gaps in shifts of tomorrow, and maybe one or two days after that. However, in our cases the shifts will be full anyway. Nevertheless, we do a run with reductions on the travel times: 10 minutes for tomorrow, 6 minutes for the day after, and 2 minutes for the day after that.

Strategy	Unplanned (%)	Lates (%)	Avg late	Travel
First possible day	13.69	81.11	2.01	22.5
Min travel time	11.19	77.68	2.66	20.5
Add at end	10.27	76.57	2.85	20.1

Results with empty shift reduction and reduction for early days

As expected the results are almost the same as in Subsection 6.3. For the 'First possible day' strategy there is no difference (of course). The results for the other two strategies are slightly worse, what could be expected as the 'First possible day' strategy is worse; the planning is so tight that it is a waste of capacity to assign earlier at the expense of higher travel time.

7 Dynamic clustering

Before turning to results with optimization, we want to discuss 'dynamic clustering'. In VRP clustering is well-known; an early reference is [Beasley (1983)]. In particular in periodic VRP, see [Campbell & Wilson (2014)], clustering techniques are widely used.

Although the problem here is not periodic, it shares that there is freedom to decide on which day and time window we place a request. The benefits are comparable: if we have requests to the north and to the south, it would be nice to create tasks on one day for the north, and on the other day for the south, see also Section 4. In periodic VRP it might not be (fully) possible, because some locations in the north and south have to be visited each day. In our situation, we have the clients that have preferences. We ignored these, assuming that the client will accept the proposed time window; in practice the client will have choice from some, for example three, time windows. From the point of view of the service provider, it would be wise the limit the possibilities, especially when the service area is large. The company might do this by assigning its engineers to geographical regions, making sure that tasks for an engineer are in an acceptable range.

Nevertheless, the travel times can be rather high, especially if the engineer is a specialist, serving a large area. In this case it is inevitable that the engineer has to visit a location 'A' far from home in one of the shifts, but we would like that other tasks in this shift are not too far away from location A. This strategy we call 'dynamic clustering': once one or more tasks are assigned to a shift, the new task should be close (in travel time) to *all* already assigned tasks. This maximum travel time between tasks in a cluster, we call the *cluster diameter*. To avoid the risk of partly idle shifts, we do not enforce dynamic clustering for tomorrow and the day after tomorrow. For any time window where dynamic clustering is active, we require that a request is added to a cluster, if possible; only if no cluster is found, an empty shift can be used. In this way we hope to fill the clusters to its capacity.

It depends on the instance what is the best cluster diameter and what is the best day to abandon it. After some experiments on our datasets, it turned out that a diameter of 18 minutes works well. Note that this is around 10% lower than the average travel time per task in the experiments till now. If the cluster diameter is too small, we can expect that many requests will remain unplanned. If the diameter is too high, we can expect less effect from dynamic clustering.

We keep on using the travel time reductions of 10, 6, and 2 minutes, for the first three days; i.e. we use the same parameters as in Subsection 6.4. We can combine dynamic clustering with the strategies described there, because dynamic clustering only restricts the possible options. The simulations lead to the following results.

Strategy	Unplanned (%)	Lates (%)	Avg late	Travel
First possible day	0.68	30.32	1.58	13.7
Min travel time	0.16	18.19	1.93	12.6
Add at end	0.97	27.93	2.17	13.7

Results with dynamic clustering

The improvement is spectacular. In the previous experiments, it seemed that not nearly all tasks could be scheduled. However, when using dynamic clustering and the 'Min travel time' strategy, only 323 out of 200,361 tasks were not planned. The travel time per task drops by 40%; without clustering the travel time consumed around 30% of the shift time, but when using clusters this drops to 21.0%. The conclusion is, that if only construction is applied, dynamic clustering is the best to do. It remains to investigate the effect of optimization.

8 Optimization

We will study the effect of optimization, for the situation without or with dynamic clustering. In both cases we apply an optimization algorithm per day for 30 second. The algorithm we use is an ALS method [Pisinger & Ropke (2007)], in the way described in [Curtois et al (2018)]. For the experiments without clustering, we apply optimization on the first five days; for the experiments with dynamic clustering we only optimize the first two days; we leave the clusters as they are. The simulation without clustering gives the following results.

Strategy	Unplanned (%)	Lates (%)	Avg late	Travel
First possible day	0.73	28.87	1.67	14.4
Min travel time	0.61	32.35	2.35	13.7
Add at end	3.74	58.58	2.70	13.8

Results with optimization on 5 days, without dynamic clustering

Compared to Subsection 6.4, we see a large improvement. This is not unexpected, as optimization in VRPs usually largely improves the solution. Note, however, that these solutions are worse or at best comparable to the results in Section 7. That makes us curious about the result with optimization and dynamic clustering. These are presented in the table below.

Strategy	Unplanned (%)	Lates (%)	Avg late	Travel
First possible day	0.00	5.75	1.19	13.1
Min travel time	0.01	9.80	1.86	11.9
Add at end	0.23	16.36	2.07	12.0

Results with dynamic clustering and optimization on 2 days

The results improve, but not as much as without using clusters. We could have expected this, because all tasks in a route are already close together, and tasks in other routes probably are at some distance. In view on the number of tasks

planned and the preferred deadline, the 'First possible day' strategy works the best; all 200,361 tasks are planned, and only 5.75% of the tasks is outside the preferred deadline. However, if reducing travel time is important, the strategy 'Min travel time' could be used. In this case only 23 tasks are not planned, the lateness is higher, but the travel time per task is lower, saving 4075 hours of travel time.

9 Conclusion

We discussed the call intake process for service planning. We described several methods on how to choose time windows for the requests. We ran a simulation on rather complex data. On this data we noted that giving preference to empty routes, dynamic clustering and intermediate optimization are important aspects; in our simulations the results improve considerably.

This simulation is a simplified model of reality. First there is the client: the best time window for the company might not be feasible for the client, which might worsen the results. However, some preliminary tests with random choices among the options available, show that this effect is limited. It will not change the validity of the discussions in the previous sections.

Another aspect is the homogeneity of the data. First, the geometry is very simple, as well as the expectations for the task properties. In a real situation the service area might be split in regions, and the engineers work only in some of the regions. This restricts the options for the requests, but ensures that if there is only one available engineer, the engineer will not drive two hours to the other side of the service area and two hours back. Experiments show that removing the region restriction during optimization can have a positive effect on the results.

Another aspect of the geometry is that population densities can vary in the regions we consider. In such cases it might be beneficial to decrease the cluster diameter for an engineer working in highly populated regions. It is difficult to predict what choice is the best, but in practice we can monitor the results, and adjust the parameters accordingly.

Finally, there is the planner's acceptance. Especially looking at a *single* route, it is important that the route is optimal at all times. Since we usually there are not more than 10 tasks in a shift, we can guarantee optimality, by solving a dynamic program, see [Held & Karp (1962)]. We call this method Optimized Best Fit (OBF). Each (shift/time window)-combination can be solved in a few milliseconds, making it feasible to combine the strategies with optimization per shift. OBF did not improve the results in our experiments. For the same reason of planner's acceptance, we always use the reductions on the first days. In times that the workload is low, for sure it is preferable to fill the upcoming days. The experiments above show that the'First possible day' strategy competitive with the 'Min travel time' strategy even when the planning is tight.

The method OBF with dynamic clustering supports several companies using PCA's product Marlin, see [PCA]. It helps them in easily providing good options to their clients, and reduces the effort of planning as well as the total travel time, thus improving the productivity.

References

[Beasley (1983)] J.E. Beasley, *Route first—cluster second methods for vehicle routing*, Omega **11**, 403–408, (1983).

[Caceres-Cruz et al (2014)] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A. A. Juan, *Rich vehicle routing problem: Survey*, ACM Computing Surveys **47**, 1–28, (2014).

[Campbell & Wilson (2014)] A.M. Campbell and J.H. Wilson, *Forty years of periodic vehicle routing*, Networks **63**, 2–15, (2014).

[Curtois et al (2018)] T. Curtois, D. Landa-Silva, Dario, Y. Qu, and W. Laesanklang, *Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows*, EURO Journal on Transportation and Logistics **7**, 151–192, (2018).

[Held & Karp (1962)] M. Held and R.M. Karp, *A dynamic programming approach to sequencing problems*, Journal of the Society for Industrial and Applied Mathematics **10**, 196–210, (1962).

[Misir et al (2015)] M. Misir, P. Smet, and G. Vanden Berghe, *An analysis of generalised heuristics for vehicle routing and personnel rostering problems*, Journal of the Operational Research Society **66**, 858–870, (2015).

[PCA] See https://pca.nl/en/functionalities/service-and-maintenance-planning/.

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

[Solomon (1987)] M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Operations Research **35**, 254–265, (1987).

[Strauss et al (2021)] Arne Strauss, Nalan Gülpınar, Yijun Zheng, *Dynamic pricing of flexible time slots for attended home delivery*, European Journal of Operational Research **294**, 1022–1041, (2021).

[Visser (2019)] Thomas Visser, *Vehicle Routing and Time Slot Management in Online Retailing*, EPS-2019-482-LIS, (2019).