Scheduling last-mile deliveries with truck-based autonomous robots

Nils Boysen, Stefan Schwerdfeger, Felix Weidinger

PII: S0377-2217(18)30477-6
DOI: 10.1016/j.ejor.2018.05.058
Reference: EOR 15171

To appear in: *European Journal of Operational Research*

Received date: 1 August 2017
Revised date: 23 May 2018
Accepted date: 27 May 2018

Please cite this article as: Nils Boysen, Stefan Schwerdfeger, Felix Weidinger, Scheduling last-mile deliveries with truck-based autonomous robots, *European Journal of Operational Research* (2018), doi: 10.1016/j.ejor.2018.05.058

Highlights

- We consider a novel last-mile concept relying on autonomous robots.
- A novel scheduling problem for launching robots from trucks is derived.
- Efficient solution procedures for this problem are provided.
- We benchmark the novel concept with traditional truck deliveries.

Working Paper

Scheduling last-mile deliveries with truck-based autonomous robots

Nils Boysen[1,*], Stefan Schwerdfeger[2], Felix Weidinger[1]

August 2017
Revised: March 2018

[1]: Friedrich-Schiller-Universität Jena
Lehrstuhl für Operations Management
Carl-Zeiß-Straße 3, 07743 Jena, Germany
http://www.om.uni-jena.de
{nils.boysen,felix.weidinger}@uni-jena.de

[2]: Friedrich-Schiller-Universität Jena
Lehrstuhl für Management Science
Carl-Zeiß-Straße 3, 07743 Jena, Germany
http://www2.wiwi.uni-jena.de/Entscheidung/
stefan.schwerdfeger@uni-jena.de

[*] Corresponding author, phone +49 3641 9-43100

Abstract

To reduce the negative impact of excessive traffic in large urban areas, many innovative concepts for intelligent transportation of people and freight have recently been developed. One of these concepts relies on autonomous delivery robots launched from trucks. A truck loads the freight dedicated to a set of customers in a central depot and moves into the city center. Also on board are small autonomous robots which each can be loaded with the freight dedicated to a single customer and launched from the truck. Then, the autonomous robots move to their dedicated customers and, after delivery, autonomously return to some robot depot in the city center. The truck can replenish robots at these decentralized depots to launch further of them until all its customers are supplied. To assess the potential of this innovative concept, this paper develops scheduling procedures which determine the truck route along robot depots and drop-off points where robots are launched, such that the weighted number of late customer deliveries is minimized. We formulate the resulting scheduling problem, investigate computational complexity, and develop suited solution methods. Furthermore, we benchmark the truck-based robot delivery concept with conventional attended home delivery by truck to assess the potential of this novel last-mile concept.

Keywords: Scheduling; Transportation; City logistics; Autonomous robots

1 Introduction

To attenuate the negative effects of transportation on congestion, safety, and environment in large city centers, plenty of innovative concepts for moving people and freight have recently been developed. Among those concepts, especially focusing on freight transportation, are, for instance, goods distribution with electric vehicles [17], drone-based freight transport [16], delivery into the trunk of a parked car [18], and crowdsourcing of deliveries [4]. An overview of the latest developments and concepts in city logistics is, for instance, provided by Savelsbergh and Van Woensel [19]. The novel concept focused in this paper relies on autonomous robots launched from a truck (see Figure 1).

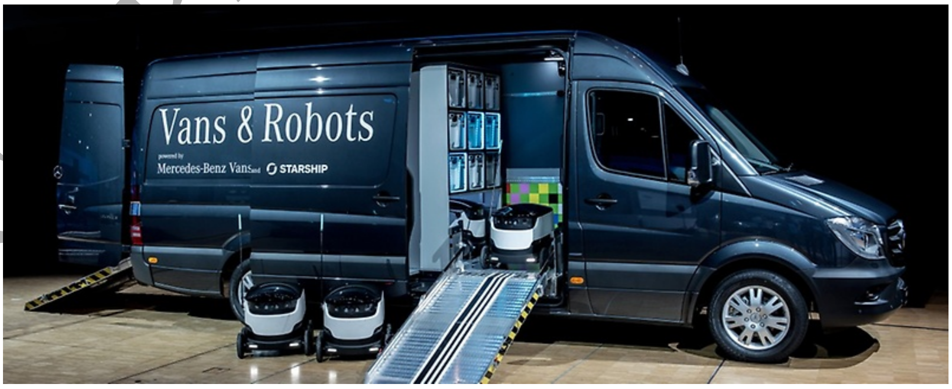

Figure 1: Truck-based autonomous robots[1]

In September 2016, German truck producer Mercedes-Benz Vans announced a strategic partnership with Starship Technologies [8]. The latter is an Estonian start-up company which

[1]Source: Daimler https://www.daimler.com/innovation/specials/future-transportation-vans/

develops autonomous robots for last-mile deliveries. Starship's robots move along sidewalks and weigh no more than 40 pounds, fully loaded. They can be applied to deliver parcels or groceries directly from stores or specialized hubs. Customers can monitor deliveries via smartphones which are also applied to open the locked cargo bay of the robots upon arrival. Afterwards, the robots autonomously return to their store or hub. Due to safety reasons, the robots are only permitted to move at pedestrian speed, so that either a dense (and costly) network of stores or depots is required or (comparatively) long delivery times have to be accepted. To avoid these drawbacks, the aforementioned alliance advocates a concept where trucks are used as mobile launching platforms for the robots.

The truck-based robot delivery concept works as follows. A truck loads the shipments for a set of customers at a central depot where the goods to be shipped are stored. A fixed part of the truck's loading capacity is reserved to also load autonomous robots on board. The truck, then, moves into the city center and, once a drop-off point is reached, one or multiple robots are loaded with shipments and launched to autonomously deliver their goods to customers. Each robot has a capacity for a single shipment and after delivery, returns to a decentralized robot depot within the city center. Note that at these decentralized depots, only robots are stored but not goods, so that only a small garage is required but no complete distribution center. The truck moves onwards to successive drop-off points until all robots are released. If the truck still has further shipments for additional customers on board, it can move to one of the decentralized robot depots to load another batch of robots. In this way, the process goes on until all shipments on board of the truck are launched with a robot and the truck can return to the central depot in order to load more shipments for the next set of customers.

This paper introduces scheduling procedures for an efficient truck-based robot delivery. Specifically, we aim at a truck route along a sequence of stops consisting of drop-off points and decentralized robot depots along with a launching plan of robots, such that the weighted number of late customer deliveries, after the announced delivery date, is minimized. This problem is defined, computational complexity is proven, and exact as well as heuristic solution methods are introduced. The devil's advocate could say that the second problem is treated before the first. First, all technological challenges should be solved and government admission for the robots on public roads should be reached (first problem). Afterwards, there is still enough time to develop suited scheduling procedures (second problem). However, it seems hard to properly anticipate the potential gains of the truck-based robot delivery concept without a detailed scheduling procedure which is able to exactly quantify important performance indicators, such as the resulting travel distances. Therefore, to get a first impression of the efficiency of a truck-based robot delivery, we apply our scheduling procedures to different data sets and benchmark them with traditional attended home delivery by truck. Furthermore, we compare our approach to an alternative process of the truck-based robot delivery concept. Instead of applying decentralized robot depots, the truck could also wait for the return of launched robots. Our computations, however, show that, due to the robots' low travel speed, this alternative mode of operation is considerably inferior.

The remainder of the paper is structured as follows. A brief literature review is provided in Section 2. Then, Section 3 defines our optimization problem and Section 4 elaborates on suited exact as well as heuristic solution procedures, whose computational performance is tested in Section 5. The comparison of the two alternative modes of operation of truck-based robot deliveries and their benchmarking with conventional attended home deliveries by truck is provided in Section 6. Finally, Section 7 concludes the paper.

2 Literature review

Several innovative concepts for the transport of people and goods have recently been developed. Instead of trying to summarize the vast body of literature which has accumulated in this area, we refer to the recent survey papers on trends in transportation [21], distribution with electric vehicles [17], shared transportation [15], and city logistics [19]. One of the latest concepts, announced in September 2016 [8], is truck-based robot delivery. To the best of the authors' knowledge, this paper is the first to derive scheduling procedures for this concept. Therefore, our literature survey will only address two related fields: (i) truck-based drone delivery and (ii) related transportation problems with similar mathematical structure.

(i) Another innovative last-mile concept where smaller delivery vehicles are launched from a truck serving as a mobile depot is truck-based drone delivery (see [2]). Here, unmanned aerial vehicles (also denoted as drones) are launched from the top of a truck to deliver shipments to customers. In spite of all physical differences between the delivery vehicles, from the planning perspective, both concepts are very similar. Drones also have a small capacity and can service just one customer at a time. The main differences, however, which lead to distinct routing problems, are the following:

- Our autonomous robots require the customers to be at home, so that a person can unlock the robot's cargo bay and withdraw the shipment. Drones can deliver unattended, for instance, just be releasing their shipment on the balcony of a customer's apartment. Therefore, existing papers (i.e., [2, 16, 22, 5]) on truck-based drone deliveries do not consider customer due dates, but rather minimize the makespan until all customers are serviced.
- Furthermore, drones travel unobstructed at a high speed through the air, so that the truck can wait for the return of a drone after each delivery. This is also a possible mode of operation for our autonomous robots and we benchmark this alternative in Section 6. However, our robots only travel at pedestrian speed and unavailable customers may further delay the robots. Therefore, this mode of operation would cause excessive waiting times for the truck. Small robot depots, to which the robots can autonomously return and where the truck can refill its capacity for robots, seem to be the better choice. To the best of the authors' knowledge, comparable drone depots have not been considered in the literature yet.

(ii) As the routing of vehicles has a long tradition, it is anything but astounding that there are quite a few related problems which share some characteristics with our truck-based robot delivery:

- Time-related versions of the traveling salesman problem (TSP), such as the traveling repairman problem (also known as minimum latency problem [20]), come pretty close to our problem. In this extension of the TSP, the travel between a pair of cities is related with a travel time and the objective is to minimize the sum of travel times along the tour towards each city (e.g., [1]). There are also other variants like the traveling repairman problem with time windows [12] and the time-dependent TSP [11]. Our problem also has a time-dependent objective function and aims to minimize the weighted number of late deliveries. However, in contrast to the TSP (and its aforementioned variants) we do not have to visit each drop-off point but just a selection of them. Moreover, the same robot depots may be visited multiple times and customers (cities) are not directly approached by the truck but only by launched robots.

- Another variant of the TSP related to our problem is the covering salesman problem introduced by Current and Schilling [7]. Here, the shortest tour along a subset of given nodes is sought, such that every node that is not on the tour is within a predefined covering distance of a node on the tour (see also [3]). Similar to our problem, not every node is to be visited and reducing the distances for the robots between their launching point and the customer they serve is certainly a good idea. In our problem, however, each customer has an individual due date for parcel delivery, so that a detailed launching schedule of robots is required.

- In our problem, the route of the truck and the launching schedule of robots have to be synchronized, so that our problem falls into the domain of vehicle routing problems with synchronization (for a survey see [9]). Among these problems, the truck and trailer routing problem (TTRP) comes quite close to our problem. In the TTRP, a vehicle composed of a truck with or without a detachable trailer serves the demand of a set of customers reachable by truck and coupled trailer or just by the truck (e.g., see [14]). The main difference between TTRP and our problem is that the robots, once launched, separately serve customers, while the trailer cannot serve customers on its own and remains where it is parked.

To conclude, this paper introduces a novel routing problem that has not been treated in the literature yet.

3 Problem definition

Consider a single truck loaded with shipments for a set C of customers to be supplied. Initially, the truck is positioned at location γ with δ robots on board. This location may be the central depot where the customers' shipments are loaded and an initial delivery schedule is determined. In case of unforeseen events, such as congestion, however, a short-term adaption of the initial plan may become necessary, so that γ and δ can also represent the truck's current status-quo on a half-finished tour once a replanning is initiated. Each customer $j \in C$ is associated with a deadline d_j, representing the latest desirable delivery time, and a weight w_j. This weight quantifies the penalty term if the customer's shipment arrives after the deadline.

Our problem assumes a given discrete set $\mathcal{P} = \mathcal{C} \cup \mathcal{D} \cup \mathcal{R}$ of locations which can be visited by the truck and the robots. This set contains the following location types:

- Set $\mathcal{C}$ defines the customer locations, so that $\mathcal{C}_j$ defines the location of customer j. Customers get exclusively supplied by robots and not directly by the truck, so that only robots may access these locations.

- Set $\mathcal{D}$ represents the drop-off points where the truck stops, loads robots with shipments, and launches them towards a customer.

- Finally, we have set $\mathcal{R}$ representing decentralized robot depots. Here, the truck can also launch (a non-restricted number of) robots and, before departure, replenish robots. Up to K new robots can be taken on board where K defines the truck's maximum robot capacity.

Given the speed of the truck as well as the much lower velocity of the robots and the distances among all locations $\mathcal{P}$, parameters $\vartheta^t_{v,v'}$ and $\vartheta^r_{v,v'}$ define the travel time of truck and robot, respectively, between two points $v, v' \in \mathcal{P}$.

One part of the solution for the truck-based robot delivery scheduling (TBRD) problem is a truck route. A route is defined by a sequence of stops starting at initial location γ. Each following stop of the truck can only be at a drop-off point $\mathcal{D}$ or a robot depot $\mathcal{R}$. Furthermore, a solution for TBRD requires the definition of a robot launching schedule. Such a schedule defines the number of robots launched from a stop of the truck and the subset of customers supplied from there. Each robot can only load one shipment at a time, so that the number of robots launched has to equal the cardinality of the customer subset supplied. The objective of TBRD is to find a truck route and a launching schedule that minimizes the weighted number of late deliveries. More formally, TBRD can be defined as follows:

A solution π for TBRD consists of a sequence of tuples (v, L) defining the subset $L \subseteq C$ of customers supplied by the robots launched from location $v \in \mathcal{D} \cup \mathcal{R}$. We say a solution π is feasible, if the following conditions hold:

- We have $\bigcup_{(v,L)\in\pi} L = C$, that is, all customers are supplied.
- For any pair of distinct tuples $(v, L) \in \pi$ and $(v', L') \in \pi$, we have $L \cap L' = \emptyset$, that is each customer is serviced by a single robot launched from a single location.
- For the very first tuple (v, L) of π, we have $v = \gamma$, that is the truck route starts at the truck's initial position γ.
- Let π^0 be the set of tuples referring to the foremost drop-off points of π before the first decentralized robot depot is accessed. We have $\sum_{(v,L)\in\pi^0} |L| \leq \delta$, that is, no more robots can be launched before reaching the first robot depot than those that were initially loaded.
- Let π^i be the set of tuples referring to all drop-off points following on the i-th visit at a robot depot before either the next robot depot is visited or the end of the route is reached. For all visits i of robot depots within π we have $\sum_{(v,L)\in\pi^i} |L| \leq K$, that is at most K robots can be launched after any robot depot visit. Once the capacity K for robots is depleted and further customers have to be supplied, another depot has to be visited to replenish the robots on board of the truck.

Let $\sigma(j)$ and v_i return the number of the stop in the truck route of π where the robot supplying customer $j \in C$ is launched and the location of the i-th stop within π, respectively. Then, the delivery time φ_j of a customer j is

$$\varphi_j = \sum_{i=1}^{\sigma(j)-1} \vartheta^t_{v_i,v_{i+1}} + \vartheta^r_{v_{\sigma(j)},j}.$$

Delivery time φ_j amounts to the driving time of the truck from its initial location up to the location where the robot supplying customer j is launched plus the driving time of the robot from its launching location up to the location $\mathcal{C}_j$ of the customer. Consider U being the set of customers supplied late, i.e., those customers with $\varphi_j > d_j$, TBRD seeks one feasible solution that minimizes the weighted number of late deliveries, i.e., minimizes

$$Z(\pi) = \sum_{j \in U} w_j.$$

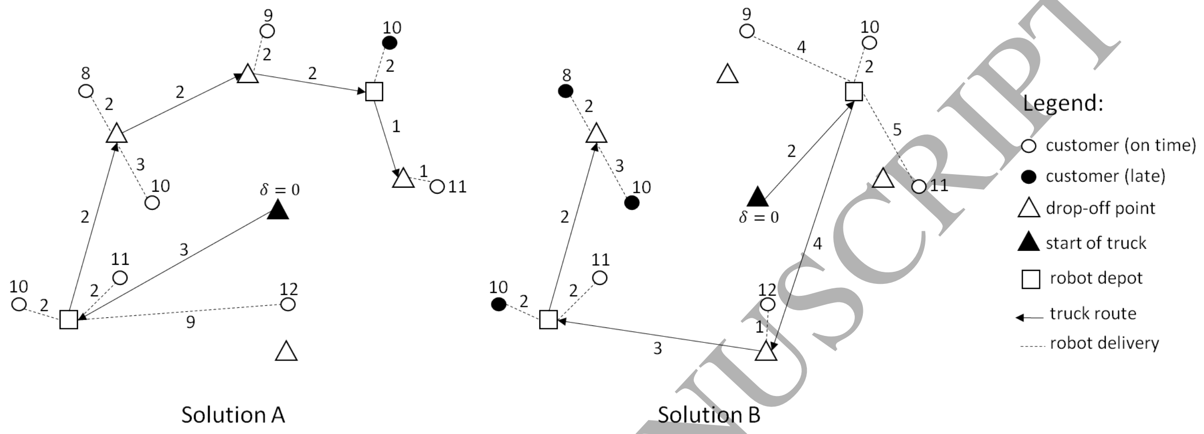

Figure 2: Two alternative solutions for an example instance of TBRD

Example: Consider the example of Figure 2 where two alternative solutions for an instance of TBRD are depicted. Here, eight customers are to be supplied which are represented by the (non-)filled circles. The deadlines of the customers are given above the circles and all customers are assumed to have unit weights. The filled triangle defines the start position of the truck which starts without robots on board and has a capacity of $K = 3$. Drop-off points (robot depots) are symbolized by non-filled triangles (squares). The arcs starting from the truck's starting point γ represent the truck routes which only go through drop-off points and robot depots. Robot deliveries either from drop-off points or depots to customers are symbolized by dashed lines. The duration of truck and robot movement along the respective relations are given by the weights. Customers supplied late, where the shipments arrive after the deadline, are highlighted by filled circles. Thus, solution A is better than solution B with two delayed shipments less.

Our TBRD is based on some (simplifying) assumptions that are elaborated in the following:

- The truck exclusively acts as a mobile depot which has the shipments on board and launches robots. It does not access customer locations to also service customers directly. We leave the evaluation of this alternative mode of operation, where we can choose between robot and truck delivery, up to future research.

- We presuppose that robots are not a bottleneck resource. This implies that we have no shortages of robots at depots and trucks can always be replenished with robots to capacity. If robots are scarce, they need to be modeled individually which adds plenty of complexity to the problem. Their return to the robot depots would have to be considered, so that either only fewer robots can be loaded by the truck or additional waiting time for returning robots occurs. We leave this extension of our basic TBRD up to future research.

- If we assume that there are always enough robots, this also impacts the objective. The robots have to be unloaded at each customer location, so that we have attended home delivery. This necessitates that a delivery time window has been agreed with each customer in which he/she has to be at home to receive the shipment. If robots are no bottleneck, however, a premature arrival at a customer location is unproblematic because the robot can wait until the time window is reached and the customer is available. Only an arrival after the time window is to be avoided because otherwise unsatisfied customers result. Therefore, the end of the delivery time window agreed with the customer constitutes the deadline d_j within our TBRD.

- We restrict our problem to a single delivery vehicle serving a given set of customers. Thus, we assume that a distribution of customers among multiple trucks has already been planned on a higher planning level. A dynamic adaption of the customer set, which may occur if a customer cancels a shipment on short notice, is not considered. However, in this case a new planning run of TBRD for the altered situation can be executed even when the truck is already on its way.

- We presuppose deterministic travel times of truck and robots. A simple way to integrate the inevitable uncertainty coming along with these values is to integrate safety buffers, e.g., by reducing either the customers' deadlines or travel speeds of the vehicles. Recall that also a new planning run of TBRD can be executed whenever travel times have (considerably) altered.

- To keep the problem structure as basic as possible, we do not consider the stop times of the truck at drop-off points and depots explicitly. Compared to the driving times stop times are short, so that they seem negligible and not that much precision should be lost. However, waiting times of the truck can be integrated implicitly by adding a service time s (stop times, time to replenish/launch robots) to the travel times $\vartheta^t_{v,v'}$ as a constant factor, i.e. $\bar{\vartheta}^t_{v,v'} = \vartheta^t_{v,v'} + s$. Furthermore, in a preprocessing step the time for launching robots can be either added to travel times $\vartheta^r_{v,v'}$ or respectively subtracted from the customers' deadlines d_j.

Given this problem definition, the following theorem constitutes the complexity status of TBRD.

Theorem 1. *TBRD is $\mathcal{NP}$-hard in the strong sense.*

The proof is by reduction from the feasibility version of the traveling salesman problem (TSP) which asks whether a feasible tour exists that visits each of the given n cities exactly once, returns to the origin city, and has a tour length not exceeding T. The TSP is well-known to remain $\mathcal{NP}$-complete even if only a path through all cities is sought and the starting city is fixed [10].

Proof. Our transformation scheme from TSP to TBRD is defined as follows. The starting city of the TSP is also the starting location γ of the truck which, initially, has $\delta = n - 1$ robots on board. At each location of the $n - 1$ remaining cities of the TSP, we also have a drop-off point within TBRD and a customer location in direct vicinity. Any pair of customer location and drop-off point referring to the same city of TSP have no travel distance between each other. Note that if per problem definition drop-off points and customers cannot reside at the same location, this proof also works with a tiny distance among them. To keep this proof as simple as possible, we, however, assume that they may have the same location. The truck moves with a velocity of one distance unit per time unit and the speed of the robots is prohibitively slow, i.e., smaller than $\overline{d}/T$ where $\overline{d}$ denotes the smallest distance among any pair of distinct cities of the TSP instance. The deadlines and weights of all customers $j \in C$ are unanimously set to $d_j = T$ and $w_j = 1$, respectively. The question we ask is whether a feasible solution exists where all customers are timely supplied, i.e., $Z = 0$.

First, we show that each feasible solution to TBRD is also a feasible TSP solution. Due to the prohibitively low robot speed, each customer's robot has to be launched at the drop-off point directly next to the customer. Otherwise, even if launched from the closest alternative drop-off point the respective customer will inevitably be supplied too late. Furthermore, we have no robot depot and enough robots initially on board, so that each pair of drop-off point and customer, corresponding to a city of TSP, can be visited exactly once without robot replenishment. The delivery time of each customer is only determined by the travel time of the truck up to the respective customer. As all customers have to be timely supplied (even the last one), the total truck route through all pairs of drop-off point and customer may therefore not exceed T. Just the same tour through all corresponding cities is also a feasible TSP solution with a tour length smaller than or equal to T.

Finally, each feasible solution of TSP is also a YES-instance of TBRD. The truck route with TBRD just goes through the corresponding pairs of drop-off points and customers and the tour length is identical. $\square$

4 Solution methods

This section is dedicated to suited solution procedures of our TBRD problem. After presenting a MIP model in Section 4.1, we provide an efficient approach to determine an optimal assignment of customers to drop-off points and robot depots for a predetermined truck route in Section 4.2. Based on this, we introduce a tailor-made local search procedure in Section 4.3.

4.1 Mixed-integer model for TBRD

Applying the notation as summarized in Table 1, our mixed-integer program (TBRD-MIP) consists of objective function (1) and constraints (2) to (16).

$$\textbf{TBRD-MIP: } \text{Minimize } F(L,S,T,X,Z) = \sum_{k \in C} z_k \cdot w_k \tag{1}$$

C	set of customers
D	set of drop-off points
R	set of robot depots
d_k	deadline of customer k
w_k	weight of customer k
δ	number of robots initially loaded on the truck
γ, γ^e	initial position, final (dummy) position of the truck
K	the truck's maximum loading capacity for robots
$\vartheta^t_{v,v'}$, $\vartheta^r_{v,v'}$	travel time of truck and robots between two points v, v'
l_j	continuous variables: amount of robots on board at the departure from location j
$s_{i,j}$	binary variables: 1, if location j is the direct successor of location i; 0, otherwise
t_j	continuous variables: arrival time at location j
$x_{j,k}$	binary variables: 1, if customer k is supplied from location j; 0, otherwise
z_k	binary variables: 1, if customer k is supplied late; 0, otherwise

Table 1: Notation for MIP model

$$\sum_{j\in D\cup R\cup\{\gamma\}} x_{j,k} = 1 \qquad \forall k \in C \tag{2}$$

$$l_\gamma \leq \delta - \sum_{k\in C} x_{\gamma,k} \tag{3}$$

$$l_j \leq K + M\cdot(1-s_{i,j}) - \sum_{k\in C} x_{j,k} \qquad \forall i\in R;\ j\in D \tag{4}$$

$$l_j \leq l_i + M\cdot(1-s_{i,j}) - \sum_{k\in C} x_{j,k} \qquad \forall i\in D\cup\{\gamma\};\ j\in D\setminus\{i\} \tag{5}$$

$$t_\gamma = 0 \tag{6}$$

$$t_j \geq t_i - M\cdot(1-s_{i,j}) + \vartheta^t_{i,j} \qquad \forall i\in D\cup R\cup\{\gamma\};\ j\in D\cup R\setminus\{i\} \tag{7}$$

$$M\cdot z_k \geq t_j - M\cdot(1-x_{j,k}) + \vartheta^r_{j,k} - d_k \qquad \forall j\in D\cup R\cup\{\gamma\};\ k\in C \tag{8}$$

$$\sum_{j\in D\cup R\cup\{\gamma^e\}} s_{\gamma,j} \leq 1 \tag{9}$$

$$\sum_{i\in D\cup R\cup\{\gamma^e\}\setminus\{j\}} s_{j,i} = \sum_{i\in D\cup R\cup\{\gamma\}\setminus\{j\}} s_{i,j} \qquad \forall j\in D\cup R \tag{10}$$

$$\sum_{k\in C} x_{j,k} \leq M\cdot \sum_{i\in D\cup R\cup\{\gamma\}\setminus\{j\}} s_{i,j} \qquad \forall j\in D\cup R \tag{11}$$

$$\sum_{k\in C} x_{j,k} \geq \sum_{i\in D\cup R\cup\{\gamma\}\setminus\{j\}} s_{i,j} \qquad \forall j\in D \tag{12}$$

$$\sum_{k\in C} x_{j,k} + \sum_{i\in D} s_{j,i} \geq \sum_{i\in D\cup R\cup\{\gamma\}\setminus\{j\}} s_{i,j} \qquad \forall j\in R \tag{13}$$

$$s_{i,j}\in\{0,1\} \qquad \forall i\in D\cup R\cup\{\gamma\};\ j\in D\cup R\cup\{\gamma^e\}\setminus\{i\} \tag{14}$$

$$x_{i,k}, z_k\in\{0,1\} \qquad \forall i\in D\cup R\cup\{\gamma\};\ k\in C \tag{15}$$

$$l_j \geq 0 \qquad \forall j\in D\cup\{\gamma\} \tag{16}$$

Objective function (1) minimizes the weighted number of late deliveries which are determined by constraints (8). Constraints (2) ensure a one-to-one mapping between customers and locations where the robots were launched. The amount of robots on board is controlled by constraints (3) to (5), i.e., the initial loading (3) and the loading after leaving a drop-off point having a robot depot (4) or a drop-off point (5) as direct predecessor, respectively. While constraints (6) and (7) bound the arrival times at each location from below, subtour elimination constraints are established by (9) and (10). Constraints (11) guarantee that robots are launched from visited locations only. Note that constraints (12) and (13) are useful to eliminate redundant subtours. Specifically, these constraints enforce that drop-off points are part of the tour only if robots are launched from there and that depots are visited only if either robots are launched from there or a drop-off point is visited next. Finally, the domains of the variables are set by (14) to (16).

Solving TBRD instances with the proposed model requires the following points to be considered:

- TBRD-MIP can directly be applied if the truck's starting point γ is a drop-off point. However, in case γ is a robot depot, we have to execute the following preprocessing steps. First, we remove all customers $k \in C$ from our customer set which can be supplied in time from the current depot (i.e., $\vartheta^r_{\gamma,k} \leq d_k$). Then, we set $\delta = K$, so that the truck is loaded with robots to capacity and ensure that no customer is served from the starting point (i.e., $\sum_{k \in C} x_{\gamma,k} = 0$). Afterwards, we execute the model as described above.

- Our problem setting allows to visit locations, e.g., robot depots, more than once. To enable this property, we can simply duplicate locations. More precisely, if all deliveries are executed from a single drop-off point, $\lceil |C|/K \rceil$ visits are necessary. Thus, depending on the initial amount of loaded robots, $\lceil |C|/K \rceil + 1$ duplicates for each drop-off point are required. In addition, a depot is visited at most $\left\lceil \frac{|C|}{\lceil (K+1)/2 \rceil} \right\rceil$ times because in the worst case $\lceil (K+1)/2 \rceil$ robots are required at each drop-off point.

- To keep the model as simple as possible, it is desirable to reduce the number of duplicates to a minimum. First, we are allowed to remove all customers $k \in C$ who cannot be delivered in time. This is given whenever $\min \left\{ \vartheta^r_{\gamma,k}, \min \left\{ j \in D \cup R | \vartheta^t_{\gamma,j} + \vartheta^r_{j,k} \right\} \right\} > d_k$ holds. Next, if it is not possible to supply a set of customers $\bar{C} \subseteq C$ from drop-off point $j \in D$ in time, i.e., we have $\vartheta^t_{\gamma,j} + \vartheta^r_{j,k} > d_k$, $k \in \bar{C}$, then we duplicate drop-off point $j \in D$ only $\left\lceil \frac{|C \setminus \bar{C}|}{K} \right\rceil + 1$ times.

In spite of these considerations, however, it seems most likely that TBRD-MIP struggles with large instances of real-world size. Therefore, the following sections provide an efficient heuristic solution procedure. This heuristic repeatedly solves a subproblem where we derive the launching plan of robots for a given truck route. This subproblem is shown to be solvable to optimality in polynomial time first.

4.2 An efficient solution procedure for given truck routes

Once the truck route, i.e., a sequence of visits at drop-off points and robot depots starting in γ, is given, TBRD reduces to find a launching plan of robots. The optimal assignment of customer deliveries to the given stops of the truck can be determined by transferring the problem to a Hitchcock transportation problem, which is well-known to be efficiently solvable in polynomial time [13, 6]. Specifically, the transformation goes as follows.

Given a fixed truck route v, where v_i denotes the i-th stop, we first define $R(v)$ and $S(v)$ which represent the set of all robot depots within truck route v and the set of all subtours $S_k = \left\{v_{k_1}, \ldots, v_{k_{i_k}}\right\}$ along drop-off points ($v_{k_1}, \ldots, v_{k_{i_k}} \in D$) either between two consecutive depots ($v_{k_1-1}, v_{k_{i_k}+1} \in R$) or the beginning or end of truck route v ($k_1 = 1/k_{i_k} = |v|$), respectively. Given these definitions the supply and demand nodes as well as the transportation costs among them, which define the Hitchcock transportation problem, are constructed as follows:

- A supply node with supply quantity $a_i = |C|$, $i = 1, \ldots, |R(v)|$, is added for each depot $r_i \in R(v)$.
- A supply node with supply quantity $a_k = K$, $k = |R(v)| + 1, \ldots, |R(v)| + |S(v)|$, is added for each subtour $S_k \in S(v)$. In case $\gamma \notin R$, the supply quantity of the first subtour $S_{|R(v)|+1}$ equals the initial amount of loaded robots $a_{|R(v)|+1} = \delta$.
- Each customer $j \in C$ constitutes a demand node $\mathcal{C}_j$ with demand quantity $b_j = 1$.
- We introduce a single dummy demand node d with demand quantity $b_0 = \sum_{i=1}^{|R(v)|+|S(v)|} a_i - |C|$.
- Finally, we specify the transportation costs $c_{i,j}$ between supply and demand nodes. The costs from each supply node to the dummy demand node are set to zero. Costs $c_{i,j}$ between nodes referring to a depot $r_i \in R(v)$ at stop $v_{r(i)}$ and a customer $j \in C$ are set to zero if the supply is in time and w_j otherwise, i.e.,

$$c_{i,j} = \begin{cases} 0 & \text{, if } \sum_{l=1}^{r(i)-1} \vartheta^t_{v_l,v_{l+1}} + \vartheta^r_{v_{r(i)},j} \leq d_j \\ w_j & \text{, else.} \end{cases}$$

  Furthermore, costs $c_{k,j}$ between nodes referring to a subtour $S_k \in S(v)$ and a customer $j \in C$ are set zero if there is at least one drop-off point v_{k_i} ($i = 1, \ldots, i_k$) within S_k, so that the supply is in time and w_j otherwise, i.e.,

$$c_{k,j} = \begin{cases} 0 & \text{, if } \min_{i=k_1,\ldots,k_{i_k}} \left\{\sum_{l=1}^{i-1} \vartheta^t_{v_l,v_{l+1}} + \vartheta^r_{v_i,j}\right\} \leq d_j \\ w_j & \text{, else.} \end{cases}$$

Given this bipartite graph, which consists of supply and demand nodes, solving the Hitchcock transportation problem yields the optimal min-cost transportation plan in polynomial time. By considering which drop-off points and robot depots the supply nodes and which customers the demand nodes refer to, we can directly derive the optimal launching plan by minimizing the weighted sum of delayed deliveries (for the given truck route).

Example: Consider the example of Figure 3. On the left side, a problem instance of TBRD is given with a fixed route v, a truck capacity for $K = 1$ robot, and an initial amount of loaded robots $\delta = 1$. The two customers $C = \{1, 2\}$ have weights $w_1 = w_2 = 1$ and deadlines $d_1 = 3$ and $d_2 = 6$, respectively. The truck route $v = \{v_1, v_2, v_3\}$ contains a depot r_1 at

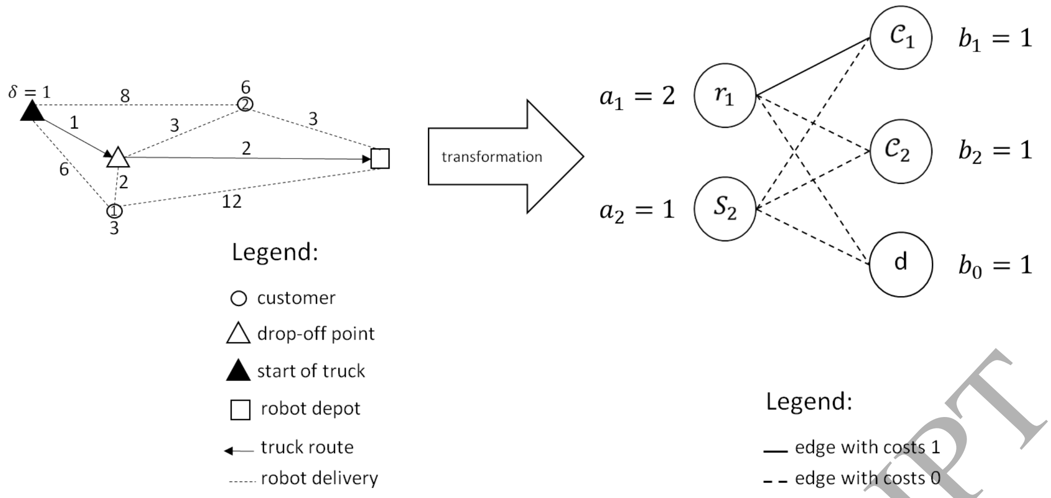

Figure 3: An instance of TBRD with a fixed route and the corresponding transportation problem

stop v_3 with supply $a_1 = |C| = 2$ and two drop-off points $v_1 = \gamma$ and v_2 which form the subtour $S_2 = \{v_1, v_2\}$ with supply $a_2 = \delta = 1$. The demand nodes (customers) have demands $b_1 = b_2 = 1$ and dummy node d has a demand of $b_0 = 1$. While customer 1 can be satisfied in time only if its corresponding robot is launched within subtour S_2 (at stop v_2), customer 2 can timely be supplied either from subtour S_2 (at stop v_2) or from depot r_1 (at stop v_3). Thus, we obtain the corresponding instance of the Hitchcock transportation problem given on the right side of Figure 3. Solving this problem reveals that the min-cost transportation plan (i.e., having an objective value of 0) corresponds to a launching plan where customer 1 is served from subtour S_2 and customer 2 from depot r_1.

An optimal assignment of customers to depots and drop-off points for a given truck route v may contain unnecessary drop-off points, where not even a single robot is launched, and depots, where robots are neither launched nor taken on board of the truck. If the triangle inequality for our travel times $\vartheta^t_{v,v'}$ and $\vartheta^r_{v,v'}$ holds, which we assume throughout this paper, then we can remove these superfluous stop from the truck route. According to the triangle inequality removing a stop will not enlarge the tour and therefore, will not increase the objective function value. First, we are allowed to remove each superfluous drop-off point where no robot is launched. Next, we remove each superfluous depot, which requires that no customer is served from this depot and the removal will not combine two subtours $S_k, S_{k+1} \in S(v)$ for $k > |R(v)| + 1$ ($k = |R(v)| + 1$) with more than K (δ) customers to be served.

4.3 A multi-start local search procedure

Based on the findings of Section 4.2, we propose a straightforward multi-start local search procedure. This simple yet efficient optimization approach can be subdivided into two steps: (1) Generation of a first tour employing priority rules and (2) ensuring feasibility of the tour by considering capacity constraints and improving the feasible solution via local search.

In the first step, a start solution is generated where we considerably simplify the problem by ignoring capacity constraints. The priority rules *PR1* and *PR2*, employed to generate a first tour of the truck, are solely based on travel times and deadlines. Note that we investigated a larger selection of rules, but these were the two working best. Our formal description is based on the notation introduced in Section 3.

- *PR1: Move to position with most satisfiable customers*

  1. Start from initial position $v_n = \gamma$ at time $t = 0$, such that $\pi_n = (\gamma, \emptyset)$ with $n = 1$. Initialize the set of already processed customers $\bar{C} = \emptyset$.
  2. Determine all customers $C^t_{v_n}$ satisfiable in time from current position v_n at the current point in time t as follows: $C^t_{v_n} = \{k \in C \setminus \bar{C} \mid t + \vartheta^r_{v_n,k} \leq d_k\}$. Update the current tuple $\pi_n = (v_n, C^t_{v_n})$ of customers served from the current stop and the set of already processed customers $\bar{C} = \bar{C} \cup C^t_{v_n}$. If all customers are processed, that is $\bar{C} = C$, terminate.
  3. Determine the number of (not yet processed) customers satisfiable in time for each depot and drop-off point $C_i^{t+\vartheta^t_{v_n,i}}$ $\forall i \in D \cup R$.
  4. If no customer can be satisfied in time anymore, that is $C_i^{t+\vartheta^t_{v_n,i}} = \emptyset$ $\forall i \in D \cup R$, visit the nearest depot, set the current tuple as follows $\pi_{n+1} = (\text{arg min}_{i \in R}\{\vartheta^t_{v_n,i}\}, C \setminus \bar{C})$, and terminate. Otherwise, visit the location with the highest number of satisfiable customers next $\pi_{n+1} = (\text{arg max}_{i \in D \cup R}\{|C_i^{t+\vartheta^t_{v_n,i}}|\}, \emptyset)$, $t = t + \vartheta^t_{v_n,v_{n+1}}$, $n = n + 1$ and go to step 2.

- *PR2: Move to position with highest urgency*

  1. Start from initial position $v_n = \gamma$ at time $t = 0$, such that $\pi_n = (\gamma, \emptyset)$ with $n = 1$. Initialize the set of already processed customers $\bar{C} = \emptyset$ and assign each customer to its nearest depot or drop-off point: $\tilde{i}_k = \text{arg min}_{i \in D \cup R}\{\vartheta^r_{i,k}\}$.
  2. (same as for *PR1*)
  3. For each depot and drop-off point $i \in D \cup R$ determine the set of customers timely satisfiable assigned to i, which have not been processed yet $\tilde{C}_i^{t+\vartheta^t_{v_n,i}} = \{k \in C \setminus \bar{C} \mid \tilde{i}_k = i \wedge t + \vartheta^t_{v_n,i} + \vartheta^r_{i,k} \leq d_k\}$.
  4. If no customer can be satisfied in time anymore, that is $\tilde{C}_i^{t+\vartheta^t_{v_n,i}} = \emptyset$ $\forall i \in D \cup R$, visit the nearest depot, set the current tuple as follows $\pi_{n+1} = (\text{arg min}_{i \in R}\{\vartheta^t_{v_n,i}\}, C \setminus \bar{C})$, and terminate. Otherwise, visit the location with the highest urgency, e.g., the smallest (positive) time budget to reach an assigned customer in time considering the velocity of truck and robot, $\pi_{n+1} = (\text{arg min}_{i \in D \cup R}\{\min_{k \in \tilde{C}_i^{t+\vartheta^t_{v_n,i}}}\{d_k - (t + \vartheta^t_{v_n,i}) - \vartheta^r_{i,k}\}\}, \emptyset)$, $t = t + \vartheta^t_{v_n,v_{n+1}}$, $n = n+1$ and go to step 2.

A solution generated in the first step may be infeasible as it can happen that either no depot is visited at all or more than K customers are assigned to the same subtour. In the second step we, therefore, fix infeasible solutions and try to further improve the obtained feasible solution by a local search procedure. In the following, we name the exact, polynomial time procedure introduced in Section 4.2 as function $HTP(v) = \pi$ and the solutions generated by priority rules *PR1* and *PR2* as π_{PR1} and π_{PR2}, respectively. The function $Z(\pi)$ returns the objective value of solution π. Applying this notation the second step proceeds as follows:

1. Try to improve π_{PR1} and π_{PR2} by applying HTP, such that $\pi_{PR1} = HTP(v_{PR1})$ and $\pi_{PR2} = HTP(v_{PR2})$.

2. Generate solution pools Π_{PR1} and Π_{PR2} using depot insertions. A depot insertion is possible at all index positions $n > 1$ of tour v, for which $v_{n-1}, v_n \in D$ hold. At each potential insertion position, we only consider the one depot that causes the least additional distance when added to the tour. In this way, we analyze all combinations of possible depot insertions, such that having a tour with m successively visited pairs of drop-off points 2^m tours are evaluated. Pretests have shown that HTP is fast enough, so that we can easily afford the evaluation of all additional tours. All these solutions are added to solutions pools Π_{PR1} or Π_{PR2}, provided that a solution $\pi' = (v', HTP(v'))$ has an objective value less than the respective initial solution π_{PR1} and π_{PR2} that is $Z(\pi') < Z(\pi_{PR1})\ \forall \pi' \in \Pi_{PR1}$ and $Z(\pi') < Z(\pi_{PR2})\ \forall \pi' \in \Pi_{PR2}$, respectively.

3. Evaluate solutions pools Π_{PR1} and Π_{PR2} multiple times in a parallelized manner using a multi-thread framework. Each pool is evaluated $\frac{\kappa_{rs}}{2}$ times and each evaluation starts with the initial solution $\pi_{local} = \pi_{PR1}$ or $\pi_{local} = \pi_{PR2}$, respectively. During an evaluation, the procedure iteratively picks a solution π' of the local copy of the selected pool by random choice. Afterwards, a local search procedure as described below, is performed for κ_{ls1} iterations starting from π' and π_{local} is updated according to the result of this search. π' and all solutions with objective values higher than or equal to π_{local} are then removed from the local copy of the selected pool. Once a pool contains no solution anymore, an additional run of local search is performed for κ_{ls2} iterations starting from π_{local} and the result of this search run is returned to the framework.

4. Return the best solution found during all pool evaluations.

The local search procedure performed multiple times during pool evaluation is based on the following neighborhoods:

- *N1*: Remove a random position of the tour if it is not the single robot depot of the tour (otherwise, feasibility is not guaranteed).
- *N2*: Insert a random depot or drop-off point into the tour at a random position.
- *N3*: Swap two randomly chosen positions of the tour.
- *N4*: Select a position of the tour. If the selection is a depot, do nothing. Otherwise, insert the depot closest to the drop-off point right before the selected position into the tour.

The resulting tour v' is transformed into a full solution of our TBRD $(v', HTP(v'))$. If the generated solution is better than the current one, it is accepted and set as the starting point of preceding search steps. Otherwise, it is rejected and another neighbor of the current solution is evaluated. The search is executed until the given number of iterations has been performed. The whole multi-start local search procedure is summarized in Algorithm 1.

Algorithm 1: Multi-start local search procedure

```
1  solution $\pi_{best} = null$ ;            // best solution found during all restarts
2  $\pi_{PR1}$ = HTP(solution found by PR1);
3  $\Pi_{PR1}$ = set of tours generated by depot insertions into $\pi_{PR1}$;
4  $\pi_{PR2}$ = HTP(solution found by PR2);
5  $\Pi_{PR2}$ = set of tours generated by depot insertions into $\pi_{PR2}$;
6  for ($i = 1; i \leq \kappa_{rs}; i$ += $1$) do
7      if ($i \leq \frac{\kappa_{rs}}{2}$) then
8          $\pi_{local} = \pi_{PR1}$; $\Pi' = \Pi_{PR1}$.Clone();
9      else
10         $\pi_{local} = \pi_{PR2}$; $\Pi' = \Pi_{PR2}$.Clone();
11     while ($|\Pi'| > 0$) do
12         $\pi'$ = randomly selected solution in $\Pi'$;
13         $\pi_{local}$ = LS(#iterations: $\kappa_{ls1}$, start solution: $\pi'$);    // start local search
14         $\Pi' = \Pi' \setminus \{\pi'\} \cup \{\pi'' \in \Pi' | Z(\pi'') \geq Z(\pi_{local})\}$;
15     $\pi_{local}$ = LS(#iterations: $\kappa_{ls2}$, start solution: $\pi_{local}$);    // start local search
16     if ($\pi_{best} = null$ OR $Z(\pi_{local}) \leq Z(\pi_{best})$) then
17         $\pi_{best} = \pi_{local}$ ;          // update best solution during all restarts
18 return $\pi_{best}$ ;                     // return best solution during all restarts
```

5 Performance of algorithms

In this section, we test the performance of our solution methods. Since no established testbed is available for our TBRD, we first elaborate how our instances have been generated (see Section 5.1). Afterwards in Section 5.2, we benchmark the performance results of our heuristic solution procedure with a standard solver solving TBRD-MIP.

All computations have been executed on a 64-bit PC with an Intel Core i7-6700K CPU (4 x 4.0 GHz), 64 GB main memory, and Windows 7 Enterprise. The procedures have been implemented using C# (Visual Studio 2015) and off-the-shelf solver Gurobi (version 7.0.2) has been applied for solving the MIP models.

5.1 Instance generation

We generate two different sizes of test instances. The dataset labeled as *small* consists of instances which can be solved to proven optimality in a sufficient percentage of cases employing a standard solver, while the dataset labeled *large* represents instances of real-world size. The parameters handed over to our data generator are presented in Table 2. The procedure of instance generation is summarized in the following.

Layout: First, we generate a square with side length w. The square is divided into a grid of squares of side length 1/6 kilometers (about 1/10 miles). A reasonable policy when planning a depot network is to equally spread the facilities in the urban area to be serviced. Therefore, we allocate the given number of robot depots in an equidistant manner within our grid. Afterward, we randomly scatter drop-off points and customers, making sure that each point is assigned only once. Finally, a randomly chosen depot or drop-off point is set as the initial truck position, such that $\gamma \in D \cup R$ (with $\delta = K$).

symbol	description	small	large
w	side length of main square	2 km (1.3 miles)	5 km (3.1 miles)
$\|R\|$	number of robot depots	4	16
$\|D\|$	number of drop-off points	6	30
$\|C\|$	number of customers	6	40
$[\rho_{min}; \rho_{max}]$	deadline factor interval (named {*tight*, *wide*})	{[2, 4]; [3; 5]}	{[4; 12]; [4; 15]}
$[w_{min}; w_{max}]$	weights interval (named {*homo*, *hetero*})	{[1; 1]; [1; 3]}	
K	trucks robot capacity	2	8
	speed truck/robot	30 $\frac{\text{km}}{\text{h}}$ (18.6 mph) / 5 $\frac{\text{km}}{\text{h}}$ (3.1 mph)	

Table 2: Parameter values for instance generation

Distances and travel times: We use the Euclidean metric to compute distances between points. Based on these distances and given truck and robot speeds, travel times are calculated. Note that using the Euclidean metric ensures that the triangle inequality is never violated considering a single mode of transportation, e.g., trucks or robots.

Customer deadlines: Customer deadlines are generated on the basis of the travel times. First, we determine the minimum travel time to each customer $k \in C$ by $\vartheta_{min}(k) = \min_{j \in D \cup R}\{\vartheta^t_{\gamma,j} + \vartheta^r_{j,k}\}$. Second, we draw a random number r_k from the interval $(0, 1]$. Then we determine the deadline of customer k as $d_k = \vartheta_{min}(k) \cdot (\rho_{min} + (\rho_{max} - \rho_{min}) \cdot r_k)$.

Weights: For each customer $k \in C$, we draw a weight w_k randomly from interval $[w_{min}; w_{max}]$ applying a uniform distribution.

We generate ten geographical configurations per dataset. For each of these layouts, we combine different deadlines and weights in a full factorial approach and repeat instance generation five times per parameter setting. In total, we receive 200 small and 200 large instances in each dataset.

5.2 Computational study of algorithm performance

In this section, we evaluate the computational performance of our solution methods. Table 3 summarizes the results for our small dataset. For both solution methods, i.e., our multi-start local search procedure presented in Section 4.3 and standard solver Gurobi solving TBRD-MIP of Section 4.1, we report the number of best solutions found (best), e.g. how many times the solution approach was able to find the minimal known objective value, the average relative gap to the optimal solution for all instances a proven optimum is available (gap), as well as the average CPU-time in seconds (sec) required by the respective solution method per instance of the whole dataset. Additionally, we report the number of proven optima found by Gurobi (prov. opt.) and the number of optima found (but not proven) by the heuristic procedure (opt.). Gurobi is applied with a time limit of 30 minutes. Based on the results presented in Table 3, the following conclusions can be drawn:

Applying standard solver Gurobi solving TBRD-MIP, obviously, becomes easier if customer deadlines are rather generously set and heterogeneous weights are chosen. However, a significant correlation between hetero- and homogeneous weights and the performance of Gurobi

		Gurobi (time limit=1800sec)			Heuristic (κ_{rs}=100;κ_{ls1}=1000;κ_{ls2}=500)			
deadlines	weights	best	prov. opt.	sec	best	opt.	gap opt.	sec
thight	homo	50	36	578.38	49	35	2.78%	0.77
	hetero	50	37	513.88	50	37	0.00%	0.85
wide	homo	50	43	278.60	50	43	0.00%	0.99
	hetero	50	44	286.57	50	44	0.00%	1.04
total/average		200	160	414.36	199	159	0.63%	0.91

Table 3: Solution performance for the small dataset

is only available for the large dataset. For the small dataset, our heuristic procedure is able to find all but one proven optima Gurobi found within its time limit. Indeed, except for this single outlier, the heuristic solutions are all of the same quality than the ones found by Gurobi, proven optimum or not. Please note that in the single case our heuristic approach was not able to achieve a best known solution, two instead of one customer are delivered late, which is the lowest deviation possible. Nonetheless, the average time consumption of the heuristic is just 0.91 seconds. In comparison, Gurobi's mean computational time amounts to 414.36 seconds.

		Gurobi (time limit=1800sec)					Heuristic (κ_{rs}=80;κ_{ls1}=4000;κ_{ls2}=500)				
deadlines	weights	best	prov. opt.	gap best	gap LB	sec	best	opt.	gap best	gap LB	sec
tight	homo	7	6	248.60%	658.80%	1614.06	50	6	0.00%	117.60%	53.92
	hetero	13	11	308.20%	758.60%	1496.51	50	11	0.00%	110.30%	55.04
wide	homo	10	9	316.00%	550.00%	1530.82	50	9	0.00%	56.30%	39.45
	hetero	20	19	258.10%	470.40%	1245.04	50	19	0.00%	59.30%	43.39
total/average		50	45	283.10%	614.60%	1471.61	200	45	0.00%	86.50%	47.95

Table 4: Solution performance for the large dataset

For the large dataset (see Table 4), a substantial percentage of instances could not be solved to optimality. Therefore, we compare solution values to the best solution available per instance (gap best) and to a lower bound (gap LB). Please note that we determined lower bounds by combining an LP-relaxation and Gurobi's lower bound after 1800 seconds of computation time. For all 200 instances, Gurobi's bound was tighter and therefore employed for evaluation. However, the bound does not seem to be tight enough for a qualified evaluation of the solution procedures. For the large dataset the gap to the bound amounts to nearly 90% for our heuristic procedure and even when applied to the small dataset the gap to the best found solution, of which 80% are proven optima, is about 11%.

When comparing to the best solution found, our heuristic is successful for all instances. Gurobi obtains a solution of the same quality in 50 out of 200 cases (25%). For 45 of these 50 instances Gurobi proves optimality. However, Gurobi's solution values for the remaining instances are far behind those found by our heuristic, so that the average gap of Gurobi's solutions amounts to almost 290%. Moreover, Gurobi's average CPU time is 1470 seconds (time limit 1800 seconds) while the heuristic's mean CPU time amounts to merely 48 seconds.

It can be concluded that our heuristic seems well suited to solve even larger instances of TRBD in reasonable time at a good quality.

symbol	parameter	value
w	side length of main square	4 km (2.5 miles)
$\|R\|$	number of depots	1, 2, 4, **12**, 24
$\|D\|$	number of drop-off points	6, 12, **24**, 36
$\|C\|$	number of customers	25
$[\rho_{min}; \rho_{max}]$	deadline factor interval	[4; 10]
$[w_{min}; w_{max}]$	weights interval	[1; 1]
K	trucks robot capacity	3, 5, **8**, 10
μ_r	handling time per robot [sec]	{ 20, **40**, 60 }
μ_t	handling time per truck delivery [sec]	{ 60, **120**, 180 }
	speed truck [km/h(mph)]	30(18.6)
	speed robot [km/h(mph)]	3.5(2.2), **5(3.1)**, 6(3.7), 10(6.2), 15(9.3), 30(18.6)

Table 5: Parameter values used in managerial study

6 Managerial aspects

Beyond the pure computational performance, this section is dedicated to managerial aspects. Specifically, we report on the results of a sensitivity analysis in Section 6.1. Here, we explore how the network density of robot depots and drop-off points as well as the velocity of the robots and the truck's capacity for robots impact the timeliness of deliveries. Furthermore, in Section 6.2 we benchmark our delivery policy where the robots return to decentralized robot depots with two alternative policies. One alternative is that the truck waits for the return of the robots, so that they are taken on board again and no decentralized robot depots are required. Our other competitor is a traditional delivery by truck without the assistance of autonomous robots. In this way, some decision support on the right delivery policy is provided.

The computational tests of this section are based on a basic dataset of 10 instances generated using the procedure described in Section 5.1. The parameters employed are listed in Table 5. Note that default values are given in bold. If not explicitly specified otherwise, these default values are applied. Whenever parameters are changed, the standard setting of a basic instance remains identical and only properties influenced by the new parameter setting are recomputed, such that 10 instances derived from the basic dataset are solved for each setting investigated. Consequently, the whole dataset solved in Section 6.1 contains a total of 200 instances, where 190 are obtained from the basic dataset. The study in Section 6.2 involves another 90 instances, which are each solved twice by differing MIP models, e.g., TBRD-MIP and R2T-MIP or MinTruck-MIP, respectively.

Parameters μ_r and μ_t representing fixed handling times per robot and truck delivery, respectively, are included in the instance data as described in Section 3. These fixed times required for loading a robot with a shipment and handing over a shipment by a deliveryman are added to the travel time matrices. Further note that we assume the waiting and handling times at each customer required to unload a robot being equal to the fixed robot loading time μ_r when applying the policy where the robots have to return to the truck. Thus, twice μ_r adds to the driving time of the robots after their launch from the truck under this policy.

All instances are solved employing Gurobi with a time-limit of one hour per instance. Please note that, with only a few randomly scattered exceptions, all instances could be solved to optimality.

6.1 Impact of network density, robot speed, and truck capacity

The performance of the truck-based robot delivery concept heavily depends on the following four factors, whose impact is explored in a sensitivity analysis:

(a) The truck has to visit a decentralized robot depot from time to time in order to take another batch of robots on board. Clearly, the denser the network of robot depots, the shorter the detours of the truck to visit them. To explore the impact of the depot density on the delivery performance, we place different number $|R|$ of depots in our grid in the equidistant manner defined in Section 5.1.

(b) Loading the robots with customer shipments may take some minutes, especially, if a complete batch of K robots is to be handled. Thus, stopping the truck, loading the robots and launching them safely may not be possible just anywhere, so that regulatory authorities may restrict robot handling to specifically designated parking spaces. Consequently, it seems interesting to also explore the impact of the network density of potential drop-off points $|D|$.

(c) Due to safety reasons robots driving along sideways are bound to pedestrian speed. However, there may still be some flexibility whether robots are allowed to rather travel at 4.5 or at 6 km/h (2.8 to 3.7 mph). Therefore, we also explore the impact of the robot speed on the delivery performance.

(d) The limited capacity of the truck has to be partitioned among the space reserved for shipments and that for robots. Thus, we also explore the impact of the truck's maximum loading capacity for robots K on the solution performance.

Delivery performance of our truck-based robot delivery concept is measured by the mean number of customers supplied late (averaged over all 10 instances per data-point) and the results of our sensitivity analysis are depicted in Figure 4. The following conclusions can be drawn from these results:

(a) As expected, a denser network of robot depots increases delivery performance. However, the good news is that marginal returns of additional depots diminish the more depots are added. Therefore, not too much investment costs for decentralized garages where robots can be parked and recharged need to be spent.

(b) An analogous impact can be observed for the density of drop-off points. In our tests, more than 1.5 drop-off points per square kilometer (about 0.4 square mile) could not further increase delivery performance. Within such a relatively large area it should be easily possible to find secure launching points for robots in any city center.

(c) Also, the robot speed shows great influence on delivery performance. The gray interval marks the range of pedestrian speeds, i.e., between 4.5 and 6 km/h (2.8 to 3.7 mph), which is often attributed as walking speed. Our results reveal that choosing either the lower or higher range of this interval is rather critical. For instance, if the speed is increased from 5 km/h to 6 km/h in our tests, the number of delayed deliveries can be reduced by 75% from 0.4 to 0.1 on average. Due to the decreasing marginal utility, a further increase of speed is less effective.

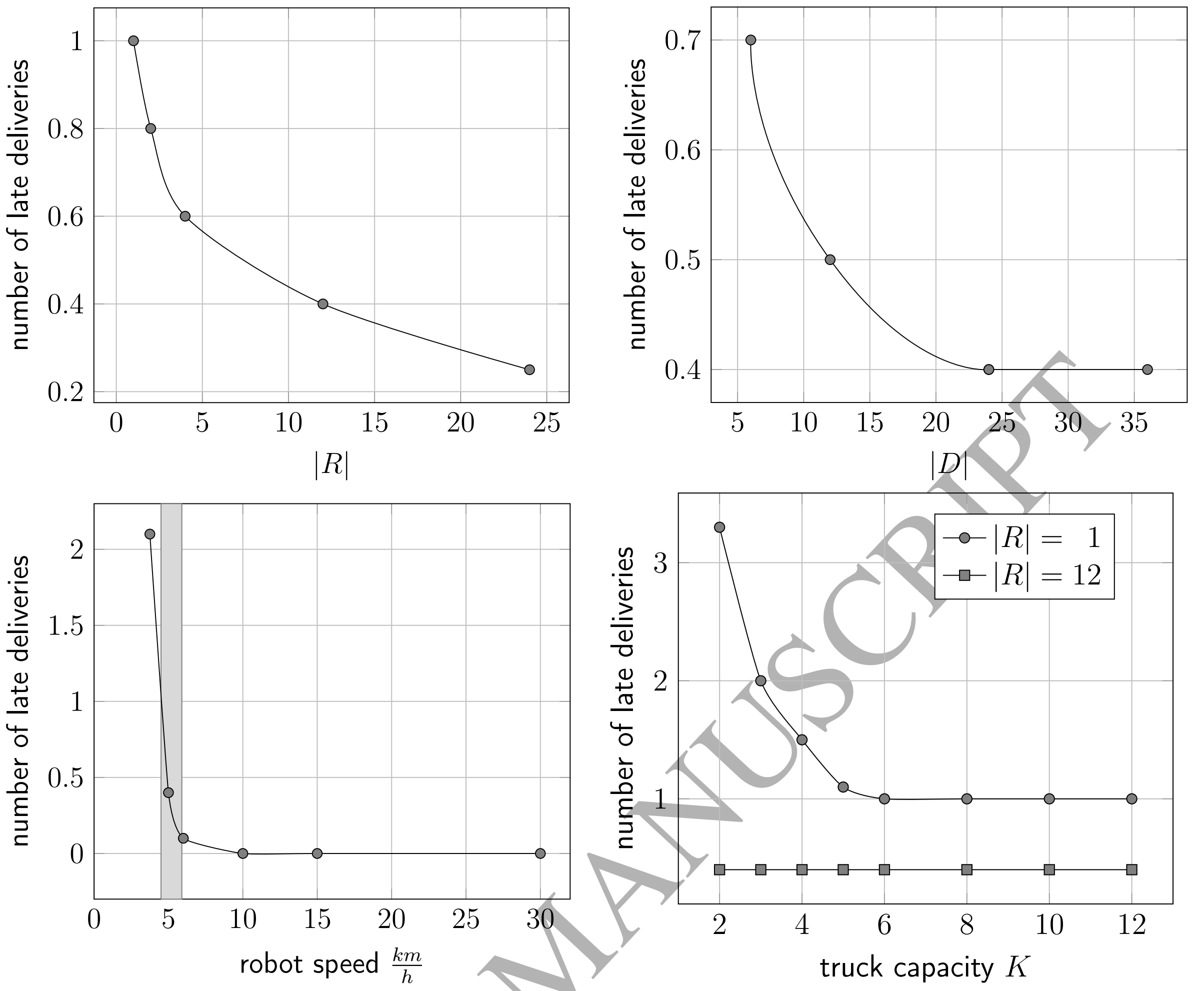

Figure 4: Sensitivity analysis: Impact of network density of depots (top left) and drop-off points (top right) as well as robot speed (bottom left) and truck capacity (bottom right)

(d) The last impact factor investigated is the truck capacity K. It shows to be highly interdependent especially with the depot density. Having a moderate depot density of 0.75 depots per square kilometer (about 1.94 per square mile), i.e., 12 depots in total, the solution quality shows astonishingly robust to changing truck capacity, because depots are always close by and, thus, repeatedly visited anyway. Having a very low density of only 0.06 depots per square kilometer (about 0.16 per square mile), i.e., just a single depot, truck capacity K becomes a critical factor and is able to reduce the number of delayed customers by about 70%. However, our tests reveal that even in the latter case a decreasing marginal utility of additional capacity is observable. Consequently, a capacity for $K = 8$ robots per truck, which is currently planned in the case of Mercedes-Benz Vans and Starship Technologies, rather seems at the upper limit of reasonable capacities. Consequently, it might be advisable to reduce truck capacity for robots in favor of a higher capacity for shipments, especially if quite a few robot depots are available.

Our results show that, especially, weighing off the investment costs for a denser network of robot depots and the security gains by slower robots need to be carefully traded off against the delivery performance. A suitable truck capacity has to be chosen based on several other setup factors, such as the depot density. In the next section, we explore whether or not it is indeed a good idea to let the robots return to decentralized depots.

6.2 Benchmarking different operational policies

To explore the value of last-mile deliveries with truck-based autonomous robots, we benchmark three alternative organizational settings for delivering goods to customers:

- First, we have our truck-based robot delivery policy where robots are launched from the truck but return autonomously to decentralized robot depots. We dub this policy *robot-to-depot* or *R2D* for short. Using this policy, truck stops remain pretty short because robots only have to be loaded and launched by the driver. On an operational level, these brief stops come for the price of additional detours towards robot depots. Additionally, we have the investment and operational costs for the decentralized robot depots. To evaluate the performance of the R2D policy, we solve TBRD-MIP with standard solver Gurobi and record the average number of delayed customers.
- To save these costs, the truck-based robot delivery concept can be altered. If the truck waits for the return of launched robots, they are taken on board again and can be launched at the same or another truck stop without the need for robot depots. We dub this policy *robot-to-truck* or *R2T* for short. On the negative side, truck stops become much longer under this policy as the truck has to wait for the last robot to return. Additionally, robot delays due to longer service times or unsuccessful waiting at customer locations have to be considered. To determine the average number of delayed customers for the R2T policy, we solve the MIP model presented in Appendix A.
- Finally, we also benchmark the status quo which is a home-attended truck-based delivery without further assistance of autonomous robots. Here, each customer is supplied directly by the driver of the truck, so that customer locations are directly accessed and unnecessary stops at drop-off points or robot depots are avoided. Nonetheless, additional driving time to each single customer is required and each single truck stop takes comparatively long time. The deliveryman has to find a suited parking space, walk the remaining distance, wait for the customer, and then return to the vehicle. To benchmark the traditional truck delivery with our R2D-policy, we determine the minimum truck fleet that is required to also reach the service level of R2D. Specifically, we hand over the number Θ of delayed customers determined when solving TBRD-MIP for an instance and determine the minimum fleet size of trucks, so that at most Θ customers are supplied late by solving the MIP model presented in Appendix B.

Clearly, the performance of each policy is heavily dependent on the service times, e.g., how long it takes a robot or the driver to hand over a shipment, so that we benchmark the three delivery policies and quantify how their performance is impacted by different delivery times. Specifically, we vary parameters μ_r and μ_t which define the delivery time of the robot and the truck, respectively. Loading a robot at a drop-off point takes μ_r time units which – in addition to the robot's driving time between drop-off point and customer location – is the only delivery time to be considered under the R2D policy. For the R2T policy, we have the time of the R2D policy for loading the robot and driving to the customer and have to add the unloaded time by the customer and additionally the driving time back to the truck. Unloading the robot by a customer is assumed to take another μ_r time units. For the traditional truck-based delivery, we presuppose the complete delivery process, i.e., parking, walking to the customer location, handing over the shipment, and returning to the truck, to take μ_t time units. The results of the three delivery policies and their dependency of different delivery times are depicted in Figure 5. Each data-point represents the mean value of the respective objective function, averaged over all 10 instances of the represented scenario. From these results, the following conclusions can be drawn:

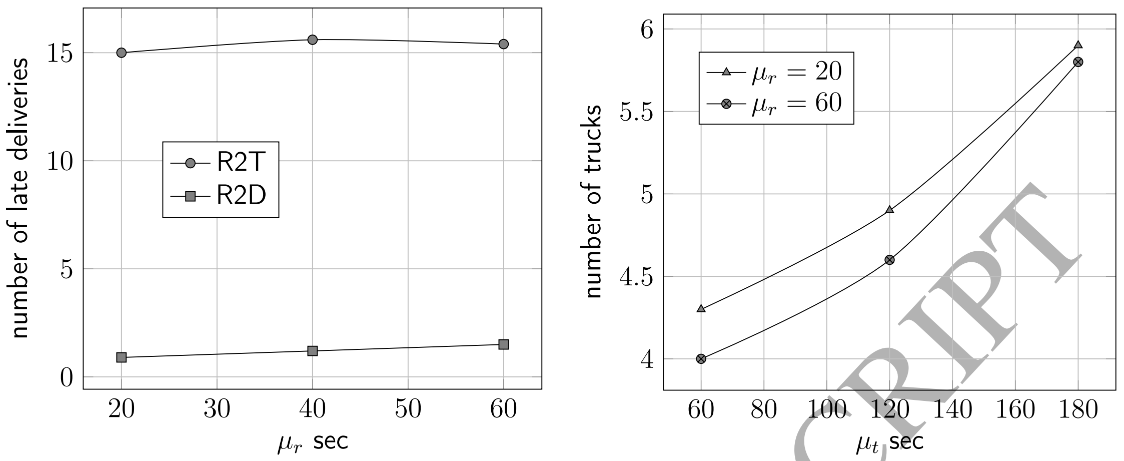

Figure 5: Comparison of policies for varying delivery times for robots (μ_r) and truck (μ_t)

- First, it can be concluded that our R2D policy where the robots return to depots considerably outperforms the R2T policy where the truck has to wait for the robots' return. The latter policy leads to significantly more dissatisfied customers whose shipments are supplied late. The huge gap between both policies as depicted on the left of Figure 5 remains rather unaffected by larger delivery times μ_r. This is not that astounding because both policies equally suffer if loading (and unloading) of robots takes more time.
- Moreover, in our setting, also, the traditional truck-based delivery is clearly outperformed by the R2D policy (see Figure 5, right). To reach the same service level as R2D, which means that at most the same number of customers are supplied late, at least three additional trucks are required if no autonomous robots are deployed. Even more trucks are required if the loading time μ_r of the robots is short and the delivery time μ_t of the deliveryman is large.

These results are a bit biased in favor of our R2D because we assume that robots are no bottleneck and the truck can always take the desired number of robots on board in each robot depot. To realize this assumption in the real world, this would require additional investment into a sufficiently sized robot fleet. If – to avoid the investment costs – our premise is not given in reality and robots may become scarce, then the advantage of R2D decreases. However, the large advantage of R2D in our computational study compared to R2T and traditional truck-based deliveries is, nonetheless, a strong indicator that R2D is a promising delivery policy when applying a truck-based robot delivery.

7 Conclusion

This paper investigates an innovative last-mile concept where autonomous robots are launched from trucks to deliver shipments towards customers. After delivery, the robots return to decentralized robot depots where a delivery truck can take them on board again. This paper focuses on the scheduling of the delivery truck and the launching of robots along the truck route, such that customers are timely delivered. We formulate the resulting scheduling problem, prove computational complexity, and derive an efficient heuristic solution procedure. Our computational study shows that the decentralized robot depots greatly contribute to an efficient delivery process. If the truck has to wait for the return of the robots instead, this leads to considerable waiting times for the truck and much more unsatisfied customers. We also benchmark our novel delivery concept with traditional attended-home delivery by truck. Our computational study reveals that the truck fleet can considerably be reduced if autonomous robots support the delivery process.

There are quite a few opportunities to extend our first findings on the truck-based robot delivery concept. First, our study only considers a single truck where the set of customers to be serviced is already given. On a preceding planning level, however, all current customer orders have to be partitioned among multiple vehicles. Deriving a larger solution framework, where our solution procedure for the single truck case could be a valid building block, would be a challenging task. Furthermore, our (simplifying) assumption that robots are no bottleneck resource and the truck can always load the demanded number of them on board in each robot depot could be reinvestigated. If robots are scarce, each individual robot and its tour towards customers and back to depots has to be tracked in detail, which adds a lot of complexity to the problem. Furthermore, recent developments in smart-lock technologies (e.g., Amazon Key), make unattended home delivery with autonomous ground vehicles a tangible possibility. The implications of unattended home deliveries on the problem and our heuristics are sketched in Appendix C, but require further research effort. Decision support in these directions could help to understand the economics of the truck-based robot delivery concept and whether it is indeed a promising concept for future last-mile deliveries.

References

[1] Afrati, F.; Cosmadakis, S.; Papadimitriou, C.H.; Papageorgiou, G.; Papakostantinou, N. (1986): The complexity of the travelling repairman problem. Informatique théorique et Applications 20, 79-87.

[2] Agatz, N.; Bouman, P.; Schmidt, M. (2016): Optimization approaches for the traveling salesman problem with drone. Working Paper Rotterdam School of Management.

[3] Arkin, E.M.; Hassin, R. (1994): Approximation algorithms for the geometric covering salesman problem. Discrete Applied Mathematics 55, 197-218.

[4] Arslan, A.; Agatz, N.; Kroon, L.G.; Zuidwijk, R.A. (2016): Crowdsourced delivery: A dynamic pickup and delivery problem with ad-hoc drivers. Working Paper Rotterdam School of Management.

[5] Boysen, N.; Briskorn, D.; Fedtke, S.; Schwerdfeger, S. (2017): Drone delivery from trucks: Drone scheduling for given truck routes. Working Paper Friedrich-Schiller-University Jena.

[6] Brenner, U. (2008): A faster polynomial algorithm for the unbalanced Hitchcock transportation problem. Operations Research Letters 36, 408-413.

[7] Current, J.R.; Schilling, D.A. (1989): The covering salesman problem. Transportation Science 23, 208-213.

[8] Daimler (2017): Vans & Robots Paketbote 2.0. https://www.daimler.com/innovation/specials/future-transportation-vans/paketbote-2-0.html.

[9] Drexl, M. (2012): Synchronization in vehicle routing - A survey of VRPs with multiple synchronization constraints. Transportation Science 46, 297-316.

[10] Garey, M.R.; Johnson, D.S. (1979): Computers and intractability: A guide to the theory of NP-completeness. New York, Freeman.

[11] Gouveia, L.; Voss, S. (1995): A classification of formulations for the (time-dependent) traveling salesman problem. European Journal of Operational Research 83, 69-82.

[12] Heilporn, G.; Cordeau, J.F.; Laporte, G. (2010): The delivery man problem with time windows. Discrete Optimization 7, 269-282.

[13] Kleinschmidt, P.; Schannath, H. (1995): A strongly polynomial algorithm for the transportation problem. Mathematical Programming 68, 1-13.

[14] Lin, S.W.; Vincent, F.Y.; Chou, S.Y. (2009): Solving the truck and trailer routing problem based on a simulated annealing heuristic. Computers & Operations Research 36, 1683-1692.

[15] Miller, H.J. (2013): Beyond sharing: Cultivating cooperative transportation systems through geographic information science. Journal of Transport Geography 31, 296-308.

[16] Murray, C.C.; Chu, A.G. (2015): The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C 54, 86-109.

[17] Pelletier, S.; Jabali, O.; Laporte, G. (2016): 50th anniversary invited article – Goods distribution with electric vehicles: Review and research perspectives. Transportation Science 50, 3-22.

[18] Reyes, D.; Savelsbergh, M.W.P.; Toriello, A. (2017): Vehicle routing with roaming delivery locations. Transportation Research Part C 80, 71-91.

[19] Savelsbergh, M.W.P.; Van Woensel, T. (2016): 50th anniversary invited article – City Logistics: Challenges and opportunities. Transportation Science 50, 579-590.

[20] Silva, M.M.; Subramanian, A.; Vidal, T.; Ochi, L.S. (2012): A simple and effective metaheuristic for the minimum latency problem. European Journal of Operational Research 221, 513-520.

[21] Speranza, M.G. (2018): Trends in transportation and logistics. European Journal of Operational Research 264, 830-836.

[22] Wang, X.; Poikonen, S.; Golden, B. (2017): The vehicle routing problem with drones: several worst-case results. Optimization Letters 11, 679-697.

Appendix A: A MIP model if the truck waits for the robots' return

To compare the usage of decentralized depots, i.e., the R2D policy, with the concept of the truck waiting for the return of launched robots, i.e., the R2T policy, we introduce a MIP model, which minimizes the weighted number of late deliveries under the R2T policy. Within this setting, there is no need for robot depots, so that we do not differentiate between depots and drop-off points, i.e., we set $\bar{D} = D \cup R$. The new setting allows us to relinquish all l variables of model TBRD-MIP provided in Section 4.1. Instead, we introduce variable C_k representing the delivery time at customer location k. Furthermore, we have to keep track which robot serves a customer, so that we have to adapt our x variables (see Table 6). Our modified model R2T-MIP consists of objective function (17) and constraints (18) to (31).

$\bar{R}$	set of robots (i.e. $\bar{R} = \{1, \dots, K\}$)
C_k	continuous variables: delivery time at location k
$x_{j,k,r}$	binary variables: 1, if customer k is supplied from location j by robot r; 0, otherwise
y_{k_1,k_2}	binary variables: 1, if customer k_1 is supplied before customer k_2 from the same location by the same robot; 0, otherwise

Table 6: Additional and altered notation for R2T-MIP

$$\textbf{R2T-MIP: } \text{Minimize } F(S,T,X,Y,Z) = \sum_{k \in C} z_k \cdot w_k \tag{17}$$

$$\sum_{r\in\bar{R}}\sum_{j\in\bar{D}\cup\{\gamma\}} x_{j,k,r} = 1 \qquad \forall\, k \in C \tag{18}$$

$$\sum_{k\in C} x_{j,k,r}\cdot\vartheta^r_{j,k} \geq \sum_{k\in C} x_{j,k,r-1}\cdot\vartheta^r_{j,k} \qquad \forall\, j\in\bar{D}\cup\{\gamma\}\,;\ r\in\bar{R}\setminus\{1\} \tag{19}$$

$$C_j \geq 2\cdot\sum_{k\in C} x_{j,k,K}\cdot\vartheta^r_{j,k} \qquad \forall\, j\in\bar{D}\cup\{\gamma\} \tag{20}$$

$$t_\gamma = 0 \tag{21}$$

$$t_j \geq t_i - M\cdot(1-s_{i,j}) + \vartheta^t_{i,j} + C_i \qquad \forall\, i\in\bar{D}\cup\{\gamma\}\,;\ j\in\bar{D}\setminus\{i\} \tag{22}$$

$$x_{j,k_1,r} + x_{j,k_2,r} \leq 1 + y_{k_1,k_2} + y_{k_2,k_1} \qquad \forall\, j\in\bar{D}\cup\{\gamma\}\,;\ k_1<k_2\in C;\ r\in\bar{R} \tag{23}$$

$$\begin{aligned} M\cdot z_k \geq\ & t_j - M\cdot(1-\sum_{r\in\bar{R}} x_{j,k,r}) - d_k \\ & + 2\cdot\sum_{k_1\in C\setminus\{k\}} \vartheta^r_{j,k_1}\cdot y_{k_1,k} + \vartheta^r_{j,k} \qquad \forall\, j\in\bar{D}\cup\{\gamma\}\,;\ k\in C \end{aligned} \tag{24}$$

$$\sum_{j\in\bar{D}\cup\{\gamma^e\}} s_{\gamma,j} \leq 1 \tag{25}$$

$$\sum_{i\in\bar{D}\cup\{\gamma^e\}\setminus\{j\}} s_{j,i} = \sum_{i\in\bar{D}\cup\{\gamma\}\setminus\{j\}} s_{i,j} \qquad \forall\, j\in\bar{D} \tag{26}$$

$$\sum_{r\in\bar{R}}\sum_{k\in C} x_{j,k,r} \leq M\cdot\sum_{i\in\bar{D}\cup\{\gamma\}\setminus\{j\}} s_{i,j} \qquad \forall\, j\in\bar{D} \tag{27}$$

$$\sum_{r\in\bar{R}}\sum_{k\in C} x_{j,k,r} \geq \sum_{i\in\bar{D}\cup\{\gamma\}\setminus\{j\}} s_{i,j} \qquad \forall\, j\in\bar{D} \tag{28}$$

$$s_{i,j}\in\{0,1\} \qquad \forall\, i\in\bar{D}\cup\{\gamma\}\,;\ j\in\bar{D}\cup\{\gamma^e\}\setminus\{i\} \tag{29}$$

$$x_{i,k,r},\, z_k\in\{0,1\} \qquad \forall\, i\in\bar{D}\cup\{\gamma\}\,;\ k\in C;\ r\in\bar{R} \tag{30}$$

$$y_{k_1,k_2}\in\{0,1\} \qquad \forall\, k_1\in C;\ k_2\in C\setminus\{k_1\} \tag{31}$$

Objective function (17) as well as constraints (18), (21) to (28) and (29) to (30) directly correspond to objective function (1) as well as constraints (2), (6) to (12) and (14) to (15) of TBRD-MIP described in Section 4.1. Furthermore, constraints (19) and (20) determine the delivery time at each location. Note that (19) eliminates significant symmetry from the model. According to the deadline of each customer it might become necessary to visit a location more than once. To comply with that requirement we can duplicate each location $\left\lceil\frac{|C|}{2}\right\rceil$ times. At worst, the truck has to drive back and forth between two drop-off points and, each time, merely launches a single robot.

Appendix B: A MIP model for traditional truck-based delivery

To benchmark our R2D policy, we compare the robot-based last-mile delivery with the traditional usage of trucks where no autonomous robot is available. To benchmark these policies we introduce the following MIP model, which determines the minimum number of trucks needed under the classical policy to obtain a solution with the same service level as R2D, i.e., at most the same number of late customers deliveries are allowed. The number of allowed deadline violations Θ is given by the objective value of TBRD-MIP presented in Section 4.1. The resulting MIP model consists of objective function (32) and constraints (33) to (40). The additional notation is summarized in Table 7.

V	set of vehicles ($\lvert V\rvert = \lvert C\rvert$)
Θ	maximum number of deadline violations allowed
t_k	continuous variables: arrival time at customer k
$x_{k_1,k_2,v}$	binary variables: 1, if customer k_1 visited direct before customer k_2 by truck v; 0, otherwise
z_k	binary variables: 1, if customer k is supplied late; 0, otherwise

Table 7: Additional notation for the MIP modelling the traditional truck-based delivery

$$\textbf{MinTruck-MIP: } \text{Minimize } F(T,X,Z) = \sum_{k\in C}\sum_{v\in V} x_{\gamma,k,v} \tag{32}$$

$$\sum_{k_1\in C\cup\{\gamma\}}\sum_{v\in V} x_{k_1,k_2,v} = 1 \qquad \forall\, k_2 \in C \tag{33}$$

$$\sum_{k\in C\cup\{\gamma\}} x_{\gamma,k,v} = 1 \qquad \forall\, v \in V \tag{34}$$

$$\sum_{k_1\in C\cup\{\gamma\}} x_{k_1,k_2,v} = \sum_{k_1\in C\cup\{\gamma\}} x_{k_2,k_1,v} \qquad \forall\, k_2 \in C; v \in V \tag{35}$$

$$t_{k_1} + \vartheta^t_{k_1,k_2} - M\cdot(1 - x_{k_1,k_2,v}) \le t_{k_2} \qquad \forall\, k_1,k_2 \in C; v\in V \tag{36}$$

$$\vartheta^t_{\gamma,k} - M\cdot(1 - x_{\gamma,k,v}) \le t_k \qquad \forall\, k \in C; v \in V \tag{37}$$

$$t_k - M\cdot z_k \le d_k \qquad \forall\, k\in C \tag{38}$$

$$\sum_{k\in C} z_k \le \Theta \tag{39}$$

$$x_{k_1,k_2,v}, z_{k_1} \in \{0,1\} \qquad \forall\, k_1,k_2 \in C\cup\{\gamma\}; v\in V \tag{40}$$

The objective function (32) minimizes the number of utilized trucks. Constraints (33) to (35) make sure that each customer is delivered by a vehicle and a well-defined sequence of visits is determined. The arrival times at the customers are set in constraints (36) and (37). Delays are detected by constraints (38) and the number of delays is limited by constraints (39). The domains of the binary variables are finally set by constraints (40).

Appendix C: Unattended home delivery

In the current planning stage, the delivery robots are not able to drop off loaded freight without human assistance, so that their application is restricted to the attended home delivery context. Recent developments in smart-lock technologies (e.g., Amazon Key), however, make unattended robot delivery a possible option for the future. This would be advantageous for both, customers, who can order goods without worrying about suited delivery times, and couriers, who are relieved from tight delivery time windows. Once unattended home delivery by robots is realized, there is no need to agree delivery time windows with customers. For our problem setting, this means that delivery deadlines are superfluous and other objectives, e.g., traditional vehicle routing objectives, need to be incorporated into our MIP model and the heuristic.

First, we explore one of the most popular vehicle routing objectives of minimizing the total travel distance. Adapting the introduced model TBRD-MIP necessitates to remove constraints (8) and to substitute the objective function (1) by

$$\text{Minimize } F(L,S,T,X) = \sum_{j\in D\cup R\cup\{\gamma\}} \sum_{k\in C} x_{j,k}\cdot \vartheta^r_{j,k} + \alpha \cdot \sum_{i\in D\cup R\cup\{\gamma\}} \sum_{j\in D\cup R\cup\{\gamma^e\}\setminus\{i\}} s_{i,j}\cdot \vartheta^t_{i,j},$$

where α weights the travel distance of the truck compared to the robots.

To adapt our heuristic procedure to this problem setting, the cost matrix of the transportation problem has to be determined in the following way:

- The costs from each supply node to the dummy demand node d are set to zero.
- The costs $c_{i,j}$ between nodes referring to a depot $r_i \in R(v)$ at stop $v_{r(i)}$ and a customer $j \in C$ are set to
$$c_{i,j} = \vartheta^r_{v_{r(i)},j}.$$
- Furthermore, costs $c_{k,j}$ between nodes referring to a subtour $S_k \in S(v)$ and a customer $j \in C$ are set to the minimum travel distance of the robots, such that
$$c_{k,j} = \min_{i=k_1,\dots,k_{i_k}} \left\{\vartheta^r_{v_i,j}\right\}.$$

Finally, the weighted truck route length has to be added to the value determined by the transportation problem. Note that we do not integrate the distances between customers and robot depots into our calculations. They are constant, since robots have to return to depots irrespective of the current assignment decision. In consequence, distances traveled after customer delivery can be neglected.

Another prominent vehicle routing objectives is the minimization of the trip duration. We can deal with this objective by introducing a further decision variable z representing the completion time of the tour. For our TBRD-MIP we, then, have to replace objective function (1) by

$$\text{Minimize } F(L,S,T,X,Z) = z$$

and constraints (8) by

$$z \geq t_j - M\cdot(1-x_{j,k}) + \vartheta^r_{j,k} \qquad \forall\, j \in D\cup R\cup\{\gamma\};\, k\in C.$$

For a given truck route, the minimal duration can be determined by binary search on the completion time z solving the introduced Hitchcock transportation problem. In this way, our heuristic solution procedure can be adapted to solve the considered problem by solving the binary search each time a truck route is evaluated. To assess if a given trip duration z can be kept for a predefined tour, the cost matrix has to be computed as follows:

- The costs from each supply node to the dummy demand node are set to zero.
- The costs $c_{i,j}$ between nodes referring to a depot $r_i \in R(v)$ at stop $v_{r(i)}$ and a customer $j \in C$ are:

$$c_{i,j} = \begin{cases} 0 & \text{, if } \sum_{l=1}^{r(i)-1} \vartheta^t_{v_l,v_{l+1}} + \vartheta^r_{v_{r(i)},j} \leq z \\ 1 & \text{, else.} \end{cases}$$

- Additionally, costs $c_{k,j}$ between nodes referring to a subtour $S_k \in S(v)$ and a customer $j \in C$ are:

$$c_{k,j} = \begin{cases} 0 & \text{, if } \min_{i=k_1,\dots,k_{i_k}} \left\{ \sum_{l=1}^{i-1} \vartheta^t_{v_l,v_{l+1}} + \vartheta^r_{v_i,j} \right\} \leq z \\ 1 & \text{, else.} \end{cases}$$

If the given trip duration z is satisfiable, i.e., each customer is timely supplied within trip duration z, the optimal objective value of the transportation problem amounts to 0. Otherwise, it is greater than zero. Based on this, the binary search can be continued to achieve the minimal trip duration for a given truck tour.

Note that in this configuration, the trip duration is measured based on the delivery time at the customer. This approach is valid whenever a fast delivery of customers is of high importance and robots are not a scarce resource. Whenever robots are scarce, however, the return time to the nearest depot should be modeled, too, to avoid shortages. Such an objective can easily be implemented by computing the return time to the next depot $\bar{\vartheta}^{depot}_k = \min_{j \in R}\{\vartheta^r_{k,j}\}$ for each customer $k \in C$ and considering this value in the formulas above.