On the call intake process in service planning

Gerhard Post · Stefan Mijsters

Abstract In this paper we introduce the call intake process for field services. This process assigns a time window to a client's request for a service. We describe the characteristics of this process and the considerations that can lead to the choice for a time window. The chosen time window will have influence on the quality of the final planning. The quality is measured by the number of planned requests, the lateness of planned requests, and the total travel time for the engineers of the field service.

To investigate the effect of different strategies for the decisions in the call intake process, we constructed challenging datasets for 25 engineers and an average of 200 requests per day. Using simulations on these datasets, we study the effects of several different strategies. Aspects that turn out to be beneficial for the quality are: give preference to empty shifts, cluster tasks in geometrical way, and use intermediate optimization.

The methods explained here are used in practice in the software created by PCA in the Netherlands. The datasets and the simulation program are available at `https://github.com/gfpost/CallIntakeProcess`.

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

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

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

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 at a specific time. A task can be a small repair (in case the company is a housing corporation or an installation company), or some other service. We assume in our datasets that the tasks require between 30 minutes and 60 minutes of service time. An engineer has daily shifts, usually starting at the home address of the engineer, driving to and executing the tasks, and driving to the end destination, which is either the home or the company address[1]. This leads to shifts in which the service time is present in relatively large time blocks. The tasks might be relatively far apart: the service provider might cover a large area in which the tasks are relatively sparse. This leads to the question how to handle incoming requests; at what time window would the request fit the best? This process is called the 'call intake' and is the subject of this paper.

Section 2 describes the routing problem that we have to solve; it is the result of the decisions made during the call intake process that is describes in Section 3 . Section 4 discusses a small example that can be analyzed completely. In Section 5, we describe the simulation set-up, and Section 6 gives the simulation results. The concluding remarks are in Section 7.

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 that defines the earliest and 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. Hence the routing problem essentially consists of daily subproblems. Especially tomorrow is relevant; we need to finalize tomorrow's schedules and inform the engineers on their routes.

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

Quite similar to the Maintenance Personnel Scheduling Problem (MPSP) in [8], 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.
- 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 [8] 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. Minimizing the travel time reduces the direct costs and in addition 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 is.

3 The call intake process

In the previous section we discussed the routing problem to be solved. As explained there, the routing problem usually splits 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 service planning differs quite of lot from home deliveries, as discussed in [15] and [16]:

- 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 care center.

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 6.4. 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':

  State 1, denoted by $(1,1) \odot (.,.)$. Tomorrow the engineer has two tasks at position 1, nothing planned yet for the day after.

  State 2, denoted by $(-1,-1) \odot (.,.)$. Tomorrow the engineer has two tasks at position -1, nothing planned yet for the day after.

  State 3, denoted by $(1,-1) \odot (.,.)$. Tomorrow the engineer has one task at

position 1 and one at position -1, nothing planned yet for the day after.

State 4, denoted by $(1,.) \odot (-1,.)$. Tomorrow the engineer has a task at position 1, and the day after at position -1.

State 5, denoted by $(-1,.) \odot (1,.)$. 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. In practical situations, this is impossible due to the huge number of tasks, additional constraints, the complicated geometry, and the unknown distributions underlying the requests.

In the literature we didn't find any papers describing a similar situation, apart from applications for retail (see for example [15]) and requests that have to be handled within a short time frame (for an overview, see [2,11]). A somewhat related paper is [9], where engineers also have to handle stochastic requests. However, these request include already the time window. In the call intake process the essential feature is that we can choose the time window, based on the previous requests (with a time window) and the expectations of the upcoming requests.

A general framework for this type of problems, under the title of 'Online Stochastic Combinatorial Optimization', can be found in [7]. Here, it is assumed that during the day requests appear, that have to be processed. In our case we cannot collect the requests during the day, as the client is online or on the phone: each request has to be handled immediately. Handling a request means that we tell the client in which time window, the engineer will *arrive* to handle the request. The request with an assigned time window, we will call a *task*. The task is the result of a negotiation between the client and the planner. The planner will offer a favorable time window, favorable in the sense that this time window seems to fit nicely in the schedules of the engineers. However, the client might not be available at this time. Consequently, the planner will offer a different one. In online situations, the client might be able to pick one of, say, three proposals for the first week. If none of these is suitable, the client can switch to the next week, and repeat the process.

To study the effect of different strategies we will run simulations using different strategies. 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. The data files are available at
`https://github.com/gfpost/CallIntakeProcess`.
The data was created in the following way.

5.1 Locations and travel times

All locations are picked randomly from a 100×100 grid, all points with equal chance 0.0001. 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 $\sqrt{20^2 + 17^2} = 43.462...$, which gives a travel time of 43 minutes and 28 seconds, or 2608 seconds.

5.2 Engineers

There are 25 engineers, of which the properties were generated randomly. It resulted in the engineers 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 (randomly generated). The frequencies of the skills among the engineers are 10, 14, 15, 18, and 21, respectively, an average of 3.12 skills per engineer.

5.3 Requests

The requests have the following properties.

- Each request requires one skill, which is chosen uniformly from the 5 available skills. Only engineers with this skill can serve the request.
- Each request has a *location*, which is taken randomly from the grid.
- Each request has a *duration* (in minutes), the service time needed for the task. The durations are 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 a request has a duration of $30, 35, 40, \ldots$, or 60 minutes.

- Each request has an *intake day*, that is 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 that 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, taken from the triangular distribution as above. The deadline reflects the situation that a company has internal or external rules (Service Level Agreements, SLAs) 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 request 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 request on day 110, 111, or 112; such a task we will call *late*. If a request is not scheduled, i.e. if it is not turned into a task, the request is registered as *unplanned*. From the company's point of view, an external resource is required to handle such request.
- Per day we generate between 180 and 220 requests. This number is also chosen from the triangular distribution described above.

5.4 Time windows

We turn a request to a task by assigning a time window to it. This time window is determined by the strategy that we are running. In our simulations we use time windows of 2 hours, aligned with the shifts of 8 hours; each shift intersects with 4 time windows.

5.5 Validation

The data in an instance (270 working days) represent more or less one year. We use the first 50 days as warm-up period, the validation uses the next 200 days. Since we can plan at most 15 days in advance and have 20 days left, there are no end of period effects in the validation period. The results in validation period are uniform during the whole period implying that the warm-up period of 50 days is sufficient. In the validation, we consider the requests with the intake day in the validation period. We are interested in the number of unplanned requests, the number of the late tasks, and the total travel time.

5.6 Strategies

We investigate the effect of different strategies, and the use of optimization at the end of the day. For the basic strategies we consider the ones like in Section 4.

- **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. In case of equality we use the earliest possible time window. The other tasks remain in the same order. While inserting we have to consider the time window we try to assign to the request, and respect the time windows of the already present tasks in the route.
- **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**. This corresponds to the Greedy algorithm, as mentioned in [7].

The strategies first consider shifts within the preferred deadline. If in this period options are found, the 'best' one of these is 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. If in this interval there are options, the strategy will choose the 'best' one of these. Here 'best' refers to the opinion of current strategy.

6 Simulation results

In this section we present the results of the simulations we executed. We start with the basic strategies. Based on the results we try to improve the simulations by guiding the strategies to other choices.

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 results on the following performance indicators:

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

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

Results for the basic set-up

Note that the instances are (seem) very tight. The 'First possible day' strategy is worse than the 'Min travel time' strategy except for the average lateness. The travel time is higher, explaining why less requests could be handled by the 'First possible day' strategy.

6.2 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 [14]. Since we are still in the range of construction, and empty routes will not be present in the end, we expect to see this effect here as well. We will favor empty routes by giving a 50% reduction: as extra travel time, we use 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

Results with travel time reduction for empty shifts

The strategies clearly benefit from this change. 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.

6.3 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

Results with travel time reduction for early days

As expected the results are almost the same as in Subsection 6.2. For the 'First possible day' strategy there is no difference (of course). The results for the 'Min travel time' strategy 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.

6.4 Dynamic clustering

Before turning to results with optimization, we want to discuss 'dynamic clustering', which seems to be new in this area. In VRP, clustering is well-known; an early reference is [1]. In particular in periodic VRP, see [4], 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 west and to the east, it would be nice to create tasks on one day for the west, and on the other day for the east, see also Section 4. In periodic VRP it might not be (fully) possible, because some locations in the west and east 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 time window calculated by the strategy. In practice the client will have choice from some 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 in the time range that dynamic clustering is active, we require that a request is added to a cluster, if possible. Only if *no* cluster is found for a time window, 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. 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. After some experiments on our datasets, it turned out that a diameter of 18 minutes works well. Note that this is more than 10% lower than the average travel time per task in the experiments till now.

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 6.3. We can combine dynamic clustering with the strategies described there, because dynamic clustering only restricts the availably 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

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 total time in the shifts, but when using clusters this drops to 21.0%. The conclusion is that if only construction is applied, dynamic clustering is very effective. It remains to investigate the effects of optimization.

6.5 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 seconds. The algorithm we use is an ALS method [12], in the way described in [5]. 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

Results with optimization on 5 days, without dynamic clustering

Compared to Subsection 6.3, 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 6.4. 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

Results with dynamic clustering and optimization on 2 days

The results still 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 around 800 hours of travel time per instance.

7 Conclusion

We discussed the problem of the call intake 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.

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 usually there are not more than 10 tasks in a shift, we can guarantee optimality within a shift, for example by solving a dynamic program [6]. 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 simulation

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 is competitive with the 'Min travel time' strategy even when the planning is tight.

The OBF method with dynamic clustering supports several service providers using PCA's product Marlin, see [10]. 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 their productivity.

Acknowledgement

This version was created in August 2023 as corrigendum to the paper of 2022. We adjusted the paper based on the useful remarks of reviewers, who are mostly anonymous. Our thanks go to them.

References

1. J.E. Beasley, *Route first - cluster second methods for vehicle routing*, Omega **11**, 403–408, (1983).
2. Y. Borenstein, N. Shah, Nazaraf, E. Tsang, Edward, R. Dorne, A. Alsheddy, and C. Voudouris, *On the partitioning of dynamic workforce scheduling problems*, Journal of Scheduling **13**, 411–425, (2010).
3. 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).
4. A.M. Campbell and J.H. Wilson, *Forty years of periodic vehicle routing*, Networks **63**, 2–15, (2014).
5. 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).
6. 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).
7. P. van Hentenryck and R. Bent, *Online stochastic combinatorial optimization*, The MIT Press, (2006).
8. 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).
9. M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler, *A stochastic and dynamic vehicle routing problem with time windows and customer impatience*, Mobile Networks and Applications **14**, 350–364, (2009).
10. https://pca.nl/en/functionalities/service-and-maintenance-planning/.
11. V. Pillac, M. Gendreau, C. Guéret, and A. Medaglia, *A review of dynamic vehicle routing problems*, European Journal of Operational Research **225**, 1–11, (2013).
12. D. Pisinger and S. Ropke, *A general heuristic for vehicle routing problems*, Computers & Operations Research **34**, 2403–2435, (2007).
13. N. Soeffker, M.W. Ulmer and D.C. Mattfeld, *Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review*, European Journal of Operational Research **298**, 801–820, (2021).
14. M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Operations Research **35**, 254–265, (1987).
15. 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).
16. Thomas Visser, *Vehicle Routing and Time Slot Management in Online Retailing*, EPS-2019-482-LIS, (2019).