A probabilistic approach to pickup and delivery problems with time window uncertainty[☆]

Péter Györgyi, Tamás Kis*

Institute for Computer Science and Control, Kende str. 13-17, Budapest, 1111, Hungary

Abstract

In this paper, we study a dynamic and stochastic pickup and delivery problem proposed recently by Srour, Agatz and Oppen. We demonstrate that the cost structure of the problem permits an effective solution method without generating multiple scenarios. Instead, our method is based on a careful analysis of the transfer probability from one customer to the other. Our computational results confirm the effectiveness of our approach on the dataset of Srour et al., as well as on new, large problem instances.

Keywords: routing, dynamic and stochastic pickup and delivery problems, network flows

1. Introduction

In this paper, we consider dynamic pickup and delivery problems with time window uncertainties as defined recently by Srour et al. (2016). In that model, there is a transportation service provider that gets calls from customers with exact pickup and drop-off locations, but with inaccurate estimations of the time windows for the transportations. The time windows of the service requests become known with certainty only after a second call from the customers, shortly before the service may start.

Srour et al. (2016) describe a couple of real-world scenarios where the above uncertainty is predominant. For instance, harbor pilots, who drive ships to berth, know the locations of the ships, and also where they will berth, but the arrival times of the ships are often uncertain. A related problem is the transportation of containers by tracks from pickup points to drop-off locations, where the exact time of releasing a container at the pickup terminal is not known in advance. They also mention transportation of patients after medical treatments from the hospital to home, where the exact completion time of the treatments is not known with certainty. A related application is on-demand chauffeur services that drive home clients in their own cars after a party. We can extend this list by transportation tasks in a workshop, where semi-finished goods must be transported by fork-lifts, or autonomously guided vehicles between the machining cells, and the pickup and

[☆]This work has been supported by the National Research, Development and Innovation Office (NKFIH), grant no. K112881, and by the GINOP-2.3.2-15-2016-00002 grant of the Ministry of National Economy of Hungary.

*Corresponding author

Email addresses: gyorgyi.peter@sztaki.mta.hu (Péter Györgyi), kis.tamas@sztaki.mta.hu (Tamás Kis)

drop-off locations are perfectly known, but the time window of service is uncertain even if a schedule of the manufacturing operations is broadcasted in advance. As Srour et al. noted, in their examples the customers can request the transportation service by giving the exact location of the pickup and drop-off locations, while providing the time window of starting the service only approximately, e.g., around 2pm. Then, when the customer has more information about its service requirements, it calls the service provider again telling the time window in which it expects the transportation to be started from the pickup to the drop-off location. Since the pickup and drop-off locations may be known well in advance, and also some estimation of the time window of starting the service is preannounced by the customers, the service provider may exploit this information to increase service level, and to reduce its costs.

The main result of this paper is a new algorithm that may help transportation service providers that operate in the above context to find better vehicle tours. Our method is based on estimating the expected operational costs, where missing a customer request is heavily penalized, and the other cost component is the total deadhead cost (operating empty while going to the next pickup location or to the depot). The novelty of our approach is that we solve only a single minimum cost flow problem at each decision point, which determines the next task for each vehicle. In contrast, Srour et al. maintain a set of scenarios, and solve a mixed-integer linear program (MIP) for each of them at each decision point, and then they synthesize the routings of the vehicles. Yet, our method outperforms their method in terms of average total cost on several classes of instances with various characteristics, while it is inferior only in a well-characterized setting. We believe that the success of our approach is due to the cost-structure of the problem at hand, where the penalty of rejecting a customer request is very high compared to deadhead costs. Another advantage of our method is its low running time, the entire simulation run with 100 customers and 40 vehicles was less than a second. The exact solution of the same instance with perfect information and large desired time windows was frequently more than 20 minutes on a modern notebook. This would prohibit the application of scenario based approaches which would repeatedly solve MIPs, as the solution time of a single MIP would be too large, not mentioning that for a large number of customers, one may have to consider much more scenarios than Srour et al. did on their 20-customer instances.

In Section 2, we review the related literature, and in Section 3 we give a formal description of the problem studied. Our method is presented in Section 4, and the datasets used in our computational experiments are described in Section 5. In Section 6, we summarize computational results, where on the one hand, we compare our method to that of Srour et al., and on the other hand, we evaluate it on new, large instances. We conclude the paper in Section 7.

2. Literature review

Dynamic pickup-and-delivery is a rapidly developing field of transportation research, which is certified by a series of recent review papers, see e.g., Berbeglia et al. (2010); Pillac et al. (2013); Psaraftis et al. (2016). In Psaraftis (1988), a vehicle routing problem is characterized as *dynamic*, if the input of the problem is received and updated concurrently with the determination of the routes. Using the terminology of Berbeglia et al. (2010), in this paper we focus on a *one-to-one* problem, where each request has an origin and a destination. In a *dynamic and stochastic problem*, some exploitable stochastic information is available about the dynamically revealed information (Pillac et al., 2013).

The problem studied in this paper has recently been proposed by Srour et al. (2016). In their model, each customer first preannounces its request, then confirms it at some later time, not much before the service actually should start. In the preannouncement, the exact pickup and drop-off locations are provided along with an estimation of the pickup time by means of a time window. However, the preannounced time window can change in the future when the customer confirms its request. On the other hand, the distribution of the difference between the start (or end) of the preannounced and the confirmed time windows is known. The authors propose 4 methods to solve the dynamic problem. All the methods are based on solving a MIP, which models a (static) pickup and delivery problem with some of the customer requests. In the "Ignore" method, preannouncements are ignored and at any time only the confirmed requests are used to determine the tours of the vehicles. In the "Naïve" method, preannounced time windows are used until the customers confirm their requests, from which time on they are replaced by the confirmed ones. However, in the more advanced "MTS-veh" and "MTS-seq" methods, first multiple scenarios are generated for the realization of preannounced, but unconfirmed time windows, which are used along with the confirmed ones in the MIP models to be solved, for more details see the Appendix. The scenario-based approach finds its roots in the paper of Bent and Van Hentenryck (2004), who propose a method for a dynamic routing problem with time windows. In their method, multiple scenarios are generated containing the known requests, and also some possible future requests. Future requests are obtained by sampling their probability distributions. In Tirado and Hvattum (2017), a dynamic and stochastic routing problem of a sea transportation company is studied, where vessels have to transport cargo between sea-ports, and part of the customer requests are known in advance, while the others arrive according to some probability distribution. The scenarios generated at each decision point contain the known, unprocessed requests, and also a sampling of the future requests. The authors propose local search based heuristics to evaluate the scenarios and to choose the next actions for the vessels.

The main novelty of the model of Srour et al. (2016) is that until the customers confirm their requests, only stochastic information is available on the desired service time windows, but the pickup and drop-off locations

are known from the preannouncements. In contrast, in most of the previous work on dynamic vehicle routing problems, the dynamic data consists of the complete user requests, i.e., pickup and drop-off locations, along with the desired time windows are revealed together. Mitrović-Minić et al. (2004) consider a dynamic pickup and delivery problem with time windows where no probabilistic information about future requests are known. Instead, they divide the time horizon into short and long term, and apply different objective functions for the two periods when inserting new customer requests into the tours of the vehicles. Günlük et al. (2006) propose a complex method for continually reoptimizing the schedule of a fleet of vehicles and drivers to adapt it to the new or updated reservations. They maintain a foreground schedule, which is always feasible, and it is modified either by incorporating into it the output of the integer programming based optimization engine run periodically, or by a fast heuristic to respond to changes since the last run of the optimization engine. Ichoua et al. (2006) study a dynamic vehicle routing problem, where the area served is divided into geographical zones, and also the planning time horizon is divided into periods. The requests are not known in advance, but the probability of receiving at least one customer request in a given geographical zone and time period can be calculated. This information is used in order to decide if a vehicle should stay in the same zone and wait for customer requests or move to another zone in the next period. The authors adapt the method of Gendreau et al. (1999) to determine the routing of the vehicles. Ho and Haugland (2011) formulate and solve a dial-a-ride problem, where each customer request has a probability known by the service provider. For finding the routes of the vehicles, a local search, and a tabu search procedure are proposed, in which the next solution is chosen by selecting the best (non-tabu) neighbor of the current solution. The value of a solution is its expected cost, and a procedure is devised for finding the best neighbor in $O(n^5)$ time. Therefore, the computation time of a single iteration is $O(n^5)$, which is considerable if the number of customers n is large. Ferrucci et al. (2013) devise a pro-active real-time control approach for a dynamic vehicle routing problem in which dummy customer requests are generated based on historic data to anticipate future requests. The authors classify the quality of stochastic knowledge attainable from past request information, and they identify structural diversity as a crucial criterion. Albareda-Sambola et al. (2014) consider a multi-period vehicle routing problem with probabilistic information. In their model, the time horizon is divided into time periods, and for the current as well as for the future periods, the probability that the given period is in the time window of the customer is known. For the current period it is 0 or 1, but for future periods, it can be any value between 0 and 1. In each time period, it is decided which customers to serve, and also the tours of the vehicles serving them are planned. Muñoz-Carpintero et al. (2015) propose a method based on evolutionary algorithms to solve a dial-a-ride problem, in which future requests are not known in advance, but the average service patterns from the past are taken into account to devise robust tours for the vehicles.

Table 1: Notations

V, J	fleet of vehicles, and set of customers
$[e_i, \ell_i]$, $[\hat{e}_i, \hat{\ell}_i]$	desired, and respectively estimated (preannounced) time window of customer i
a_i, c_i	preannouncement time, and confirmation time
TW_i	length of the time window ($TW_i = \ell_i - e_i = \hat{\ell}_i - \hat{e}_i$)
L_i	lead time ($L_i = e_i - c_i$)
$dist_i$	Euclidean distance between the pickup and the drop-off location of customer i
f, g	fixed constants for calculating the profit
$profit_i$	profit earned by serving customer i ($f + dist_i \times g$)
h	cost factor for computing the routing cost
J^{rej}	rejected customers
RC	routing cost: $h \times$ total distance operating empty of all the vehicles
LP	lost profit: the total profit missed of all the rejected customers ($\sum_{i \in J^{rej}} profit_i$)
$p(i), d(i)$	pickup and drop-off nodes of customer i in the network
σ	speed of the vehicles
$\tau_{\alpha,\beta}$	travel time between locations α and β
Δ	the parameter of the uniformly distribution of e_i, that is, $e_i \sim\ U(\hat{e}_i - \Delta, \hat{e}_i + \Delta)$

3. Problem statement

In this section, we first define and formalize the static and deterministic problem (Section 3.1). This is a classical pickup and delivery problem with unit vehicle-capacity: there are vehicles that have to serve customers (jobs) to earn as much profit as possible. Due to the capacity of the vehicles, each vehicle can serve at most one customer at the same time. In our problem description, we closely follow that of Srour et al. (2016), however, our integer programming formulation is different from theirs for technical reasons.

Then, we turn to a dynamic and stochastic model in which information about the customers is disclosed gradually over time. We present the details of the dynamic and stochastic model in Section 3.2. Again, the presented model is identical to that of Srour et al. (2016). We have summarized the most important notations of this section in Table 1.

3.1. The static, deterministic problem

A transportation service provider (service provider, for short) has a *fleet of vehicles*, V, and each vehicle can serve only one request at a time. The vehicles are identical from the point of view of the customers. The service provider receives a *set of pickup and delivery requests* from a set of customers J.

A service request (customer) $i \in J$ specifies the pickup and drop-off locations and a time window for the desired pickup time. That is, since typically the customers have some flexibility in their timing, each customer i specifies its desired pickup time by means of a time window $[e_i, \ell_i]$, where e_i is the *earliest pickup time*, and

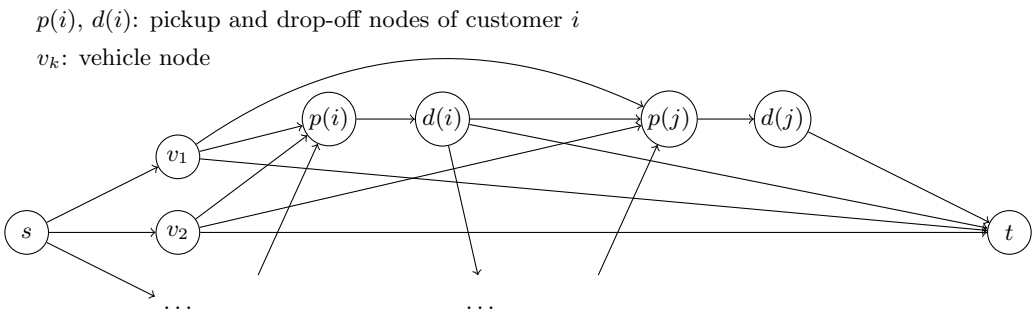


Figure 1: Fragment of the network

$\ell_i = e_i + TW_i$ is the *latest pickup time*, and TW_i is the *length of the time window*. The transportation service for customer i cannot start before e_i, or after ℓ_i. So, if no vehicle starts to serve customer i in the time window $[e_i, \ell_i]$, then the request is *rejected*.

The profit earned by the service provider by serving a customer $i \in J$ is

$$profit_i = f + dist_i \times g,$$

where f and g are fixed amounts in some monetary unit, while $dist_i$ is the Euclidean distance between the pickup and the drop-off location of i. The service provider wants to minimize its *total cost* defined as

$$total\ cost = RC + LP, \tag{1}$$

where RC is the *routing cost* and LP is the *lost profit*. The former one is computed as

$$RC = h \times \text{the total distance of the vehicles operating empty},$$

i.e., the cost of moving from the depot to the first pickup location, from a drop-off location to the next pickup location, or back to the depot. The cost of serving the requests, i.e., a function of the $dist_i$, is not added to the cost function, because that is paid by the customers. The lost profit is

$$LP = \sum_{i \in J^{rej}} profit_i,$$

where the summation is over all the rejected (unserved) customers J^{rej}.

The additional assumptions in the model are as follows. The vehicles start from a depot and have to return to the same depot after finishing operation. Like Srour et al. (2016), we assume that the travel times of the vehicles are deterministic and can be calculated accurately using the distances between locations. The travel time between locations α and β is denoted by $\tau_{\alpha,\beta}$, while their Euclidean distance is denoted by $dist_{\alpha,\beta}$. Using the notation σ for the speed of the vehicles, we have $\sigma \cdot \tau_{\alpha,\beta} = dist_{\alpha,\beta}$.

Now we formulate the problem as a mathematical program. The essence of the model is a network with a source node s and a sink node t, one node for each vehicle v, and for each customer $i \in J$, two nodes, $p(i)$

and $d(i)$, representing the pickup and drop-off locations, respectively. There are directed arcs from the source node to the vehicle nodes, from the vehicle nodes to the pickup nodes of the customers, from the pickup to the drop-off node of the same customer, from the drop-off nodes of the customers to the pickup nodes of other customers, and from each vehicle node and from each drop-off node to the sink node (see Fig. 1). The cost of these arcs are listed below:

$$cost_{\alpha,\beta} := \begin{cases} 0, & \text{if } (\alpha = s,\ \beta = v_k) \text{ or } (\alpha = v_k,\ \beta = t) \text{ or } (\alpha = p(i),\ \beta = d(i)) \\ h \cdot dist_{depot,p(i)} - profit_i, & \text{if } \alpha = v_k,\ \beta = p(i) \\ h \cdot dist_{d(j),p(i)} - profit_i, & \text{if } \alpha = d(j),\ \beta = p(i), \text{ and } j \neq i \\ h \cdot dist_{d(i),depot}, & \text{if } \alpha = d(i),\ \beta = t. \end{cases}$$

Let N denote the set of all nodes in the network and E the set of all arcs. Each arc has capacity 1. The supply of the source node s is set to $|V|$, which has to be carried to the sink node t, which has a matching demand.

Each $s - t$ path in this network represents a routing plan of a vehicle, i.e., the first node of the path after the source node is a vehicle node, then comes a (possibly empty) alternating sequence of pickup and drop-off nodes, and finally, an arc to the sink node representing the way back to the depot.

Now we define an integer program based on the network above. There is a binary routing variable $x_{\alpha,\beta}$ for each arc (α, β). If $x_{\alpha,\beta} = 1$, where α and β denote pickup or drop-off locations, respectively, or $\alpha = s$ (start from the depot), or $\beta = t$ (return to the depot), then it means that there is a vehicle that moves between these locations. In addition, there is a set of continuous variables δ_i for each $i \in J$, representing the time of starting to serve customer i.

After these preliminaries, the mathematical programming formulation is as follows.

$$\text{minimize} \sum_{(\alpha,\beta)\in E} cost_{\alpha,\beta} x_{\alpha,\beta} \tag{2}$$

subject to

$$x_{sv} = 1, \qquad \forall v \in V \tag{3}$$

$$\sum_{(\alpha,\beta)\in E} x_{\alpha,\beta} = \sum_{(\beta,\alpha)\in E} x_{\beta,\alpha}, \qquad \forall \alpha \in N \setminus \{s,t\} \tag{4}$$

$$\max\{e_i, \tau_{depot,p(i)}\} \leq \delta_i \leq \ell_i, \qquad \forall i \in J \tag{5}$$

$$\delta_j + M(1 - x_{d(i),p(j)}) \geq \delta_i + \tau_{p(i),d(i)} + \tau_{d(i),p(j)}, \qquad \forall i,j \in J \tag{6}$$

$$x_{\alpha,\beta} \in \{0,1\}. \qquad \forall (\alpha,\beta) \in E \tag{7}$$

The objective function to be minimized expresses the total cost traveling idle plus the lost profit, since the profit of serving customer i is deduced from the traveling cost for each arc $(\alpha, p(i))$ (cf. definition of $cost_{v,p(i)}$

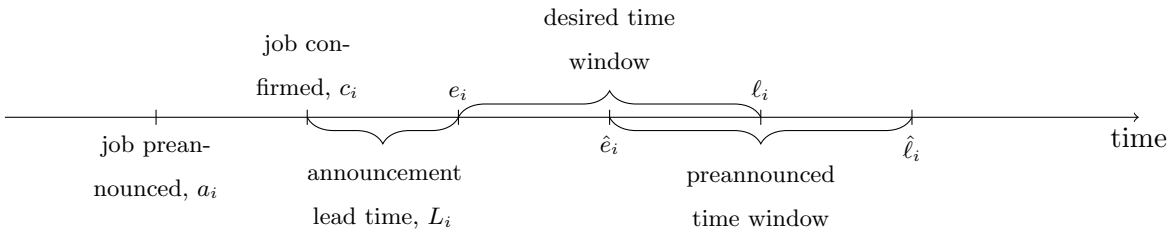


Figure 2: The various data attached to a request

and $cost_{d(j),p(i)}$), and since from $p(i)$ any path must cross the edge $(p(i), d(i))$, any minimum cost feasible solution minimizes the total cost of traveling idle plus $\sum_{i \in J} profit_i \cdot (1 - x_{p(i),d(i)})$, which is the lost profit.

Constraint (3) guarantees that each vehicle has a (possibly empty) task list, (4) ensures that the vehicle cannot stop outside the depot, (5) implies that the vehicles can serve customers only within their time windows, and the service cannot start earlier than a vehicle can get to the respective pickup location, while (6) guarantees that the vehicles have enough time to serve a customer and then travel to the next one. Note that (6) rules out cycles in the feasible solutions, i.e., each arc (α, β) with $x_{\alpha,\beta} = 1$ belongs to an $s - t$ path P, where we have $x_{\alpha',\beta'} = 1$ for each $(\alpha', \beta') \in P$.

Remark: We can drop several arcs from the network: if $e_i + \tau_{p(i),d(i)} + \tau_{d(i),p(j)} > \ell_j$ then it is impossible for a vehicle to serve $j \in J$ after serving $i \in J$, thus we can delete the arc $(d(i), p(j))$.

3.2. The dynamic, stochastic problem of Srour et al. (2016)

As we have mentioned, information about the customers is not known initially. We get these information about each customer in two steps. First, the customers preannounce their service requests. The *preannouncement* for $i \in J$ is made at time a_i, and it specifies the pickup and the drop-off locations, along with an *estimation* of the earliest and latest pickup times, $\hat{e}_i$ and $\hat{\ell}_i$, respectively. These times determine the time window of customer i, i.e. $TW_i = \hat{\ell}_i - \hat{e}_i$. Then, each customer $i \in J$ *confirms* its request by calling the service provider at some time $c_i > a_i$ again, and specifying the *desired pickup time window* with the earliest pickup time e_i, and the latest pickup time $\ell_i = e_i + TW_i$. Each customer i reports its desired time window $[e_i, \ell_i]$ by L_i time units before the service may start, where L_i is a parameter known by the service provider, i.e., $e_i - c_i = L_i$ holds. The above data is illustrated on a timeline in Fig. 2.

The preannounced time window $[\hat{e}_i, \hat{\ell}_i]$ is only an estimation, or forecast of the desired time window $[e_i, \ell_i]$. The difference of $e_i - \hat{e}_i$ can be seen as a random variable known only in distribution in the course of planning until customer i confirms its request. The distribution may be empirically learned by the service provider operating for a longer period. So, we assume that e_i is uniformly distributed in $[\hat{e}_i - \Delta, \hat{e}_i + \Delta]$, for some known parameter Δ. Likewise, the parameter L_i known by the service provider may be learnt from past experience, or may be part of a service contract. These assumptions are from Srour et al. (2016).

At any moment of time, a vehicle can be in one of the following *states*: (i) waiting idle at some location (at the depot, at the pickup, or drop-off location of a customer, or at some waiting area), (ii) on the way to some target location set by the service provider, (iii) transporting a customer to its drop-off location. The service provider can interrupt (ii), and set a new target location for a vehicle, or may simply ask a vehicle to stop and wait at its current position until the next command.

We want to suggest a strategy for the service provider that helps minimize the total cost (2). At any time moment the strategy knows all the preannounced, and confirmed requests, the announcement lead times along with the distribution of the possible realizations of the pickup time windows, and the states and current positions of the vehicles.

4. Algorithmic approach

In this section, we describe a method that helps the transportation service provider to operate its vehicles. Firstly, we give an overview of the entire process in Section 4.1. Then, we present two simple methods that can be used for improving the results: a strategy for reducing the total distance travelled idle is presented in Section 4.2, and in Section 4.3 a simple heuristic is described to estimate the number of vehicles needed to serve the preannounced requests. After that, we outline the core algorithm that has to be applied at every decision point (Section 4.4). In the proposed algorithm, a minimum cost flow problem is solved, where some of the arc weights are based on the probabilities of some events. These probabilities are computed in Section 4.5, and a numerical example is also presented. An illustration of the complete method is provided in the Appendix.

4.1. Overview of the method

The transportation service provider receives a sequence of pickup and delivery requests over time, and it maintains a routing plan for each vehicle under its control. The routing plans are adjusted time and again to take into account the new events. The vehicles get commands only for the next action. New commands can be issued at any moment of time, and the current target location or state of a vehicle can be modified arbitrarily, with the exception that the transportation of a customer cannot be interrupted. From an algorithmic point of view the service provider executes an *Event loop* as shown below.

Algorithm Event loop

Initialization: each vehicle is in the depot, no information is available about the customers.

1. Wait until a new event occurs (a customer preannounces/confirms its request or a vehicle arrives to its target location).

2. Invoke Subroutine Opt (see Section 4.4) with the actual time t_{act}, the actual positions and states of the vehicles and the preannounced or confirmed requests received until t_{act}.

3. According to the output of Subroutine Opt, send new commands to the vehicles.

4. If all customers are served or rejected, then the vehicles go back to the depot, and the processing of events is stopped. Otherwise, proceed with Step 1.

The algorithm maintains the "wall clock" time t_{act}, which is initially set to the beginning of the service period, and updated each time an event is processed. Events are processed in chronological order, no special tie-breaking rule is applied. Re-optimization occurs upon any of the following events:

- a preannouncement is received from a customer,
- a customer confirms its request,
- a vehicle arrives to the target location set by the service provider (waiting area, pickup / drop-off location).

In order to decide about the possible modification of the routing plans, the service provider has to solve an optimization problem while taking into account the state and the current position of the vehicles, the preannounced requests along with the distribution of the possible realizations of the time windows, and the confirmed requests. After solving the optimization problem, a subset of vehicles may receive new commands, i.e., if the result is that a vehicle has to change (i) its target location, or (ii) its state, then it gets a new command. Note that (i) may occur if a vehicle is on the way to a target location, but as a result of re-optimization, it has to go to another location, and (ii) may occur if the vehicle is waiting at some location, and the new routing plan sets a new target location, or it is on the way to some target location, and according to the new routing plan it has to stop at its current position and wait for the next command. As we will see in the computational results, waiting at some location may readily help reduce the total distance traveled idle.

In Section 4.4, we describe the optimization algorithm (Subroutine Opt) to determine the new routing plans for the vehicles.

4.2. Partial execution of commands

In this section, we describe a simple technique to reduce the total distance traveled idle of the vehicles. Suppose a vehicle gets a command to go to a customer which has not confirmed its request yet. This could be a good idea, because if the announcement lead times are short, and the time windows are narrow, when a customer i, say, announces its time window $[e_i, \ell_i]$ at time c_i and there is no vehicle nearby which could arrive

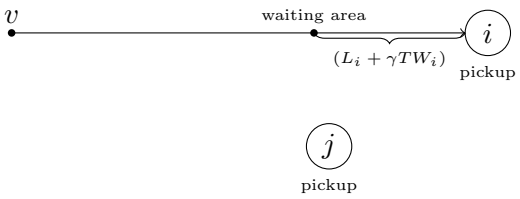


Figure 3: Partially approaching the pickup location of a customer

to the pickup location of i before ℓ_i, then the customer has to be rejected, and it is penalized in the objective function (1).

On the other hand, suppose a vehicle v is on the way to some customer i, and upon arriving the pickup location of i, another customer, say j, not too far away confirms its request, then v may go to the other customer to serve it, and later some another vehicle v_2 may serve i. However, this can be a detour for v. To reduce the total deadhead costs, the vehicles can apply the following strategy. Instead of going to the pickup location of i, the designated vehicle v only approaches i at a distance such that the time needed to arrive to the pickup location of i is $L_i + \gamma TW_i$. This guarantees that when i confirms its request at time c_i, then at time $c_i + L_i + \gamma TW_i$, vehicle v can arrive to the pickup location of i. Since $L_i = e_i - c_i$ by definition, this means that v can arrive to i after a γ fraction of the desired time window of i has passed. We call this strategy *partial execution with parameter* γ. On the other hand, if the vehicles always go to the pickup location of the unconfirmed requests, then they follow the *full execution* strategy.

For an illustration, see Fig. 3. In the figure, we assume that vehicle v has a unit speed, so time is equivalent to distance traveled. Since the travel time from the pickup location of i to the pickup location of j is larger than that from the waiting area to j, should j confirm its request before i, the service provider could modify the routing of v at a smaller cost. In Section 6, we will demonstrate that this simple strategy can reduce the total deadhead cost.

4.3. Reducing the number of vehicles based on the preannounced requests

If the number of the vehicles is significantly larger than necessary, our method may produce high routing costs, since it tries to give some task to every vehicle. To decrease it, we have implemented a simple method: at the beginning of the service period, after receiving the preannounced requests, the service provider can solve the static, deterministic problem replacing e_i and ℓ_i with the preannounced times $\hat{e}_i$ and $\hat{\ell}_i$ (the values e_i and ℓ_i are not known before c_i). Let V^* be the number of the vehicles that serve at least one customer in the optimal solution of the static, deterministic problem. Then, during the service period, the service provider uses only $\min\{|V|, (1+\varepsilon)V^*\}$ vehicles to serve the confirmed requests, where $\varepsilon \geq 0$ is a parameter that can be set experimentally. This method is usable only if all the preannounced requests are received by the service provider before the beginning of the service period.

4.4. Probabilistic model and min-cost-flows

In this section, we describe the optimization problem solved by the service provider each time it wishes to adjust the routing plans of the vehicles.

Suppose that (re)optimization occurs at time t_{act}. We say that a customer $i \in J$ is *rejected* at time t_{act}, if it has already confirmed its request ($c_i \leq t_{act}$), it is not served yet, and the latest pickup time $\ell_i < t_{act}$. Note that at t_{act}, the service provider knows the following data:

- the actual position and task (if any) of each vehicle,
- preannounced information for the customers with $a_i \leq t_{act}$,
- confirmed information (desired pickup time window $[e_i, \ell_i]$) for customers with $c_i \leq t_{act}$,
- the parameter Δ.

The first step of the method is to build a network like we have presented in Section 3.1, using only the information known at t_{act}. After that, a minimum cost flow problem is solved, and from the solution the next action of each vehilce is extracted. Note that flow problems can be solved very fast, see Ahuja et al. (1993), thus we expect very good running times for the whole procedure (see Section 6.2.1).

We construct the network for the optimization problem to be solved at t_{act} by removing some nodes and arcs of the network described in Section 3.1, and by modifying the arc costs based on a probabilistic model. At t_{act}, we abandon all the customers that are already rejected at t_{act}, or being served by a vehicle (on the way from the pickup to the drop-off location), already served, or who have not preannounced their request yet. Let J_{act} denote the set of customers which are not abandoned at t_{act}. We remove from the static network all the nodes $p(i)$ and $d(i)$ with $i \in J \setminus J_{act}$ along with all the adjacent arcs.

Since at time t_{act} each vehicle can be at some location different from the depot, or may be serving a customer, we have to redefine the distances $dist_{v,\beta}$, where $v \in V$ and $\beta \in \{t\} \cup \{p(i) \mid i \in J_{act}\}$. Let $loc(v, t_{act})$ denote the location of vehicle v at time t_{act}.

- If v is serving some customer j at time t_{act}, then for each $i \in J_{act}$, let $dist_{v,p(i)} := dist_{loc(v,t_{act}),d(j)} + dist_{d(j),p(i)}$ (the total distance to be traveled to the pickup location of i through the drop-off location of j), and let $dist_{v,t} := dist_{loc(v,t_{act}),d(j)} + dist_{d(j),t}$ (the total distance to be traveled to the depot).
- If v is not serving a customer, then for each customer $i \in J_{act}$, let $dist_{v,p(i)} := dist_{loc(v,t_{act}),p(i)}$ (the distance between $loc(v, t_{act})$ and the pickup location of i). Likewise, let $dist_{v,t} := dist_{loc(v,t_{act}),t}$ (the distance between $loc(v, t_{act})$ and the depot).

With these distances, we redefine the traveling times $\tau_{\alpha,\beta}$ as $dist_{\alpha,\beta}/\sigma$, where σ is the common speed of the vehicles.

In our model there are two sources of uncertainty: (1) the desired time window $[e_i, \ell_i]$ for each customer $i \in J_{act}$, who has not confirmed its request by t_{act}, and (2) the completion time of serving some customer $i \in J_{act}$, which has not been started by t_{act}. Therefore, we associate a feasibility indicator (a random variable) $I_{\alpha,p(j)} \in \{0,1\}$ with each arc entering the node $p(j)$ for $j \in J_{act}$, that has the following meaning:

(a) If $\alpha = v$ for some vehicle node $v \in V$, then $I_{v,p(j)} = 1$ if and only if the vehicle v can arrive to the pickup location of i before ℓ_j.

(b) If α represents the drop-off location of some customer $i \in J_{act}$, then $I_{d(i),p(j)} = 1$ if and only if it is feasible to serve both of customers i and j (in this order) by the same vehicle, that is, the completion time of serving i plus the travel time to the pickup location of j is not more than ℓ_j. Note that the possibility of serving j after i depends on two events: (i) on the completion time of i (which also depends on several events, like on ℓ_i and on the time when the vehicle that serves i arrives to the pickup location of i) and (ii) on ℓ_j. Our method is based on a simplification: when we calculate the probability of $I_{d(i),p(j)} = 1$, we only take i and j into consideration, and neglect the positions of the vehicles (for details see the next section).

If $\alpha = v$, and customer j has confirmed its request by t_{act}, then the value of $I_{v,p(j)}$ can be determined with certainty. In contrast, if customer j has not confirmed its request, then the value of $I_{v,p(j)}$ is uncertain in our model. Furthermore, the value of $I_{d(i),p(j)}$ may not be decided with certainty even if both of the customers i and j have confirmed their requests by t_{act}, since it depends on when the vehicle, which is supposed to serve both i and j, completes i. We give full details in Section 4.5.

For the sake of simpler computations, we assume that the feasibility indicators are independent.

The arc costs are redefined as follows. The cost of each arc $(\alpha, p(j))$ is redefined as $h \cdot dist_{\alpha,p(j)} - P(I_{\alpha,p(j)} = 1) \cdot profit_j$[1], i.e., we subtract the expected profit of serving customer j from the routing cost from α to the pickup location $p(j)$ of customer j. Further on, for each arc (v,t) we also redefine the cost using the updated $dist_{v,t}$ values. The cost of all other arcs remain as defined for the static, deterministic problem.

After setting the arc costs in the modified network, a minimum cost flow problem is solved. The next statement is about the interpretation of the solution.

Proposition 1. *The minimum cost flow problem always admits an optimal integral (0/1) solution. Furthermore, the arcs with flow value 1 induce $|V|$ (internally) node disjoint $s-t$ paths and possibly isolated directed cycles comprising only customer nodes.*

[1]This form of using the probabilities was suggested by a referee. Our original formula was $P(I_{\alpha,p(j)} = 1) \cdot (h \cdot dist_{\alpha,p(j)} - profit_j)$, but the new formula improved the computational results in terms of average costs by 1-2% points.

Proof. Since arc capacities are uniformly 1, and the network admits $|V|$ arc disjoint $s-t$ paths through the vehicle nodes, there always exists an optimal, integral (0/1), minimum cost $s-t$ flow in the network, see e.g., Ahuja et al. (1993). Furthermore, any feasible, integral $s-t$ flow can be decomposed into a set of $|V|$ internally node disjoint $s-t$ paths, and possibly to some isolated cycles consisting of only customer nodes $p(i)$ and $d(i)$, because from each node $p(i)$ there is a single outgoing arc (to node $d(i)$) of unit capacity. This decomposition immediately provides the tours of the vehicles. Notice that an integral optimal solution cannot contain $s-t$ walks with loops, i.e., a sequence of consecutive edges from s to t with unit flow on each arc of the sequence that passes through an arc at least twice, because such a walk should contain an arc $(p(i), d(i))$ at least twice for some customer i, which is impossible, because then the inflow at node $p(i)$ would be at least two, while the outflow can only be 1 due to the unit capacity of the arc $(p(i), d(i))$. □

Using the proposition, it is easy to determine the next action of each vehicle, we only have to find the outgoing arc from each vehicle node v with unit flow. To sum up, we present a pseudo code of the reoptimization algorithm:

Subroutine Opt

Input: actual time t_{act}, actual positions and states of the vehicles, confirmed information from each customer i with $c_i \leq t_{act}$, preannounced information from each customer with $a_i < t_{act}$.

Output: new actions for the vehicles

1. Build a minimum cost flow problem with respect to t_{act}.
2. Search an optimal (0/1) solution.
3. Determine $|V|$ (internally) node disjoint $s-t$ paths from the arcs with flow value 1 in the solution.
4. Determine the next action for each vehicle (according to the node that follows the vehicle node in a path).

It remains to determine the probabilities $P(I_{\alpha,p(j)} = 1)$, which is the topic of the next section.

4.5. Probabilities

In this section, we set up a probabilistic model for computing the probabilities $P(I_{\alpha,p(j)} = 1)$, where $\alpha \in V \cup \{d(i) \mid i \in J_{act}\}$, and $j \in J_{act}$. In order to simplify notation, for each vehicle v and customer i, let τ_{vi} denote the total time needed for vehicle v to arrive to the pickup location of customer i, i.e., $\tau_{vi} := dist_{v,p(i)}/\sigma$. Furthermore, let $\tau_{ij} := \tau_{d(i),p(j)}$ for each pair of customers $i \neq j$.

In order to set up a probabilistic model for computing $P(I_{\alpha,p(j)} = 1)$, we introduce two random variables, X_i and Y_i, for each $i \in J_{act}$. X_i represents the completion time of serving customer i, that is, the time point

when a vehicle completes the request of customer i. Y_i represents ℓ_i, the end of the desired time window of i. Now we determine the domain of X_i and Y_i, respectively.

As for X_i, if customer i has already confirmed its request by time t_{act}, then the earliest finish time of serving i is $ef_i = \max\{e_i, t_{act}\} + \tau_i$, and the latest possible time to finish i is $lf_i = \ell_i + \tau_i$, where τ_i is the travel time from the pickup location to the drop-off location of customer i. Otherwise, if i has only made the preannouncement by t_{act}, then $ef_i = \max\{t_{act} + L_i, \hat{e}_i - \Delta\} + \tau_i$, and $lf_i = \hat{\ell}_i + \Delta + \tau_i$. In either case, we assume, for the sake of simple modeling, that X_i is uniformly distributed in the interval $[ef_i, lf_i]$.

Concerning Y_i, if customer i has confirmed its request by time t_{act}, then the ℓ_i is known at time t_{act}, and the earliest, and latest time point when the pickup time window of i may end is $ep_i = lp_i := \ell_i$, and $Y_i = lp_i$ with probability 1. Otherwise, if i has only made the preannouncement by t_{act}, then $ep_i = \max\{t_{act} + L_i + TW_i, \hat{\ell}_i - \Delta\}$, and $lp_i = \hat{\ell}_i + \Delta$, and Y_i is uniformly distributed in the interval $[ep_i, lp_i]$.

Now we are ready to determine the probabilities $P(I_{\alpha,p(j)} = 1)$. We distinguish two cases. If $\alpha = v$ for some $v \in V$, then

$$I_{v,p(j)} = \begin{cases} 1, & \text{if } t_{act} + \tau_{vj} \le Y_j \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, $P(I_{v,p(j)} = 1) := P(t_{act} + \tau_{vj} \le Y_j)$. Now we can determine $P(t_{act} + \tau_{vj} \le Y_j)$ easily:

$$P(t_{act} + \tau_{vj} \le Y_j) := \begin{cases} 1, & \text{if } ep_j = lp_j \text{ and } t_{act} + \tau_{vj} \le lp_j \\ 0, & \text{if } ep_j > lp_j \text{ or } t_{act} + \tau_{vj} > lp_j \\ \frac{lp_j - \max\{t_{act} + \tau_{vj}, ep_j\}}{lp_j - ep_j}, & \text{otherwise} \end{cases}$$

Now suppose $\alpha = d(i)$ for some $i \in J_{act}$. Then we have

$$I_{d(i),p(j)} = \begin{cases} 1, & \text{if } X_i + \tau_{ij} \le Y_j \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, $P(I_{d(i),p(j)} = 1) := P(X_i + \tau_{ij} \le Y_j)$, and we have

$$P(X_i + \tau_{ij} \le Y_j) = \begin{cases} 1, & \text{if } lf_i + \tau_{ij} \le ep_j \\ 0, & \text{if } ef_i + \tau_{ij} > lp_j \\ \int_{ef_i}^{lf_i} f_{X_i}(x) P(Y_j \ge x + \tau_{ij}) dx, & \text{otherwise} \end{cases} \tag{8}$$

where $f_{X_i}(x)$ is the probability density function of X_i, i.e., $f_{X_i}(x) = 1/(lf_i - ef_i)$ for $x \in [ef_i, lf_i]$, and 0 otherwise. To compute (8), we define some quantities. Let $p := P(X_i \le lp_j - \tau_{ij})$, $\tilde{p} := P(X_i \le ep_j - \tau_{ij})$,

$q := P(Y_j \geq ef_i + \tau_{ij})$, and $\tilde{q} := P(Y_j \geq lf_i + \tau_{ij})$. Then, we distinguish four cases:

$$P(X_i + \tau_{ij} \leq Y_j) = \begin{cases} pq/2 & \text{if } p < 1, q < 1 \\ (q + \tilde{q})/2 & \text{if } p = 1, q < 1 \\ (p + \tilde{p})/2 & \text{if } p < 1, q = 1 \\ 1 - (1 - \tilde{p})(1 - \tilde{q})/2 & \text{if } p = 1, q = 1. \end{cases} \tag{9}$$

Now we provide a numerical example.

Example 1. *In this example, we have two customers and we want to determine the probability that a vehicle can serve both customers 1 and 2 in this order. Customer 1 wants to go from location* $(1, 0)$ *to* $(2, 0)$*, while customer 2 from location* $(4, 0)$ *to* $(5, 0)$*. The preannounced time window of customer 1 is* $[\hat{e}_1, \hat{\ell}_1] = [10, 12]$*, and* $[\hat{e}_2, \hat{\ell}_2] = [12, 14]$ *for customer 2, and suppose* $\Delta = 2$*. This means that a vehicle with unit speed can start serving customer 1 in the time window* $[8, 14] = [\hat{e}_1 - \Delta, \hat{\ell}_1 + \Delta]$*, thus it arrives to the drop-off location of customer 1 in the time interval* $[9, 15]$ *and to the pickup location* $(4, 0)$ *of customer 2 in* $[11, 17]$ *(see Fig. 4 (a)). On the other hand, the latest pickup time* (ℓ_2) *of customer 2 is in the time interval* $[12, 16]$ *(see the upper part of Fig. 4 (a)). The dotted area in Fig. 4 (b) depicts the possible realizations of* ℓ_2 *(horizontal axis), and the completion time of the request of customer 1 by the same vehicle (vertical axis) enabling serving customer 2 as well.*

Since the probabilities are uniform and the dotted area is exactly the half of the area of the rectangle in Fig. 4 (b), thus the probability sought is 1/2.

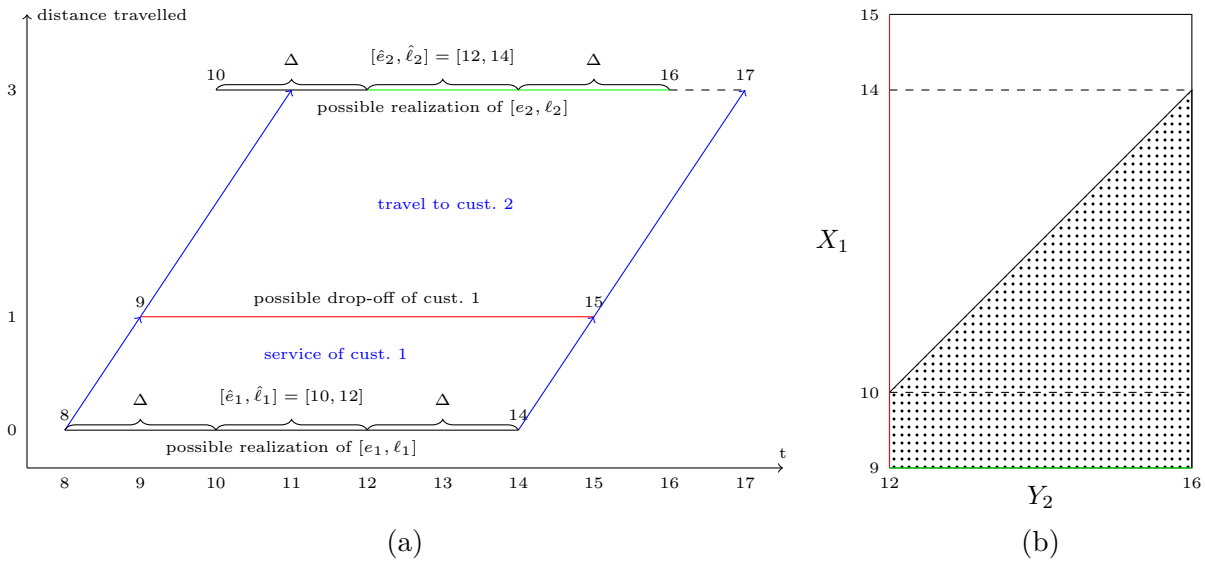


Figure 4: Illustration for Example 1: possible realizations of serving customer 1 and arrival to customer 2 by the same vehicle with the possible realization of $[e_2, \ell_2]$ (a), the possibility of serving customer 2 after customer 1 in the different realizations (b).

Now suppose that customer 1 reports its desired time window $[e_1, \ell_1] = [12, 14]$*. It means that if a vehicle*

with unit speed serves customer 1, it will arrive to the drop-off location in the time interval $[13, 15]$*, and to the pickup location of customer 2 in the time interval* $[15, 17]$*. This information largely decreases the chance of serving customer 2 after customer 1 by the same vehicle, because then the searched probability is 1/16.*

Finally, customer 2 also reports $[e_2, \ell_2] = [14, 16]$*. Since the desired time window is later than the pre-announced one, the chance of serving customer 2 after customer 1 increases. Formally, a vehicle can serve customer 2 after customer 1 only if it arrives to the pickup location of customer 2 before* $\ell_2 = 16$*. This probability is 1/2, since the vehicle will arrive to the pickup location of customer 2 in the time interval* $[15, 17]$ *(with uniform distribution).*

5. Test data

We have evaluated our method on two datasets. The first one was from Srour et al. (2016). The data files are freely available at `https://sites.google.com/site/pdptwinstances/`, accessed on March 31, 2017. These instances help compare the results of our method with a recently published one. Since all the instances of Srour et al. comprises only 20 customer requests, we have also generated larger ones containing 100 customer requests each.

Note that the data files contain all the necessary information: the parameter Δ, and for each customer the coordinates of the pickup and drop-off locations, the preannouncement time a_i, the preannounced time window $[\hat{e}_i, \hat{\ell}_i]$ (and from these, we know $TW_i = \hat{\ell}_i - \hat{e}_i$), the announcement lead time L_i, and also the confirmation time c_i, the desired time window $[e_i, \ell_i]$, where $c_i + L_i = e_i$, and $\ell_i = e_i + TW_i$. It is important that our method uses the preannounced information ($a_i, \hat{e}_i, \hat{\ell}_i$, TW_i and L_i, and the coordinates) only after a_i, and the desired time window $[e_i, \ell_i]$ along with the value of c_i, only after c_i, for each customer i.

5.1. Test data of Srour et al.

The test instances of Srour et al. (2016) are based on transportation requests from a dial-a-chauffeur service in The Netherlands. The parameters that determine the total cost of the service are the same in each case: $f = 6$, $g = 2.7$ and $h = 0.3$ (cf. Section 3). There are 9 vehicles and 20 customers in each instance. The pickup and the drop-off locations are in a 100×100 area and the depot is located at a corner of this area. The vehicles can travel in a straight line between any two points at unit speed. This latter assumption means that travel time (in minutes) and distance travelled have the same nominal values.

The test data contains instances with different geographies, announcement lead times, time windows and parameters Δ (recall that e_i is uniformly distributed in $[\hat{e}_i - \Delta, \hat{e}_i + \Delta]$). The preannounced earliest pickup times, $\hat{e}_i$, are drawn from a uniform distribution spanning a 6 hour period of operation, while the confirmation times are derived from the randomly generated desired earliest pickup time e_i and by the announcement lead time L_i, i.e. $c_i := e_i - L_i$. The preannounced information is known from the beginning.

The default setting is the following: each announcement lead time as well as the length of the time windows is 5 minutes, i.e., $L_i = TW_i = 5$ for each $i \in J$, while the value of Δ is 60. The geography of the customer requests is based on the concept of a center region like a city center: 4 customers want to go from the perimeter to the center, 6 customers in the opposite direction, and the last 10 customers have random pickup and drop-off locations (geography BUS).

Srour et al. developed several datasets, each comprising 100 data files. The datasets were obtained by varying only one of the problem parameters, while keeping the others at the default values. Notice that in all data files in the same test set, all customers have the same L_i, and TW_i values, respectively, and the pickup and drop-off location of the customers do not vary over the instances in the same dataset. There are test cases with modified announcement lead times ($L_i \in \{0, 15, 30, 60\}$), modified time window lengths ($TW_i \in \{10, 15, 30, 60\}$), modified Δ values (30, 45, 90 and 120), and modified geographies (IO20, where each customer wants to go out from the center and RR20, where each customer has random pickup and drop-off locations). For each setting they generated 100 different data files.

5.2. New test data

We have created new, much bigger test cases to assess the performance of our method. The parameters f, g and h are the same as in the data of Srour et al., as well as the speed of the vehicles. The main differences are in the number of the customers, which we have increased to 100, and in the number of the vehicles, we have examined fleets with 20 and 40 vehicles.

Similar to the test data of Srour et al., there are instances with different geographies, announcement lead times, time windows, and parameters Δ. Due to the bigger instances, we also examined cases, where the preannounced, and desired pickup times are drawn from a longer, 12 hours of operation (distributed uniformly). The preannounced time windows are known from the beginning, the confirmation times are determined by $c_i := e_i - L_i$, like before.

We did not change the default setting ($L_i = TW_i = 5,\ \Delta = 60$, geography BUS) and the modified test cases are generated similarly to those of Srour et al. In geography BUS, there are 20 customers who want to go to the center from the perimeter, 30 customers who want to go out from the center, and 50 customers with random pickup and drop-off locations. The other examined geographies are IO100, where each of the 100 customers wants to go out from the center, and RR100, where each customer has randomly generated pickup and drop-off locations. The examined announcement lead times, time windows, and parameters Δ are the same as in the test data of Srour et al. (2016).

For each setting we generated 100 data files, but in contrast to Srout et al., in each data file, we generated not only new time parameters, but also new pickup and drop-off locations for each customer in each data file in a set. The new test instances are available at (Györgyi and Kis, 2017).

6. Computational results

In this section, we give some details of the computer implementation of our solver, information about the running time of our method, and summarize our results in case of both set of instances. The presentation of the results closely follow that of Srour et al. (2016) to get a fair comparison. We will also compare our results to the optimal solution of the static, deterministic problem with perfect information (PI for short). In the sequel, optimal solution will always mean that of the latter problem.

6.1. Implementation

To assess the performance of our method for solving the dynamic, and stochastic problem, we have implemented a simple simulation environment in C++. For solving the minimum cost flow problems, we have used Google Optimization Tools of Google Inc. (2016). Further on, at each decision point, a single run of the minimum cost flow algorithm suffices. For computing the arc costs, we have used the formula (9). The threshold value for the probability of picking an arc has been set to 0.01. By default, we run our method with $\gamma = 0$, i.e., the vehicles approach the pickup location of unconfirmed customers to a distance of L_i. When applying the method of Section 4.3, we set parameter ε to 0.1.

We have mentioned in Section 4 that the structure of the network permits cycles in the solution (consisting of arcs with unit flow) containing only customer nodes. We have developed a variant of our baseline implementation in which if a cycle is detected in a solution, then we eliminate one arc from each strongly connected component of the directed graph consisting of the arcs with positive flow values. The method was still very fast, in 2-3 iterations we got a solution without any cycles, but this extra work had insignificant impact on the entire simulation run.

The results of our method on the new test files have been compared to the optimal solutions, which we have determined by solving the integer program of Section 3.1 by CPLEX 12.6.3 of IBM Inc. (2016).

6.2. Results on the instances of Srour et al.

This section summarizes the computational results on the instances of Srour et al. The main goal of this section is to compare our method to the best method *MTS-seq* of Srour et al., see the Appendix.

6.2.1. Running times

Our simulation runs were very fast, the entire run took only a fraction of a second on a modern notebook computer, so computational times are not provided. The large computation speed is due to the efficient solvability of minimum cost flow problems, see e.g., Ahuja et al. (1993). On the other hand, the MTS-seq method of Srour et al. (2016) solves several integer programs at each decision point, thus the running time of that method is obviously larger. Unfortunately, we do not have the exact running time of their method.

6.2.2. Results with varying Δ

In this set of experiments, we consider datasets with varying Δ values (100 instace for each Δ), and with $L_i = TW_i = 5$, and geography = BUS (the default values). In Table 2, we compare our results to those of Srour et al. (2016). The table is divided into 6 sections. The first line is obtained by using perfect information, i.e., using the time windows $[e_i, \ell_i]$, and solving the entire static and deterministic problem by a MIP solver exactly. Then there are 5 additional sections corresponding to the results with the given Δ values. The rows MTS-seq depict the best results of Srour et al., and the rows 'our' indicate the corresponding results obtained by our method.

The main performance measure used by Srour et al. in Table 3 of their paper, and which we will also use to compare the various methods, is the

$$Average\ Cost = 100 \cdot \left(\frac{\overline{cost}(method)}{\overline{cost}(\mathrm{PI})} - 1 \right),$$

where $\overline{cost}(method) = \sum_I cost(method, I)/100$ is the average cost of a method (MTS-seq, our, PI) over the 100 instances of a class. Furthermore, we will use the average percent deviation (Avg. dev), which is computed by the formula

$$Average\ deviation = 100 \cdot \sum_I \left(\frac{cost(method, I)}{cost(\mathrm{PI}, I)} - 1 \right),$$

and the minimum and the maximum percent deviation, respectively, of each method for each Δ. The minimum is computed as

$$Minimum\ Cost = 100 \cdot \left(\min_I \frac{cost(method, I)}{cost(\mathrm{PI}, I)} - 1 \right),$$

and the maximum is analogous. The minimum (maximum) percent deviation corresponds to the performance of a method on the instance with best (worst) performance of the class.

In the first 4 columns of Table 2, we can see these 4 statistical indicators in this order. The last four columns provide the average rejection cost (lost profit), the average number of rejections, the number of instances without any rejections (out of 100 instances), and the average deadhead distance (over served jobs), respectively. Obviously, larger Δ values imply weaker predictions of the desired time windows of the customers, and thus the percent deviation of the costs of the methods to the PI case increase. Observe that our method constantly provides significantly better results in most aspects, except the minimum and the deadhead costs, where our method is sometimes slightly worse than the MTS-seq. Notice that the main difference between the MTS-seq and our method is due to the rejection costs, and this is the cost factor that increases significantly with Δ.

Further on, the average rejection cost (fourth column) of our method is less than the half of the rejection cost of MTS-seq for every Δ value, while the average total cost of our method (first column) is almost half-way

between the average total cost of MTS-seq and the average total cost of the perfect information case. All in all, we can say that the results are very similar for each Δ: our method clearly outperforms MTS-seq.

This is likely due to the fact that the method of Srour et al. considers only very few (60) scenarios (because for each sample, an NP-hard problem has to be solved, which requires some time). Note that, even if there were only two options for the desired time window of any customer, then for 20 customers, say, there would be $2^{20} \approx 1$ million possibilities for the distinct realizations of the customers' requests. In contrast, our method makes a better use of the probabilistic information and its change over time.

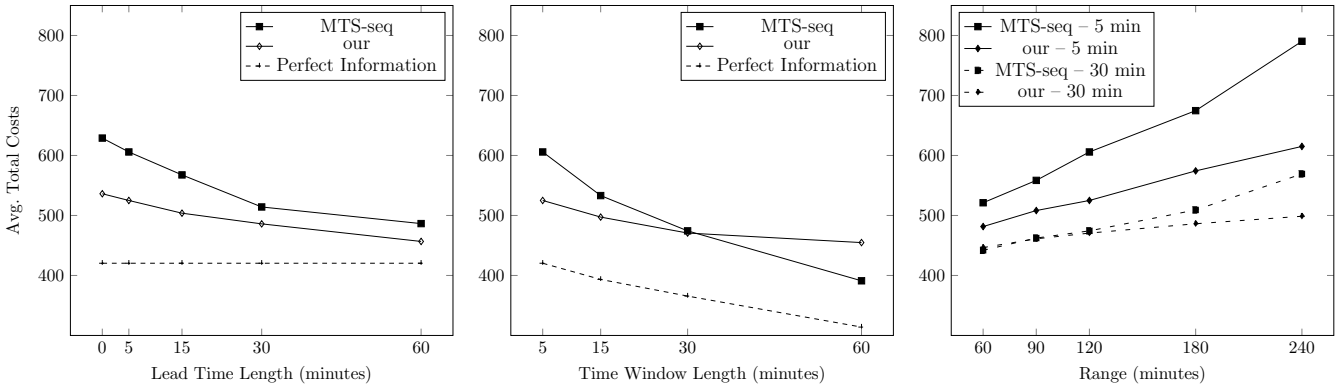
Table 2: Impact of varying Δ on the instances of Srour et al., averages are taken over 100 instances.

		Percent deviation to PI							
		Avg. Cost	Avg. dev.	Min. Cost	Max. Cost	Avg. Rejection Cost	Avg. num. of Rejections	Num. Inst with no Rejections	Empty Dist. per Job Served
	PI	0.0	0.0	0.0	0.0	13	0.2	82	68.8
Range60	MTS-seq	24.0	24.5	1.2	102.3	77.2	0.7	42	77.0
($\Delta = 30$)	our	14.5	14.9	0.2	49.7	32.2	0.5	62	76.9
Range90	MTS-seq	32.9	33.5	0.0	99.5	109.2	1.0	36	79.0
($\Delta = 45$)	our	20.8	21.3	0.3	67.9	48.9	0.7	54	79.5
Range120	MTS-seq	44.0	44.5	2.3	136.9	158.5	1.4	25	80.4
($\Delta = 60$)	our	24.9	25.3	2.0	65.5	60.8	0.9	43	81.0
Range180	MTS-seq	60.5	61.3	7.8	183.8	226.8	1.9	9	82.7
($\Delta = 90$)	our	36.6	36.7	7.5	93.3	112.1	1.4	24	83.2
Range240	MTS-seq	88.0	89.4	10.1	221.0	349.5	2.8	1	85.9
($\Delta = 120$)	our	46.3	46.6	10.9	116.1	158.3	1.9	12	84.6

6.2.3. Results with varying L_i

In this section, results on instances with different announcement lead times $L_i \in \{0, 5, 15, 30, 60\}$ are compared to those of Srour et al., see Fig. 5(a). The results are obtained using the datasets with $TW_i = 5$, $\Delta = 60$, and geography = BUS (the default values). Each point represents the average total cost of 100 instances with a given method. In the figure, the baseline is obtained by using perfect information, and we compare the performance of MTS-seq of and our method. As Srour et al. noted, bigger announcement lead times imply that the methods learn the desired time windows $[e_i, \ell_i]$ earlier, thus the results are better on instances with larger announcement lead times for both methods. In the perfect information case, the time windows $[e_i, \ell_i]$ are known in advance, thus changing the announcement lead times does not influence the results. Observe that the average total cost of our method is roughly halfway between the average total cost of PI, and that of MTS-seq for all L_i values, thus our method evenly outperforms MTS-seq on these instances.

Our method can provide even better results with the partial execution strategy, see Section 6.2.7.



(a) Announcement Lead Time (b) Time Window (c) Time Window and Range Impact

Figure 5: Comparison of the methods on the instances of Srour et al. for (a) varying announcement lead times, (b) time window lengths, and (c) time window length and range. Each point represents the average total cost over 100 instances.

6.2.4. Results with varying TW_i

In Fig. 5(b), the impact of varying the length of the time windows TW_i on the performance of various methods is depicted. The results are obtained using the datasets with $L_i = 5$, $\Delta = 60$, and geography = BUS (the default values). In this figure, we present again the average total cost of 100 instances for each $TW_i \in \{5, 15, 30, 60\}$ (each customer request has the same time window length among the 100 instances for each TW_i value). Clearly, the perfect information case also benefits from larger time windows, so its cost curve decreases as the length of the time windows increases. In contrast with the previous set of instances, the differences between the results obtained by our method and by MTS-seq are strongly depend on the length of the time windows. Our method strongly dominates MTS-seq for short time windows ($TW_i \in \{5, 15\}$), it has similar performance for $TW_i = 30$, and it gives worse results than MTS-seq for $TW_i = 60$. As Srour et al. noted, larger TW_i yields more flexibility in the assignment of jobs to vehicles. They have also observed that the greater freedom decreases the costs and this is consistent with the literature. Though our method achieves lower costs on instances with longer time windows, the difference between the average total cost of PI, and that of our method does not decrease as the lengths of the time windows increase. This is due to the relatively high routing costs of our method, but we were able to refine it with partial execution, and the results with this heuristic are significantly better in case of large TW_i values (see Sect. 6.2.7).

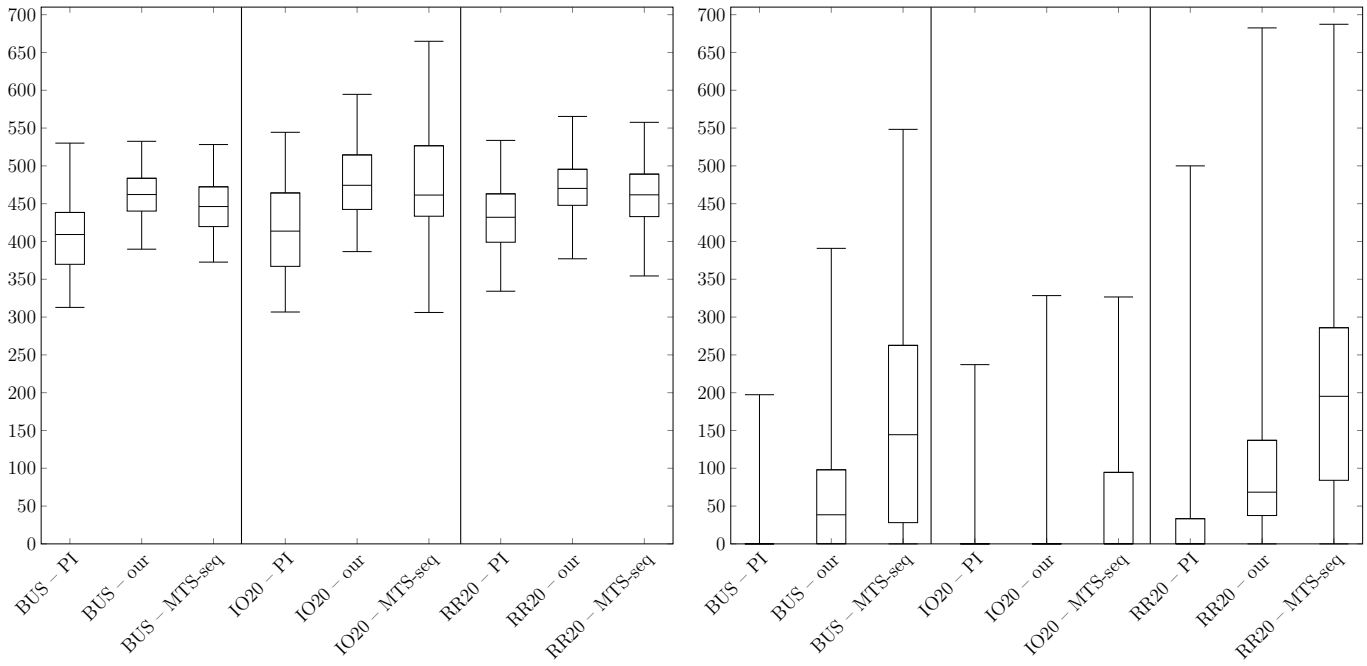
6.2.5. Results with varying Δ and TW_i

Like Srour et al., we have also made experiments with varying Δ and TW_i parameters. In Fig. 5(c), we compare our method to MTS-seq on 2×5 datasets, i.e., one series with datasets such that $TW_i = 5$ and $\Delta \in \{30, 45, 60, 90, 120\}$ (solid lines), and another with $TW_i = 30$ and Δ from the same set (dashed lines).

Notice that the range in the figure is just $2 \times \Delta$, and our figure has similar content to Fig. 7 of Srour et al. (2016). Also note that the results with $TW_i = 5$ are already summarized in Table 2, although in that table we compare the performance of the methods to the perfect information case. Observe that on both series of datasets, our algorithm provides superior results to MTS-seq, and in fact as the range (Δ) increases, the difference between the performance of the two methods increases as well.

6.2.6. Results with varying geography

Now $L_i = TW_i = 5$, $\Delta = 60$ (the default values), but the geography of the pickup and drop-off locations is varied. Fig. 6(a) shows the routing costs of the different methods on instances with the different geographies (100 instance for each geography). The method *PI* stands for the perfect information case (solved by a MIP solver), and *our* refers to our method and MTS-seq is that of Srour et al. In each case, the figure depicts the routing cost of the instance with the lowest, the 25th, the 50th, the 75th and highest (100th) routing cost, thus we can see roughly the distribution of the routing costs. E.g. the first column shows that there are 25 BUS instances, where the routing cost is between 313 and 370 in case of perfect information, 25 other, where it is between 370 and 409, etc. In Fig. 6(b), we can see the distribution of the rejection costs. Observe that while the routing costs of the solution found by our method and MTS-seq are similar, our method produces significantly lower rejection costs than MTS-seq for each geography.



(a) Routing Costs

(b) Rejection Costs

Figure 6: Comparison of the methods perfect information (PI), our, and MTS-seq of Srour et al. on different geographies. Minimum, 1st quartile, median, 3rd quartile, and maximum routing costs (a), and rejection costs (b).

Table 3: Results with full and partial execution strategies on the instances of Srour et al. Averages are taken over 100 inctances.

	our method				MTS-seq
	full execution	$\gamma = 0$	$\gamma = 0.3$	$\gamma = 0.9$	
$L_i = TW_i = 5$ (default setting)	530.3	525.0	530.9	537.6	605.9
$L_i = 60$, $TW_i = 5$	487.3	456.7	455.9	455.8	486.5
$L_i = 5$, $TW_i = 60$	456.1	454.8	435.9	417.8	391.1

6.2.7. The impact of partial execution

In this section, we summarize the results of the method described in Section 4.2. Briefly, the method modifies the results significantly only if the announcement lead times or the time windows of the instance are relatively long. This is due to the fact that on other instances, the value of $L_i + \gamma TW_i$ is small, thus the vehicles cannot reduce the routing costs considerably. Apart from the default setting ($L_i = TW_i = 5$, $\Delta = 60$, geography BUS), we considered 100-element datasets with $L_i = 60$, and with $TW_i = 60$, respectively, while the other parameters were at default values, see Table 3. The first 4 columns depict results obtained by our method with full and partial execution strategies (see Section 4.2), while the last column depicts the reference data of MTS-seq. On the default dataset, the effect of partial execution is negligible. On the dataset with $L_i = 60$, full execution provides slightly weaker results than MTS-seq, and partial execution with $\gamma = 0$ or $\gamma = 0.3$ are both better than MTS-seq. On the dataset with $TW_i = 60$, our method provides the best results with $\gamma = 0.9$, which is still weaker than MTS-seq. This is the only dataset where our methodgives weaker results than MTS-seq.

6.3. Results on large instances

Recall that there are 100 customers and fleets with 20 or 40 vehicles in our new instances. We have made some similar experiments as in Section 6.2, and we also tested the heuristic of Section 4.3 to adjust the number of the vehicles. However, now we focus on only a few aspects of the results, because they confirm our most important perceptions. Since solving the corresponding large MIPs takes a significant time, there is no chance to compare our method to MTS-seq on large instances (see the next section about the running times), and we compered our results only to the optimal solution of the static, deterministic problem (with perfect information).

6.3.1. Running times

The entire simulation run on any new instance file took less than a second, thus our method is still very fast on large instances. To assess the quality of the solutions, the integer program using perfect information was solved using CPLEX 12.6.3. In most cases, this integer program is solvable within a few seconds, however instances with larger time windows require significantly more time. For $TW_i = 60$, there are instances that

CPLEX could not solve even within 20 minutes, thus we do not present any results for these instances. Note that, the heuristic of Section 4.3 for reducing the number of vehicles based on preannounced information also requires the solution of an integer program, thus the subroutine is not applicable if solving the integer program requires too much time.

6.3.2. Results with varying Δ

Similarly to Section 6.2.2, we give detailed results for $\Delta \in \{30, 45, 60, 90, 120\}$, where for each Δ we have 100 different instances with parameters $L_i = TW_i = 5$ and with the BUS geography.

We will assess the performance of our method on 100 customer instances, while varying the fleet size and the hours of operation. The results are summarized in Table 4 (6 hours of operation), and in Table 5 (12 hours of operation). The structure of these tables is quite similar to that of Table 2, but now we have results for fleets with 20 and with 40 vehicles, respectively. Note that for a given fleet size and hours of operation, we always compare our method to the corresponding optimal solution with the same parameters.

We have three mainly different cases: (i) when the number of the vehicles is not enough to serve most of the customers (20 vehicles for 6 hours of operation), (ii) when the number of the vehicles is roughly sufficient (40 vehicles for 6 hours of operations and 20 vehicles for 12 hours of operation), and (iii) when we do not need all of the vehicles to serve almost all of the customers (40 vehicles for 12 hours of operation). In the first case, there are big rejection costs even in the optimal solutions. This implies even higher rejection costs for our method, and these costs increase as Δ increases, due to the higher uncertainty. The routing costs are similar for the optimal solution and for our method. In case of (ii), the rejection costs in the optimal solutions are very small. For smaller Δ values, the rejection cost of our method is low, however, it produces a higher routing cost than the optimal solution. For larger Δ values the rejection costs are also larger, though they are not as large as in case of (i). If we have 12 hours of operation and 40 vehicles (case (iii)), then the rejection costs are negligible for both methods. One drawback of our method is that if there are much more vehicles than needed to serve all the customers, then, since it tries to give some work to all the vehicles, the total routing costs can be significantly higher than in the optimum. This is so, because then all the vehicles leave and return to the depot, which contributes largely to the total deadhead travel. In the next section, we empirically confirm the effectiveness of the heuristic of Section 4.3 to reduce the routing costs.

Recall that these instance files differ not only in the Δ values, since they are generated separately with the same parameters (TW_i, L_i and geography). However, since the statistics are taken over 100 instances, this does not cause significant differences in the main instance characteristics (e.g., the average optimal total cost on instances with 40 vehicles and 6 hours of operation is always between 1454 and 1490).

Table 4: Impact of varying Δ on instances with 100 customers within 6 hours, and 20 or 40 vehicles. Averages are taken over 100 instances.

		Percent deviation to PI							
		Avg. dev.	Min. Cost	Max. Cost	Avg. Total Cost	Avg. Rej. Cost	Avg. Rout. Cost	Avg. Num. of Rej.	Num. Inst. with no Rej.
Range60 ($\Delta = 30$)	Perfect Inf. (20 veh.)	0	0	0	2424.3	1191.2	1233.1	12.8	0
	our (20 veh.)	41.0	19.9	78.6	3393.2	2192.0	1201.2	22.7	0
	Perfect Inf. (40 veh.)	0	0	0	1471.5	1.6	1469.9	0.1	92
	our (40 veh.)	24.9	11.4	35.5	1835.0	10.6	1824.4	0.2	86
Range90 ($\Delta = 45$)	Perfect Inf. (20 veh.)	0	0	0	2174.2	936.2	1238.0	11.7	0
	our (20 veh.)	61.0	29.8	99.7	3473.5	2288.3	1185.2	26.3	0
	Perfect Inf. (40 veh.)	0	0	0	1454.6	6.2	1448.4	0.3	77
	our (40 veh.)	35.1	16.0	49.8	1960.8	19.0	1941.8	0.4	69
Range120 ($\Delta = 60$)	Perfect Inf. (20 veh.)	0	0	0	2251.2	980.1	1271.1	12.0	0
	our (20 veh.)	67.8	36.8	127.7	3742.2	2545.7	1196.5	28.1	0
	Perfect Inf. (40 veh.)	0	0	0	1490.0	7.5	1482.5	0.2	84
	our (40 veh.)	38.3	20.7	56.6	2057.5	45.0	2012.5	0.8	48
Range180 ($\Delta = 90$)	Perfect Inf. (20 veh.)	0	0	0	2362.4	1109.1	1253.3	12.6	0
	our (20 veh.)	87.8	38.4	144.8	4381.4	3223.8	1157.6	31.7	0
	Perfect Inf. (40 veh.)	0	0	0	1473.5	3.0	1470.5	0.2	84
	our (40 veh.)	49.1	26.4	84.2	2194.3	186.5	2007.8	2.5	9
Range240 ($\Delta = 120$)	Perfect Inf. (20 veh.)	0	0	0	2314.0	1058.6	1255.4	12.1	0
	our (20 veh.)	104.7	53.8	174.1	4658.6	3516.6	1142.0	33.2	0
	Perfect Inf. (40 veh.)	0	0	0	1462.2	3.8	1458.4	0.2	84
	our (40 veh.)	61.0	38.3	92.5	2348.0	346.6	2001.4	4.2	1

6.3.3. Results with varying L_i and TW_i

Table 6 summarizes the differences between our method and the optimal solution for different announcement lead times with $TW_i = 5$, and for different time window lengths with $L_i = 5$ (in both cases $\Delta = 60$ and geography = BUS). Unfortunately, on instances with $TW_i = 60$, it was too time consuming to compute the optimal solution, thus we abandon this case. We summarize results for a period of operation of 6 and 12 hours, respectively. We considered fleet sizes of 20 and 40 vehicles, and for 40 vehicles we also used the heuristic of Section 4.3 to reduce the fleet size, the corresponding results are in the row 40*. For 20 vehicles the heuristic returned 20 vehicles in the vast majority of cases. Each entry represents the average cost over 100 instances. Observe that for large announcement lead times and for large time windows, the results with 40* vehicles are much better than with 40 vehicles, since with large announcement lead times or with large

Table 5: Impact of varying Δ on instances with 100 customers within 12 hours, and 20 or 40 vehicles. Averages are taken over 100 instances.

		Percent deviation to PI							
		Avg. dev.	Min. Cost	Max. Cost	Avg. Total Cost	Avg. Rej. Cost	Avg. Rout. Cost	Avg. Num. of Rej.	Num. Inst. with no Rej.
Range60 ($\Delta = 30$)	Perfect Inf. (20 veh.)	0	0	0	1122.1	5.3	1116.8	0.1	94
	our (20 veh.)	18.7	9.3	44.2	1331.9	59.7	1272.2	0.9	50
	Perfect Inf. (40 veh.)	0	0	0	1119.3	0.7	1118.6	0.0	97
	our (40 veh.)	19.7	11.1	42.2	1338.7	2.9	1335.8	0.0	97
Range90 ($\Delta = 45$)	Perfect Inf. (20 veh.)	0	0	0	1102.1	2.4	1099.7	0.0	97
	our (20 veh.)	28.4	15.0	59.4	1414.2	121.0	1293.2	1.6	20
	Perfect Inf. (40 veh.)	0	0	0	1100.8	0.3	1100.5	0.0	99
	our (40 veh.)	29.3	18.6	42.9	1421.4	1.2	1420.2	0.0	98
Range120 ($\Delta = 60$)	Perfect Inf. (20 veh.)	0	0	0	1109.8	6.1	1103.7	0.2	88
	our (20 veh.)	37.6	17.6	69.8	1526.9	224.7	1302.2	3.4	6
	Perfect Inf. (40 veh.)	0	0	0	1107.9	2.1	1105.8	0.1	92
	our (40 veh.)	36.7	21.3	52.3	1512.6	2.9	1509.7	0.1	88
Range180 ($\Delta = 90$)	Perfect Inf. (20 veh.)	0	0	0	1098.5	5.7	1092.8	0.1	87
	our (20 veh.)	67.4	30.9	160.3	1836.8	527.9	1308.9	7.4	0
	Perfect Inf. (40 veh.)	0	0	0	1096.4	2.1	1094.3	0.1	91
	our (40 veh.)	53.5	37.6	74.7	1679.3	8.5	1670.8	0.3	77
Range240 ($\Delta = 120$)	Perfect Inf. (20 veh.)	0	0	0	1110.7	4.3	1106.4	0.1	92
	our (20 veh.)	94.6	37.9	166.4	2159.4	862.6	1296.8	10.1	0
	Perfect Inf. (40 veh.)	0	0	0	1109.5	1.5	1108.0	0.1	95
	our (40 veh.)	67.8	41.0	91.8	1857.9	8.1	1849.8	0.2	84

time windows it is possible to serve most of the customers with fewer vehicles, while fewer vehicles incur smaller routing costs. Further on, for $L_i = 60$, the presented results are outstanding: the average percent deviation to the corresponding optimal solution is around 10% both in case of 20, and 40* vehicles, which is well beyond our expectations. Also notice that with 20 vehicles we get much better results with 12 hours of operation than with 6 hours of operation, since in the latter case the vehicles have a much denser schedule. This effect is much reduced with 40 vehicles, where the results with 6 and with 12 hours of operation are very similar.

Table 6: Average percent deviation to PI for different lead times and time window lengths. * denotes the usage of the heuristic of Section 4.3. Each table entry represents the average percent deviation of 100 instances.

period of op.	vehicles	$L_i = 0$	$L_i = 5$	$L_i = 15$	$L_i = 30$	$L_i = 60$	$TW_i = 5$	$TW_i = 15$	$TW_i = 30$
		$TW_i = 5$					$L_i = 5$		
6 hours	20	88.6	67.8	44.7	23.5	9.1	67.8	66.9	57.9
	40	43.1	38.3	33.1	26.3	21.2	38.3	45.1	58.3
	40*	70.4	48.9	28.8	14.9	7.9	48.9	40.4	33.0
12 hours	20	49.3	37.6	24.7	14.5	9.6	37.6	30.3	33.0
	40	40.5	36.7	32.6	26.1	21.6	36.7	42.0	51.0
	40*	53.7	42.7	27.4	14.6	9.3	42.7	35.0	31.4

7. Conclusions

In this paper, we have studied a stochastic pickup and delivery problem proposed recently by Srour et al. (2016). In this problem, the job locations are known in advance, but we do not have precise information about the desired pickup time windows. The customers first preannounce their transportation requests by giving the distribution of the desired pickup time windows, and the exact time parameters become known only shortly before the service can actually start. We demonstrate that a simple algorithm may outperform a more heavy scenario-based approach on several classes of problem instances. The average cost of the solutions found by our method was significantly smaller than that the MTS-seq method of Srour et al. in almost every type of instances that was proposed by Srour et al. Our method has weaker performance on instances with large time windows, but we have presented a modification of our method that improves the results on these instances. Our method is outstanding on instances where the importance of the rejection costs is high: as the uncertainty increases, the length of the announcement lead times or that of the time windows decreases, the advantage of our method increases. Furthermore, the running time of our method is negligible, since it solves a single minimum cost flow problems at each decision point. Tests on new, larger instances show similar behavior, so it scales up well with the size of the instances.

We think that the MTS-seq method of Srour et al. (for details, see the Appendix) performs well in terms of routing costs, because their MIP finds an optimal routing for each scenario, and their route synthesis also does a good job. However, as we have mentioned, their method examines only 60 scenarios due to high running time, which is quite few if we compare this with ratio of the length of a desired time window and the length of the interval, where this time window is located before the customer finalize its request. Our method makes a better use of the probabilistic information and its change over time. The results of Section 6 show that the advantage of our method mainly comes from the low rejection costs.

Our findings open up a number of further directions. For instance, for the specific problem, can we make better routing decisions in order to improve the results when large time windows allow more room for

optimization? Can a similarly simple approach be effective in other dynamic and stochastic vehicle routing problems?

Appendix A. The method of Srour et al. (2016)

In their method, Srour et al. maintain a set of 60 scenarios. Each scenario consists of a *presumed* time window $[\bar{e}_i, \bar{\ell}_i]$ for each customer i (the authors assume that all customers make a preannouncement before route planning starts). The basic idea is that until a customer confirms its request, the presumed time window is obtained by shifting the preannounced time window by a random number ξ_i drawn from $[-\Delta, +\Delta]$, i.e., $[\bar{e}_i, \bar{\ell}_i] = [\hat{e}_i + \xi, \hat{\ell}_i + \xi]$, and after confirmation, it becomes the desired time window $[e_i, \ell_i]$, (for notation see Section 3.2). After setting up the 60 scenarios, an optimal routing of the vehicles is determined for each scenario by solving a mixed integer program where the time windows of the customers is set to the presumed ones (one MIP is solved for each scenario). The result is 60 routing plans from which a sophisticated procedure synthesizes a routing plan to be executed by the vehicles. Let t_{act} be a time point when some decision is to be made (when some customer confirms its request, or some vehicle completes serving a customer). Then in each scenario, the time windows are revised as follows. If a customer confirms its request at time t_{act}, then its presumed time window becomes $[e_i, \ell_i]$. For unconfirmed customers, if the presumed earliest pickup time is passed, i.e., $\bar{e}_i < t_{act} + L_i$, a new presumed time window is drawn from the distribution while making sure that $\bar{e}_i > t_{act} + L_i$. Based on the updated time windows, the vehicles' departure times are updated for each plan, and it is checked whether the plan is still feasible. If a plan becomes unfeasible, then a MIP is solved for the corresponding scenario of (updated) time windows, and finally, a new plan is synthesized out of the routing planes for the scenarios to be executed by the vehicles.

Appendix B. Sample run of our algorithm

We illustrate our algorithm on a small example with 1 vehicle and 3 customers. Table 7 summarizes the data. Suppose that the speed of the vehicle is 1.

Table 7: Data for the example.

customer	preann. time	est. time window	pickup loc.	drop-off loc.	calltime	time window
	a_i	$[\hat{e}_i, \hat{\ell}_i]$			c_i	$[e_i, \ell_i]$
0	1	[22 23]	(3, 1)	(1, 2)	2	[27, 28]
1	1	[18, 19]	(0, 3)	(1, 1)	1	[18, 19]
2	2	[20, 21]	(2, 5)	(4, 1)	3	[20, 21]

The first event, when some information becomes known about the customers is at $t = 1$, which is the preannoucement time of customer 0 and the call time of customer 1. Note that, at $t = 1$ we do not know

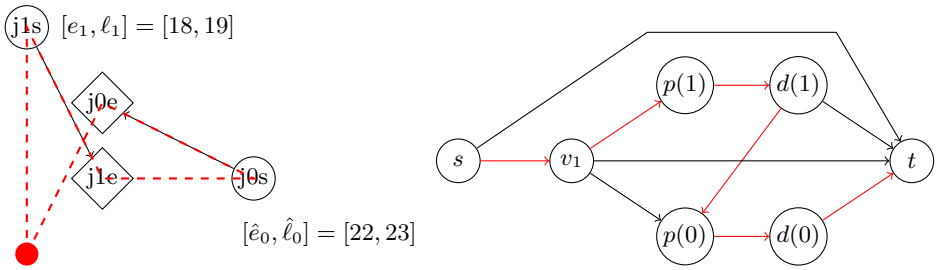


Figure 7: Left: known information at $t = 1$; the vehicle is at the red point (at $(0, 0)$). Right: the network flow problem at $t = 1$; the edges with flow value 1 in the optimal solution are red.

anything about customer 2. According to Algorithm Event loop, we have to invoke Subroutine Opt with the information shown on the left side of Fig. 7 (the pickup and the drop-off locations of customers 0 and 1, the estimated time window of customer 1 and the preannounced time window of customer 0). The right-hand-side of the same figure shows the corresponding network flow problem. The red arrows indicate the arcs with flow value 1 in the optimal solution. This solution implies that the vehicle has to depart towards customer 1 (the dashed line on the left of Fig. 7 indicates the planned routing of the vehicle according to the solution of the network flow problem). After that, the next event occurs at $t = 2$, which is the call time of customer 0 and the preannouncement of customer 2. By this time, the vehicle is at (0,1), see the left-hand-side of Fig. 8 for the actual situation. Subroutine Opt builds a new network, see the middle picture of the same figure. Again, the red arrows indicates the optimal solution, thus the vehicle has to change its target. It has to set off towards customer 2, since by serving customer 2 and then possibly customer 0 the expected total cost is smaller. The later events do not change the important parts of the network, thus the vehicle will serve customer 2 and then customer 0. Note that, in an optimal solution (which we can calculate if each time window $[e_i, \ell_i]$ is known from the beginning), the vehicle would go straight to customer 2 and then to customer 0, thus it avoids the by-pass towards customer 1. The right-hand-side of Fig. 8 depicts by a red curve the routing of the vehicle in the dynamic and stochastic model, and by a blue curve the optimal solution of the static, deterministic model.

8. Acknowledgements

The authors are indebted to the anonymous referees for constructive comments that helped to improve the manuscript. Furthermore, a key formula has been altered by following a suggestion of a referee, and this led to better results.

Conflict of interest

There is no conflict of interest.

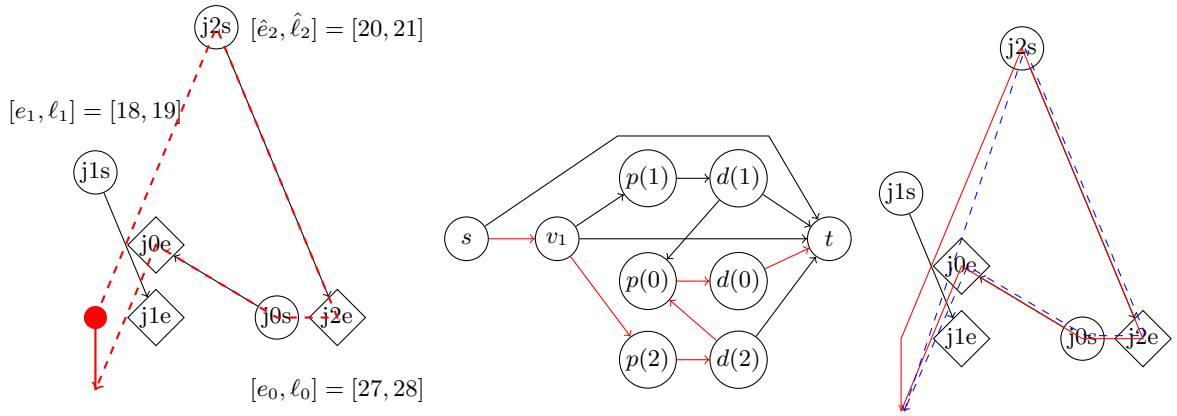


Figure 8: Left and middle: the current situation and the corresponding network flow problem at $t = 2$. Right: the solution of the algorithm (red) and the optimal solution in case of perfect information (dashed-blue).

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice hall.

Albareda-Sambola, M., Fernández, E., and Laporte, G. (2014). The dynamic multiperiod vehicle routing problem with probabilistic information. *Computers & Operations Research*, 48:31–39.

Bent, R. W. and Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987.

Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15.

Ferrucci, F., Bock, S., and Gendreau, M. (2013). A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. *European Journal of Operational Research*, 225(1):130–141.

Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science*, 33(4):381–390.

Google Inc. (2016). *Google Optimization Tools*. `https://developers.google.com/optimization/`, (accessed May 31, 2017).

Günlük, O., Kimbrel, T., Ladanyi, L., Schieber, B., and Sorkin, G. B. (2006). Vehicle routing and staffing for sedan service. *Transportation Science*, 40(3):313–326.

Györgyi, P. and Kis, T. (2017). *New test data for dynamic and stochastic pickup and delivery proplems*. `https://igor.xen.emi.sztaki.hu/~gyorgyip/pdptw_new_instances`.

Ho, S. C. and Haugland, D. (2011). Local search heuristics for the probabilistic dial-a-ride problem. *OR Spectrum*, 33(4):961–988.

IBM Inc. (2016). *IBM ILOG CPLEX Optimization Studio V12.6.3 documentation.* `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.3/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html`.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2006). Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225.

Mitrović-Minić, S., Krishnamurti, R., and Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669–685.

Muñoz-Carpintero, D., Sáez, D., Cortés, C. E., and Núñez, A. (2015). A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. *Transportation Science*, 49(2):239–253.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. In Golden, B. and Assad, A., editors, *Vehicle Routing: Methods and Studies*, volume 16, pages 223–248.

Psaraftis, H. N., Wen, M., and Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31.

Srour, F. J., Agatz, N., and Oppen, J. (2016). Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. *Transportation Science*.

Tirado, G. and Hvattum, L. M. (2017). Improved solutions to dynamic and stochastic maritime pick-up and delivery problems using local search. *Annals of Operations Research*, 253(2):825–843.