Submitted to *Transportation Science*
manuscript TS-2023-0169.R1

Synchronized Deliveries with a Bike and a Self-Driving Robot

Yanlu Zhao
Durham University Business School, Durham University, Durham, UK
yanlu.zhao@durham.ac.uk

Diego Cattaruzza
Univ. Lille, CNRS, Centrale Lille, Inria UMR 9189 - CRIStAL Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
diego.cattaruzza@centralelille.fr

Ningxuan Kang
Department of Intelligent Supply Chain Y, JD.com, Beijing, China
kangningxuan@jd.com

Roberto Roberti
Department of Information Engineering, University of Padova, Padova, Italy
roberto.roberti@unipd.it

Online e-commerce giants are continuously investigating innovative ways to improve their practices in last-mile deliveries. Inspired by the current practices at JD.com (the largest online retailer by revenue in China), we investigate a delivery problem that we call *Traveling Salesman Problem with Bike-and-Robot* (TSPBR) where a cargo bike is aided by a self-driving robot to deliver parcels to customers in urban areas. We present two mixed-integer linear programming models and describe a set of valid inequalities to strengthen their linear relaxation. We show that these models can yield optimal solutions of TSPBR instances with up to 60 nodes. To efficiently find heuristic solutions, we also present a genetic algorithm based on a dynamic programming recursion that efficiently explores large neighborhoods. We computationally assess this genetic algorithm on instances provided by JD.com and show that high-quality solutions can be found in a few minutes of computing time. Finally, we provide some managerial insights to assess the impact of deploying the bike-and-robot tandem to deliver parcels in the TSPBR setting.

Key words: self-driving robots; bike delivery; synchronization; mixed-integer linear programming; genetic algorithm; last-mile delivery

History: Submitted: May 16, 2023; Revised: September 12, 2023; Accepted: October 27, 2023

1. Introduction

The growing number of Internet users and the opportunity to easily order their preferred products through mobile devices, such as tablets and phones, are boosting e-commerce sales. Indeed, online retailing sales in the three major e-commerce global markets (i.e., the U.S., China, and Europe) are estimated to increase at an annual rate of over 6% in the next three years (Statista 2021a).

Moreover, a recent survey about customers' expectations on e-commerce markets (Statista 2021b) indicates that almost half of online shoppers expect to benefit from speedy deliveries and delivery cost reductions in the coming years. The growth rate of e-commerce sales combined with such high customer expectations calls for smarter ways to run e-commerce businesses.

Planning last-mile delivery operations is one of the most crucial tasks faced by e-commerce companies (Archetti and Bertazzi 2021). If last-mile delivery is carefully designed, not only can customers' expectations be fulfilled, but also important related issues such as urban congestion and toxic emissions can be addressed. However, smart planning of last-mile delivery operations is particularly challenging because e-commerce companies can rely on razor-thin marginal profits.

To tackle all these challenges posed by e-commerce markets, e-commerce giants, such as Amazon or JD.com, are investigating new paradigms to run last-mile deliveries. One of the most promising ways to improve the current practices in last-mile delivery is to adopt *Unmanned Autonomous Vehicles* (UAV), such as drones or *Self-Driving Robots* (SDR), to complement or replace conventional vehicles. Deliveries with UAVs can allow companies to decrease delivery costs and represent an environmentally-friendly transport mode.

In this paper, we investigate a last-mile delivery problem faced by JD.com, which is the largest (in terms of revenue) Chinese business-to-consumer online retailer company, headquartered in Beijing, with about 550,000 employees (as of 2022) and a total annual revenue of about 151.7 billion dollars in 2022. Given the lack of professional truck drivers and strict regulations, imposed by local authorities, to reduce noise and carbon emissions in urban areas, JD.com is already using a mixed delivery force of conventional vehicles and cargo bikes (see the left panel of Figure 1) in last-mile delivery operations. JD.com is also considering the adoption of SDRs (such as the one displayed in the right panel of Figure 1) to use in combination with cargo bikes (which we refer to as bikes in the following) to deliver parcels to customers.

In particular, JD.com would like to gain insights on the economic feasibility and overall impact of associating a robot with a bike in the following setting. A set of customers must be served with a bike and its robot within a given planning horizon. At the beginning of the planning horizon, the bike and its robot are located at a depot, where all parcels to deliver are also stored. The bike has enough capacity to store all these parcels at once whereas the robot has limited capacity: it can carry just a subset of the parcels at the same time because it features a limited number of containers (of various volumes) to store the parcels. The customers can be served by one of the two vehicles that travel through two different routes, which start and end at the depot. Whenever the robot is empty along its route, it can join the bike at a customer, where some parcels that are on the bike can be moved to the robot for further deliveries. The costs to serve a customer with the bike or the robot are known. The goal of the problem is to find a distribution plan for the bike

and the robot so that all customers are served within the given planning horizon and the total distribution costs are minimized. In the following, we refer to this decision-making problem as the *Traveling Salesman Problem with Bike-and-Robot* (TSPBR), which, to the best of our knowledge, has not been studied in the literature so far.

Figure 1 **A cargo bike (in the left panel) and a self-driving robot (in the right panel)**

The main contributions of this paper are the following:

- We formally describe and formulate the TSPBR with two *Mixed-Integer Linear Programming* (MILP) models. The first model is a compact MILP featuring a polynomial number of variables and constraints. The second MILP builds upon the first model but features an exponential number of constraints. We also propose a set of valid inequalities to tighten the linear relaxation of these MILP models and embed these cuts into branch-and-cut algorithms.
- We present a genetic algorithm, based on dynamic programming recursions to explore a large neighborhood of TSPBR solutions, to find high-quality primal solutions to the problem.
- We test the proposed branch-and-cut and the genetic algorithms on real-life instances provided by JD.com. We show that the branch-and-cut methods can find optimal solutions for most of the TSPBR instances with up to 60 nodes and the genetic algorithm can find high-quality solutions in a few minutes of computing time.
- We assess the economic impact of performing last-mile deliveries with the bike and the robot operating in tandem. We show that deploying the robot can attain significant cost reductions and time savings to fulfill all customer requests.

The remainder of the paper is organized as follows. Section 2 reviews the literature related to our problem. Section 3 formally defines the TSPBR. In Section 4, we present two MILP formulations for the TSPBR and a family of valid inequalities to tighten their linear relaxations. Section 5 presents a genetic algorithm to find heuristic solutions to the TSPBR. In Section 6, we report on the results of computational experiments to assess the performance of the mathematical models and the genetic

algorithm. In Section 7, we draw some conclusions from our study and discuss future research directions.

2. Literature Review

The interest of the community in routing problems with UAVs has recently significantly increased. This is mainly motivated by last-mile delivery applications, where UAVs integrate the delivery system in use. Indeed, deliveries with UAVs can help achieve quick deliveries and reduce vehicle traffic and emission (Srinivas, Ramachandiran, and Rajendran 2022).

The literature dealing with routing problems with UAVs considers fleets made up of UAVs only or conventional vehicles (trucks or vans) teamed up with drones or robots. The number of scientific papers dealing with such problems is large, so an exhaustive review is out of reach in this paper. Thus, we limit our review to papers that consider deliveries with SDRs (also called autonomous delivery robots in the literature) teamed up with other vehicles, as in our problem, and papers that investigate deliveries with drones as these problems present similarities with ours. For comprehensive reviews on these two classes of problems, we refer the reader to the survey of Srinivas, Ramachandiran, and Rajendran (2022) on autonomous robot-driven deliveries and the surveys of Otto et al. (2018) and Macrina et al. (2020) about routing problems with drones.

2.1. Deliveries with Self-Driving Robots

Srinivas, Ramachandiran, and Rajendran (2022) classify routing problems with SDRs in three categories: two-tier models, mothership models, and platoon models. We use this classification to review the main contributions from the literature.

2.1.1. Two-tier models. In two-tier models, parcels are first delivered by a conventional truck from a central depot to hubs from where SDRs operate deliveries to final customers. For example, Bakach, Campbell, and Ehmke (2021) and Alfandari, Ljubić, and da Silva (2022) study problems in this category.

Bakach, Campbell, and Ehmke (2021) investigate a two-tier robots-based urban delivery network. First, they solve a facility location problem for minimum open robot depot locations. Then, they solve a p-median problem to further minimize the operational cost of robot deliveries.

Alfandari, Ljubić, and da Silva (2022) investigate a problem setting where a single truck carries parcels from a central depot to a set of facilities. At the facilities, SDRs are launched to deliver parcels to final customers assigned to the facility. Each robot delivers a single parcel. Customers are associated with a delivery deadline, and the objective is to serve all customers while minimizing total tardiness.

2.1.2. Mothership models. Mothership models feature vehicles, called motherships, that carry SDRs and the required parcels. Motherships travel to dedicated locations with parking possibilities where SDRs leave the mothership to reach their delivery destination.

Boysen, Schwerdfeger, and Weidinger (2018) study a problem where the truck leaves a central depot loaded with a number of SDRs and all the parcels to deliver. The truck then stops at a drop-off point and launches a set of SDRs that perform single-parcel deliveries. After the deliveries, SDRs go to a dedicated depot. In the meantime, the truck moves to another drop-off point or a robot depot where it can load robots to use for the next deliveries. The objective is to minimize total tardiness. Ostermeier, Heimfarth, and Hübner (2022) study a similar problem where each customer is associated with a soft time window. Early and late arrivals are penalized in the objective function, which also accounts for routing costs calculated as a linear function of the traveling distance and travel time of the mothership and the robots.

The problem addressed by Yu, Puchinger, and Sun (2020) features SDRs that are dropped off at rendezvous nodes. Each SDR then performs several deliveries before being collected at another rendezvous node by the same mothership. The objective is to minimize the total delivery cost.

Simoni, Kutanoglu, and Claudel (2020) deal with a single mothership that collaborates with a single SDR to deliver packages to a set of customers. The SDR can be dropped off or picked up at each customer location and can serve several customers subsequently. The objective is to minimize the total time needed to complete deliveries.

Yu, Puchinger, and Sun (2021) address a problem where vans (possibly with robots onboard) leave a central depot, serve customers, or drop off/pick up, and replenish or swap robots' batteries at parking nodes. Robots then handle customer services along open routes: they are not obliged to return to parking nodes from where they set out. A service at a customer location can be either a delivery or a parcel pick-up. The objective is to minimize the total delivery cost.

Finally, Chen et al. (2021) and Chen, Demir, and Huang (2021) consider the setting in which vans carry several robots equipped with a single compartment. Vans visit customers in order to serve them as well as to drop off robots. Robots then perform deliveries and go back to the drop-off point where the van waits for them before starting its route again. Customers are associated with hard time windows. The objective is to minimize the total delivery time.

2.1.3. Platoon models. In platoon models, SDRs are only allowed to autonomously drive inside dedicated zones. Whenever they need to exit one of these zones to reach another one, they need to be guided by trucks in platoons. A truck can guide a maximum number of SDRs at a time.

Scherr et al. (2019) study an example of a platoon model, where the goal is to determine the size and mix of the fleet as well as the routing of vehicles and goods on a given network in order to minimize the total cost of the system.

2.2. Deliveries with Drones

The problem setting we investigate in this paper has similarities with two well-studied problems in the literature about deliveries with drones, namely, the *Flying Sidekick Traveling Salesman Problem* (FSTSP) and the *Traveling Salesman Problem with Drone* (TSP-D).

2.2.1. The flying sidekick traveling salesman problem. The FSTSP is the first routing problem with drones studied in the literature (Murray and Chu 2015). A single truck is aided by a drone to serve customers. While the truck is following its route, the drone can be launched from the truck at a location (depot or customer location) and rejoin the truck at another location. The drone is allowed to carry a single parcel at a time and can serve compatible customers only. Accessibility restrictions are imposed on customers that can be served by the drone for several reasons: the parcel may be too heavy, a signature from the customer may be required, or the customer's location may not allow a safe landing. Moreover, battery capacity restrictions limit the flying range of the drone. The FSTSP aims at minimizing the completion time to serve all customers and return to the depot, including potential waiting times for the truck and the drone for their synchronization.

Murray and Chu (2015) propose the first MILP model and a heuristic for the FSTSP. de Freitas and Penna (2020) present a variable neighborhood search. Liu, Li, and Khojandi (2022) describe a reinforcement learning approach. A fairly wide range of exact methods, all based on MILP models of the problem, are available to solve the FSTSP to optimality, e.g., column-and-row generation (Boccia et al. 2021), branch-and-bound (Dell'Amico, Montemanni, and Novellani 2021, 2022), and branch-and-price (Roberti and Ruthmair 2021).

2.2.2. The traveling salesman problem with drone. Even though the literature defines the TSP-D in a broad sense and different papers consider different features of the problem, the TSP-D can be briefly described as follows. A truck equipped with a single drone is located at a depot, where parcels to deliver are stored. The truck-and-drone tandem must serve a set of customers. Along its route, the truck can launch the drone to serve a single customer (i.e., deliver a single parcel) before retrieving the drone at a subsequent location. The goal is to minimize the time that the two vehicles return to the depot after serving all customers. Features such as customer incompatibility to drone delivery, limited drone flying range, possible revisits to locations, and take-off and landing at non-customer locations (to mention a few) have all been studied in the literature. All in all, the TSP-D can be seen as a class of delivery problems with drones, more than a specific problem, and, depending on its definition, can be a special case or a generalization of the FSTSP.

A wide variety of methodologies have been developed to solve the TSP-D both in a heuristic way and to optimality. Examples of heuristic methods are the dynamic-programming-based approach of

Agatz, Bouman, and Schmidt (2018), the optimization-based method of Es Yurek and Ozmutlu (2018), and the hybrid genetic algorithm of Ha et al. (2020). The main exact methods for the TSP-D are the dynamic programming recursions of Bouman, Agatz, and Schmidt (2018), the branch-and-bound algorithm of Poikonen, Golden, and Wasil (2019), the branch-and-cut algorithms of Schermer, Moeini, and Wendt (2020) and El-Adle, Ghoniem, and Haouari (2021), the Benders-like decomposition method of Vásquez, Angulo, and Klapp (2021), and the branch-and-price algorithm of Roberti and Ruthmair (2021).

It is worth mentioning that, despite all the efforts to develop efficient solution algorithms to solve the TSP-D to optimality, the algorithms currently available allow us to consistently find optimal solutions to instances with up to 30-35 customers.

3. Problem Definition

In this section, we formally describe the TSPBR. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph with vertex set $\mathcal{V}$ and arc set $\mathcal{A}$. The vertex set $\mathcal{V}$ is defined as $\mathcal{V} = \mathcal{N} \cup \{0, 0'\}$, where $\mathcal{N}$ represents a set of n customers to serve within a given planning horizon and the vertices 0 and $0'$ represent two copies of the depot (the initial and final vertex of the bike and robot routes). In the following, we also use the notation $\mathcal{N}_0 = \mathcal{N} \cup \{0\}$ and $\mathcal{N}_{0'} = \mathcal{N} \cup \{0'\}$ to refer to the set of customers plus the initial depot and the set of customers plus the final depot, respectively. The arc set $\mathcal{A}$ is defined as $\mathcal{A} = \{(0, j) \,|\, j \in \mathcal{N}\} \cup \{(i, j) \,|\, i, j \in \mathcal{N} : i \neq j\} \cup \{(i, 0') \,|\, i \in \mathcal{N}\}$. The volume of the parcel to deliver to customer $i \in \mathcal{N}$ is represented by $v_i \in \mathbb{R}_+$, and the service time to serve $i \in \mathcal{N}$ is denoted as $s_i \in \mathbb{R}_+$.

At the beginning of the planning horizon, a bike and a robot are available at the depot and are used to deliver all parcels to the customers of the set $\mathcal{N}$. The length of the planning horizon is represented by ζ (expressed in some units of time), which also represents the maximum working time of each of the two vehicles. The bike can serve all customers whereas the robot may not be able to serve some of the customers because of geographical restrictions or excessive volume of the parcels to deliver. A binary parameter r_i indicates if customer $i \in \mathcal{N}$ can be served with the robot (in this case, $r_i = 1$) or not ($r_i = 0$). The parameter r_i allows us to define the subgraph $\mathcal{G}^r = (\mathcal{V}^r, \mathcal{A}^r)$ that represents the set of nodes the robot can visit and the set of arcs it can travel through. In particular, the node set $\mathcal{V}^r \subseteq \mathcal{V}$ is defined as $\mathcal{V}^r = \mathcal{N}^r \cup \{0, 0'\}$, where $\mathcal{N}^r = \{i \in \mathcal{N} \,|\, r_i = 1\}$, and the arc set $\mathcal{A}^r \subseteq \mathcal{A}$ as $\mathcal{A}^r = \{(0, j) \,|\, j \in \mathcal{N}^r\} \cup \{(i, j) \,|\, i, j \in \mathcal{N}^r : i \neq j\} \cup \{(i, 0') \,|\, i \in \mathcal{N}^r\}$. Customers of the set $\mathcal{N}^r$ are called *robot customers* in the following. We also denote by $\mathcal{N}^b = \mathcal{N} \setminus \mathcal{N}^r$ the set of customers that must be served by the bike – we call these customers *bike customers*.

As the bike and the robot travel at different speeds, we denote by $t^b_{ij} \in \mathbb{Z}_+$ the bike travel time through arc $(i, j) \in \mathcal{A}$ and $t^r_{ij} \in \mathbb{Z}_+$ the robot travel time through arc $(i, j) \in \mathcal{A}^r$. The cost for serving a customer with the bike is denoted by c^b and corresponds, for example, to the biker's fixed delivery

service fee; the cost for serving a customer with the robot is denoted by c^r and corresponds, for example, to average battery charging fees. Operational practices indicate that it is cheaper to serve a customer with the robot than with the bike, so we assume, throughout the paper, that $c^b > c^r$. Note that these costs are customer-independent.

While the bike has enough carrying capacity to load all parcels to deliver at the same time, the robot can accommodate Q^r parcels in Q^r independent containers (at most one parcel can be stored in a container). Based on the features of the robots currently available on the market, we assume that these Q^r containers are of at most three different volumes (large, medium, and small), and we denote by Q^l, Q^m, and Q^s the number of large, medium, and small containers, resp., where $Q^r = Q^l + Q^m + Q^s$. Despite this assumption that containers are of at most three different volumes, the models and algorithms presented throughout the paper can easily be adapted if the robot features containers of more than three different volumes.

As the robot has a limited number of containers, it may not be possible to load all the parcels to deliver upon leaving the depot. Therefore, whenever the robot does not have any parcels on board, it can rejoin the bike at a customer of the set $\mathcal{N}^r$ served by the bike to take other parcels to deliver afterward. These *replenishment operations* can take place at customer locations only to avoid, for example, parking issues and bike detours. A customer where a replenishment takes place is called *rendezvous customer*. Each replenishment requires the temporal synchronization of the two vehicles (i.e., parcels can be moved from the bike to the robot only if both vehicles are at the location where the replenishment takes place). We assume that each replenishment takes η units of time, regardless of the number of packages to move.

The TSPBR calls for the definition of two routes (one for the bike and one for the robot) such that: (a) each route starts from the depot, visits a set of customers each at most once, ends at the depot, and does not take longer than ζ units of time; (b) each customer of the set $\mathcal{N}^b$ is served by the bike exactly once, and each customer of the set $\mathcal{N}^r$ is served by either the bike or the robot exactly once; (c) replenishment operations take place when the robot is empty at customers of the set $\mathcal{N}^r$ that are served by the bike; (d) parcels loaded on the robot at each replenishment operation can be feasibly stored in the containers; (e) the total cost to serve the customers is minimized.

Notice that, unlike other variants of the TSP, where the total cost is defined as the sum of the costs of the arcs traversed, in the TSPBR, the total cost is given by the number of customers served by the bike times the cost of serving a customer with the bike (i.e., c^b) plus the number of customers served by the robot times the cost of serving a customer with the robot (i.e., c^r). This objective function is motivated by two main reasons: (i) JD.com cannot assess the cost to traverse an arc with a vehicle but can only estimate the cost of serving a customer with a vehicle, and (ii) the TSPBR can be applied to last-mile deliveries in urban areas where distances are limited and

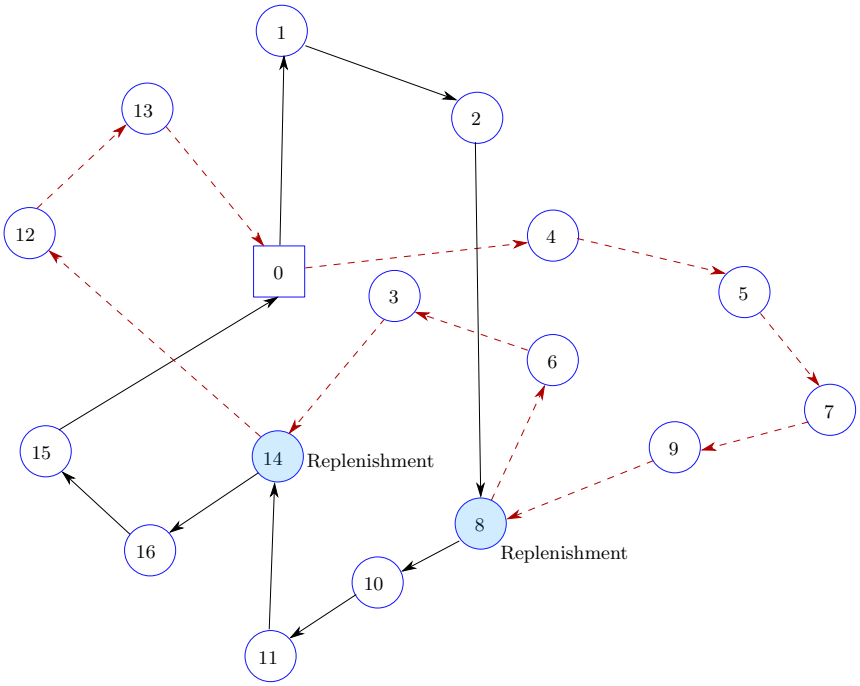

Figure 2 A feasible solution of a TSPBR instance with 16 customers

travel times between pairs of different locations have small deviations. The algorithms presented in this paper can easily be adjusted to tackle variants of the TSPBR where a cost to traverse an arc with one of the two vehicles is incurred and can be estimated.

Given the assumption that $c^b > c^r$, we can observe that minimizing the total cost to serve all customers is equivalent to minimizing the number of customers delivered by the bike.

To better clarify the definition of the TSPBR, Figure 2 illustrates a feasible solution of a TSPBR instance with 16 customers (represented with circles). For ease of presentation, all service times are equal to zero except for customers 8 and 14, i.e., $s_8 > 0$ and $s_{14} > 0$. The bike leaves the depot (represented with a rectangle) to serve customers 1, 2, and 8 while the robot leaves the depot to serve customers 4, 5, 7, and 9. At customer 8, the two vehicles must be synchronized, and the robot is replenished with the parcels destined for customers 3 and 6. The bike serves customer 8 either before or after replenishing the robot (this is a decision to make); let us assume that the service takes place prior to the robot's replenishment. Depending on the travel times along the paths $0 \to 1 \to 2 \to 8$ and $0 \to 4 \to 5 \to 7 \to 9 \to 8$, either the bike waits for the robot (if $t^b_{0,1} + t^b_{1,2} + t^b_{2,8} + s_8 < t^r_{0,4} + t^r_{4,5} + t^r_{5,7} + t^r_{7,9} + t^r_{9,8}$) or the robot waits for the bike (if $t^b_{0,1} + t^b_{1,2} + t^b_{2,8} + s_8 > t^r_{0,4} + t^r_{4,5} + t^r_{5,7} + t^r_{7,9} + t^r_{9,8}$). After the replenishment, the bike serves customers 10, 11, and 14 while the robot serves customers 6 and 3. Another replenishment takes place at customer 14 where we assume that the replenishment occurs before serving the customer. Afterward,

the bike serves customers 16 and 15, and the robot serves customers 12 and 13. Finally, the two vehicles return to the depot. The working time of this solution is

$$\begin{aligned} t = \eta + \max\{t^b_{0,1} + t^b_{1,2} + t^b_{2,8} + s_8,\ t^r_{0,4} + t^r_{4,5} + t^r_{5,7} + t^r_{7,9} + t^r_{9,8}\} + \\ \eta + \max\{t^b_{8,10} + t^b_{10,11} + t^b_{11,14},\ t^r_{8,6} + t^r_{6,3} + t^r_{3,14}\} + \\ \eta + \max\{s_{14} + t^b_{14,16} + t^b_{16,15} + t^b_{15,0'},\ t^r_{14,12} + t^r_{12,13} + t^r_{13,0'}\} \end{aligned}$$

To be feasible, the solution must satisfy $t \leq \zeta$, and the parcels loaded on the robot at the depot, at customer 8, and at customer 14 should be feasibly accommodated in the robot's containers.

In the remainder of the paper, we use the following terminology. We call *leg* a directed path that starts and ends with two replenishment operations. A *bike leg* (*robot leg*, resp.) is a leg traversed by the bike (robot, resp.). The bike leg *corresponding* to a given robot leg is the bike leg having the same start and end nodes as the given robot leg (and vice versa). A bike leg and the corresponding robot leg constitute an *operation*. For example, the solution illustrated in Figure 2 consists of three operations and six legs: three bike legs (i.e., $0 \to 1 \to 2 \to 8$, $8 \to 10 \to 11 \to 14$, and $14 \to 16 \to 15 \to 0'$) and three robot legs (i.e., $0 \to 4 \to 5 \to 7 \to 9 \to 8$, $8 \to 6 \to 3 \to 14$, and $14 \to 12 \to 13 \to 0'$). The robot leg corresponding to the bike leg $0 \to 1 \to 2 \to 8$ is $0 \to 4 \to 5 \to 7 \to 9 \to 8$. The bike leg corresponding to the robot leg $8 \to 6 \to 3 \to 14$ is $8 \to 10 \to 11 \to 14$.

Notice the TSPBR does not fall in any of the three categories of problems (two-tier, mothership, and platoon models) proposed in Srinivas, Ramachandiran, and Rajendran (2022) to classify delivery problems with SDRs. Indeed, the TSPBR is not a two-tier model as it considers a one-tier delivery system. Moreover, it does not feature a mothership vehicle carrying SDRs to dedicated locations; on the contrary, in the TSPBR, the robot starts and ends its route at the depot. Finally, the TSPBR does not consider dedicated zones where the robot is allowed to autonomously drive and other zones in which it is not; consequently, the SDR does not need to be guided by another vehicle to move among dedicated zones as it happens in platoon models. Therefore, the TSPBR is a new decision-making problem in the literature about deliveries with SDRs.

Notice, also, that the TSPBR is significantly different from the FSTSP and the TSP-D, whose features are representative of the main features investigated in the literature of deliveries with drones. Indeed, while the FSTSP and the TSP-D aim at minimizing the makespan to serve all customers, the TSPBR features a cost-minimization objective function. In terms of constraints, the capacity of the robot allows for serving multiple customers between replenishment operations whereas the drone can serve a single customer in each leg in the FSTSP and the TSP-D. Hence, to the best of our knowledge, the TSPBR is different from all the other problems studied in the literature about deliveries with drones.

In general terms, we observe that the TSPBR falls in the broad category of routing problems with synchronization aspects. Following both the classifications proposed in Drexl (2012) and Soares et al. (2023), the TSPBR is characterized by the synchronization of operations due to the requirement of having both the bike and the robot at the same location at the same time for replenishment purposes. Moreover, with respect to the scheme proposed in Soares et al. (2023), the local aspects of the TSPBR are the capacity constraints of the robot and the maximum working time imposed on the routes of the two vehicles.

4. Mathematical Formulations

In this section, we first provide a compact formulation, F_1, of the TSPBR (see Section 4.1) and, then, build upon it we derive another formulation, F_2, featuring an exponential number of constraints (see Section 4.2). Finally, we present a set of valid inequalities to tighten the linear relaxation of both formulations (Section 4.3).

4.1. Formulation F_1

The first mathematical formulation, F_1, uses the following sets of binary variables

- $x^b_{ij} = 1$ if the bike traverses arc $(i,j) \in \mathcal{A}$ (0 o/w);
- $x^r_{ij} = 1$ if the robot traverses arc $(i,j) \in \mathcal{A}^r$ (0 o/w);
- $y^b_i = 1$ if node $i \in \mathcal{N}$ is served by the bike and no replenishment takes place at i (0 o/w);
- $y^r_i = 1$ if node $i \in \mathcal{N}^r$ is served by the robot (0 o/w);
- $y^s_i = 1$ if a replenishment takes place at node $i \in \mathcal{N}^r$ and i is served by the bike (0 o/w);
- α_i (β_i, resp.) $= 1$ if a replenishment takes place at customer $i \in \mathcal{N}^r$ and the bike serves i before (after, resp.) the replenishment (0 o/w);
- $z^l_i, z^m_i, z^s_i = 1$ if the parcel of customer $i \in \mathcal{N}^r$ is stored in a large, medium, or small container, resp. (0 o/w).

Furthermore, formulation F_1 features four sets of auxiliary non-negative real variables, namely

- $f^l_{ij}, f^m_{ij}, f^s_{ij} \in \mathbb{R}_+$ representing the number of parcels stored in large, medium, and small containers, resp., when the robot traverses arc $(i,j) \in \mathcal{A}^r$;
- $a_i \in \mathbb{R}_+$ representing the arrival time of the vehicle at node $i \in \mathcal{V}$ (if i is visited by either the bike or the robot) or the time at which the replenishment starts at node $i \in V$ (if i is a rendezvous customer).

Moreover, for ease of presentation, we introduce binary parameters C^l_i, C^m_i, C^s_i indicating if the parcel of customer $i \in \mathcal{N}^r$ fits a large, medium, and small robot's container, resp.

Given these decision variables and parameters, the TSPBR can be formulated as

$$[F_1] \quad \min \sum_{i \in \mathcal{N}} (y^b_i + y^s_i) \tag{1a}$$

$$\text{s.t.} \sum_{(0,j)\in\mathcal{A}} x^b_{0j} = \sum_{(i,0')\in\mathcal{A}} x^b_{i0'} = 1 \tag{1b}$$

$$\sum_{(i,j)\in\mathcal{A}} x^b_{ij} = \sum_{(j,i)\in\mathcal{A}} x^b_{ji} = y^b_i + y^s_i \qquad i\in\mathcal{N} \tag{1c}$$

$$\sum_{(0,j)\in\mathcal{A}^r} x^r_{0j} = \sum_{(i,0')\in\mathcal{A}^r} x^r_{i0'} = 1 \tag{1d}$$

$$\sum_{(i,j)\in\mathcal{A}^r} x^r_{ij} = \sum_{(j,i)\in\mathcal{A}^r} x^r_{ji} = y^r_i + y^s_i \qquad i\in\mathcal{N}^r \tag{1e}$$

$$y^b_i + y^r_i + y^s_i = 1 \qquad i\in\mathcal{N} \tag{1f}$$

$$\sum_{(i,j)\in\mathcal{A}} (s_i + t^b_{ij}) x^b_{ij} + \sum_{i\in\mathcal{N}} \eta y^s_i \le \zeta \tag{1g}$$

$$\sum_{(i,j)\in\mathcal{A}^r} t^r_{ij} x^r_{ij} + \sum_{i\in\mathcal{N}^r} (s_i y^r_i + \eta y^s_i) \le \zeta \tag{1h}$$

$$y^s_i = \alpha_i + \beta_i \qquad i\in\mathcal{N}^r \tag{1i}$$

$$t^b_{0i} x^b_{0i} + t^r_{0i} x^r_{0i} + \eta \le a_i + M(1 - x^b_{0i} - x^r_{0i}) \qquad i\in\mathcal{N}^r \tag{1j}$$

$$t^b_{0i} x^b_{0i} + \eta \le a_i + M(1 - x^b_{0i}) \qquad i\in\mathcal{N}\setminus\mathcal{N}^r \tag{1k}$$

$$a_i + \eta y^s_i + s_i y^r_i + t^r_{ij} x^r_{ij} \le a_j + M(1 - x^r_{ij}) \qquad (i,j)\in\mathcal{A}^r : i\neq 0, j\neq 0' \tag{1l}$$

$$a_i + \eta y^s_i + s_i y^b_i + s_i\beta_i + t^b_{ij} x^b_{ij} + s_j\alpha_j \le a_j + M(1 - x^b_{ij}) \qquad (i,j)\in\mathcal{A} : i\neq 0, j\neq 0' \tag{1m}$$

$$a_i + \eta y^s_i + s_i y^r_i + t^r_{i0'} x^r_{i0'} \le \zeta + M(1 - x^r_{i0'}) \qquad i\in\mathcal{N}^r \tag{1n}$$

$$a_i + \eta y^s_i + s_i y^b_i + s_i\beta_i + t^b_{i0'} x^b_{i0'} \le \zeta + M(1 - x^b_{i0'}) \qquad i\in\mathcal{N} \tag{1o}$$

$$C^l_i z^l_i + C^m_i z^m_i + C^s_i z^s_i = y^r_i \qquad i\in\mathcal{N}^r \tag{1p}$$

$$\sum_{(j,i)\in\mathcal{A}^r} f^l_{ji} - \sum_{(i,j)\in\mathcal{A}^r} f^l_{ij} \ge C^l_i z^l_i - Q^l(1 - y^r_i) \qquad i\in\mathcal{N}^r \tag{1q}$$

$$\sum_{(j,i)\in\mathcal{A}^r} f^m_{ji} - \sum_{(i,j)\in\mathcal{A}^r} f^m_{ij} \ge C^m_i z^m_i - Q^m(1 - y^r_i) \qquad i\in\mathcal{N}^r \tag{1r}$$

$$\sum_{(j,i)\in\mathcal{A}^r} f^s_{ji} - \sum_{(i,j)\in\mathcal{A}^r} f^s_{ij} \ge C^s_i z^s_i - Q^s(1 - y^r_i) \qquad i\in\mathcal{N}^r \tag{1s}$$

$$f^l_{ij} \le Q^l x^r_{ij} \qquad f^m_{ij} \le Q^m x^r_{ij} \qquad f^s_{ij} \le Q^s x^r_{ij} \qquad (i,j)\in\mathcal{A}^r \tag{1t}$$

$$\sum_{(i,j)\in\mathcal{A}^r} f^l_{ij} \ge Q^l y^s_i \qquad \sum_{(i,j)\in\mathcal{A}^r} f^m_{ij} \ge Q^m y^s_i \qquad \sum_{(i,j)\in\mathcal{A}^r} f^s_{ij} \ge Q^s y^s_i \qquad i\in\mathcal{N}^r \tag{1u}$$

$$x^b_{ij} \in \{0,1\} \qquad (i,j)\in\mathcal{A} \tag{1v}$$

$$x^r_{ij} \in \{0,1\} \qquad f^l_{ij},\, f^m_{ij},\, f^s_{ij} \in \mathbb{R}_+ \qquad (i,j)\in\mathcal{A}^r \tag{1w}$$

$$y^b_i \in \{0,1\} \qquad i\in\mathcal{N} \tag{1x}$$

$$y^r_i,\, y^s_i,\, \alpha_i,\, \beta_i,\, z^s_i,\, z^m_i,\, z^l_i \in \{0,1\} \qquad i\in\mathcal{N}^r \tag{1y}$$

$$a_j \ge 0 \qquad j\in\mathcal{V} \tag{1z}$$

The objective function (1a) aims at minimizing the number of customers served by the bike. Constraints (1b) ensure that the bike leaves and returns to the depot once. Constraints (1c) are flow conservation constraints of the bike and link variables $\boldsymbol{x}^b$ with variables $\boldsymbol{y}^b$ and $\boldsymbol{y}^s$. Constraints (1d)–(1e) are the counterpart of constraints (1b)–(1c) for the robot. Constraints (1f) ensure that each

customer is served exactly once. Constraints (1g)–(1h) guarantee that the total working time of each of the two vehicles does not exceed ζ; notice that these two constraints are not necessary for the completeness of the formulation but are included because they can tighten the linear relaxation of the model. Constraints (1i) indicate that if a node is selected for a replenishment, either its service is scheduled before the replenishment or vice versa. Constraints (1j)–(1k) set the arrival times at the first customer visited by the two vehicles. Constraints (1l)–(1m) are sub-tour elimination constraints in the Miller-Tucker-Zemlin form and act as synchronization constraints to align the arrival times of the bike and the robot at each node; note that the value of M can be set as $M = \zeta - \min_{i\in\mathcal{N}}\{t^r_{0i}, t^b_{0i}\} - \min_{i\in\mathcal{N}}\{t^r_{i0'}, t^b_{i0'}\}$. Constraints (1n)–(1o) ensure that the bike and the robot return to the depot by the end of the planning horizon. Constraints (1p) ensure that if a customer is served by a robot, the corresponding parcel is stored in a container. Constraints (1q)–(1s) are commodity-flow constraints to manage the different sizes of the robot's containers. Constraints (1t) impose the maximum value to variables $\boldsymbol{f}^l$, $\boldsymbol{f}^m$, and $\boldsymbol{f}^s$. Constraints (1u) reset the robot capacity after each synchronization replenishment. Constraints (1v)–(1z) define the variable domains.

4.2. Formulation F_2

In formulation F_1, the role of (1j)–(1o) is twofold: they act as sub-tour elimination constraints, and they synchronize the bike and the robot in order not to exceed the maximum working time. It is well known that this type of constraint deteriorates the quality of the linear relaxation of MILP formulations, which turns in poor computational performance of the model when solved with a branch-and-bound solution method. To partly address this issue, we propose replacing constraints (1j)–(1o) as follows.

As said, constraints (1j)–(1o) are needed to eliminate sub-tours. However, sub-tours can be ruled out also by using the following sets of constraints

$$a_i + 1 \le a_j + (n+1)(1 - x^b_{ij} - x^r_{ij}) \qquad (i,j) \in \mathcal{A}^r \tag{2a}$$

$$a_i + 1 \le a_j + (n+1)(1 - x^b_{ij}) \qquad (i,j) \in \mathcal{A} \setminus \mathcal{A}^r \tag{2b}$$

$$a_i \le n \qquad i \in \mathcal{N} \tag{2c}$$

$$a_0 = 0,\ a_{0'} = n+1 \tag{2d}$$

where variable a_i, $i \in \mathcal{V}$, now represents the position of node i in the bike or robot tour.

If we simply replace constraints (1j)–(1o) with constraints (2a)–(2d), the resulting MILP may admit infeasible solutions because there is no guarantee that the constraint on the maximum working time is satisfied when considering synchronization between the bike and the robot.

To overcome this issue, the following no-good cuts (i.e., cuts that cut off a single solution at a time) are necessary for the correctness of the formulation F_2

$$\sum_{(i,j)\in\mathcal{A}^b(\boldsymbol{\xi})} x^b_{ij} + \sum_{(i,j)\in\mathcal{A}^r(\boldsymbol{\xi})} x^r_{ij} \leq |\mathcal{A}^b(\boldsymbol{\xi})| + |\mathcal{A}^r(\boldsymbol{\xi})| - 1 \quad \boldsymbol{\xi}\in\Omega \tag{3}$$

Here Ω is the set of solutions that satisfy (1b)–(1i), (1p)–(1z), and (2a)–(2d) but violate the working time restriction, and $\mathcal{A}^b(\boldsymbol{\xi})$ (resp. $\mathcal{A}^r(\boldsymbol{\xi})$) is a set containing *some* of the arcs used by the bike (the robot, resp.) in solution $\boldsymbol{\xi}$. The exact composition of $\mathcal{A}^b(\boldsymbol{\xi})$ and $\mathcal{A}^r(\boldsymbol{\xi})$ depends on the solution $\boldsymbol{\xi}$ as explained in the following.

First, let us define, for each solution $\boldsymbol{\xi}=(\boldsymbol{x},\boldsymbol{y},\boldsymbol{a},\boldsymbol{z},\boldsymbol{f},\boldsymbol{\alpha},\boldsymbol{\beta})\in\Omega$, $\mathcal{A}^b_l(\boldsymbol{\xi})$ as the set of arcs that belong to bike legs lasting at least as long as the corresponding robot leg. Similarly, let $\mathcal{A}^r_l(\boldsymbol{\xi})$ be the set of arcs that belong to robot legs lasting strictly more than the corresponding bike leg. In addition, let $\mathcal{N}^s(\boldsymbol{\xi})$ represent the set of synchronization nodes and $\mathcal{T}^s(\boldsymbol{\xi})$ the total time of $\boldsymbol{\xi}$. Then, let us calculate the following value

$$\tilde{\mathcal{T}}^s(\boldsymbol{\xi}) = \sum_{(i,j)\in\mathcal{A}^b_l(\boldsymbol{\xi})} (s_i + t^b_{ij})\cdot x^b_{ij} + \sum_{(i,j)\in\mathcal{A}^r_l(\boldsymbol{\xi})} (s_i + t^r_{ij})\cdot x^r_{ij} + \sum_{i\in\mathcal{N}^s(\boldsymbol{\xi})} (\eta - s_i)\cdot y^s_i \tag{4}$$

Note that $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})$ represents the time that the solution would have when the service time at rendezvous nodes is not taken into account, and it considers the longer leg associated with each operation. The value $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})$ clearly is a lower bound on the exact total working time, $\mathcal{T}^s(\boldsymbol{\xi})$, of the solution under consideration (i.e., $\boldsymbol{\xi}$) since incorporating the service time at rendezvous nodes can only delay operations.

Consequently, two cases can appear after obtaining the value of $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})$: (*i*) if $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})>\zeta$, the solution $\boldsymbol{\xi}$ is clearly infeasible and setting $\mathcal{A}^b(\boldsymbol{\xi})=\mathcal{A}^b_l(\boldsymbol{\xi})$ and $\mathcal{A}^r(\boldsymbol{\xi})=\mathcal{A}^r_l(\boldsymbol{\xi})$ allows to determine a no-good cut that eliminates the current solution $\boldsymbol{\xi}$; (*ii*) otherwise (i.e., if $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})\leq\zeta<\mathcal{T}^s(\boldsymbol{\xi})$), $\mathcal{A}^b(\boldsymbol{\xi})$ and $\mathcal{A}^r(\boldsymbol{\xi})$ contain all the arcs involved in the operations of $\boldsymbol{\xi}$.

4.2.1. Separation of no-good cuts (3)**.** The separation of constraints (3) is achieved by applying a two-step procedure. Firstly, the value of $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})$ is calculated. If $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})>\zeta$, the sets $\mathcal{A}^b_l(\boldsymbol{\xi})$ and $\mathcal{A}^r_l(\boldsymbol{\xi})$ are acquired by leg-wise comparison in each operation, and the corresponding cut is determined accordingly. Otherwise, if $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})\leq\zeta$, we construct a multi-partite acyclic support graph $\mathcal{G}^s(\boldsymbol{\xi})=(\mathcal{V}^s(\boldsymbol{\xi}),\mathcal{A}^s(\boldsymbol{\xi}))$, where $\mathcal{V}^s(\boldsymbol{\xi})$ contains nodes $\{0,0'\}$ and two nodes for each synchronization node $i\in\mathcal{N}^s(\boldsymbol{\xi})$, represented by i^α and i^β, where i^α indicates that the service at node i takes place before the replenishment and i^β vice versa. Moreover, let us order nodes in $\mathcal{N}^s(\boldsymbol{\xi})$ in such a way that $j<k$ if and only if i_j is visited by the robot and the bike before i_k in $\boldsymbol{\xi}$, $1\leq j<k\leq r=|\mathcal{N}^s(\boldsymbol{\xi})|$. Moreover, $\mathcal{A}^s(\boldsymbol{\xi})$ contains the following *combined* arcs:

- $(0, i_1^\alpha), (0, i_1^\beta)$, $(i_r^\alpha, 0'), (i_r^\beta, 0')$;
- $(i_k^\alpha, i_{k+1}^\alpha), (i_k^\alpha, i_{k+1}^\beta), (i_k^\beta, i_{k+1}^\alpha), (i_k^\beta, i_{k+1}^\beta)$, for each $k = 1, \ldots, r-1$.

The cost of each combined arc corresponds to the maximum duration between the corresponding bike leg's duration and the robot leg's duration, where the services and replenishment at the synchronization nodes are scheduled according to the meaning of the nodes i^α and i^β. We then calculate the shortest path between 0 and $0'$ on $\mathcal{G}^s(\boldsymbol{\xi})$ whose value corresponds to $\mathcal{T}^s(\boldsymbol{\xi})$. If $\mathcal{T}^s(\boldsymbol{\xi}) > \zeta$, the solution $\boldsymbol{\xi}$ is infeasible, and the corresponding no-good cut with $\mathcal{A}^b(\boldsymbol{\xi})$ and $\mathcal{A}^r(\boldsymbol{\xi})$ containing all the arcs involved in the bike and robot legs is added.

The separation of cuts (3) can be performed in polynomial time as described with the following illustrative example.

4.2.2. Illustrative example of no-good cuts (3). We further explain the no-good cuts (3) and their separation through Figures 3–7, which depict different solutions of a toy instance with seven customers. Bike legs are represented by solid lines, while robot legs by dotted lines. For the sake of simplicity, we suppose that replenishment operations take zero units of time and that the service at each customer takes one unit of time. Traveling times are reported next to the corresponding arcs. The maximum working time ζ equals 15. We also assume that all customers require a small parcel and can be visited by the robot. Finally, we suppose that the robot has two compartments.

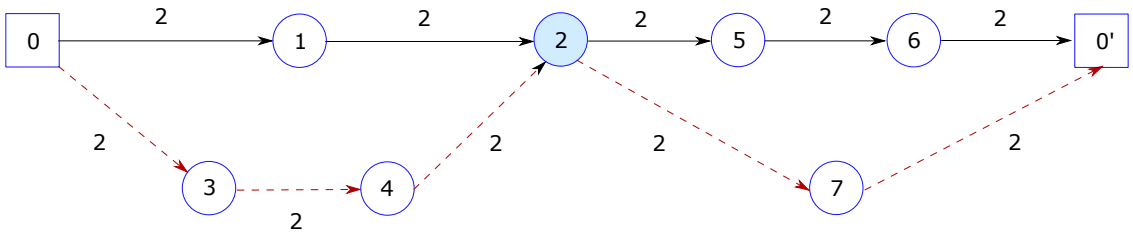

Figure 3 **A solution with two operations and $\tilde{\mathcal{T}}^s(\boldsymbol{\xi}) = 16$**

Figure 3 represents a solution consisting of two operations. In the first operation, the robot takes eight units of time to reach the rendezvous customer 2 while the bike only takes five units of time. In the second operation, the bike and the robot take eight and five units of time, resp., to reach the final depot after leaving the rendezvous customer 2. Thus, the value of $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})$ is 16 and the solution is infeasible as $\tilde{\mathcal{T}}^s(\boldsymbol{\xi}) > \zeta$. The following inequality cuts off this infeasible solution

$$x^b_{25} + x^b_{56} + x^b_{60'} + x^r_{03} + x^r_{34} + x^r_{42} \leq 5$$

Figure 4 represents another solution consisting of two operations. In the first operation, the robot takes seven units of time to reach the rendezvous customer 2 while the bike only takes 5 units

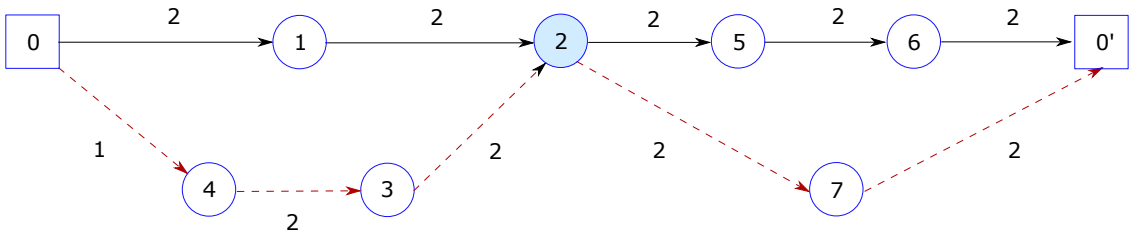

Figure 4 **A solution with two operations and $\tilde{\mathcal{T}}^s(\boldsymbol{\xi}) = 15$**

of time. In the second operation, the bike takes eight units of time to reach the final depot after leaving the rendezvous customer 2 and the bike just five units of time. Thus, the value of $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})$ is 15 in this case. Since $\tilde{\mathcal{T}}^s(\boldsymbol{\xi}) \leq \zeta$, we construct the support graph associated with this solution as depicted in Figure 5, and we compute the shortest path from 0 to $0'$ with a value of 15 (i.e., $0 \rightarrow 2^\alpha \rightarrow 0'$), indicating the solution to be feasible.

Note that the cost associated with arc $(0, 2^\alpha)$ is 7. In this case the service at 2 takes place before the replenishment. Thus, when the bike arrives at node 2 at time 5, it serves the customer and is ready for replenishment at 6. Replenishment operations start as soon as the robot arrives at 7, time at which both vehicles are ready to leave the node since $\eta = 0$ in the example. The cost associated with arc $(0, 2^\beta)$ is again 7. But in this case the operation time does not consider the service at 2, since this takes place after the replenishment. Thus, the cost of arc $(2^\beta, 0')$ is 9, as the bike takes 6 units of travel time to reach $0'$, plus 3 units of time to serve 2, 5 and 6. The corresponding robot leg takes 5 units of time. Finally, the cost of arc $(2^\alpha, 0')$ is 8, as the bike takes 6 units of time of travel time, plus 2 units to serve customers 5 and 6.

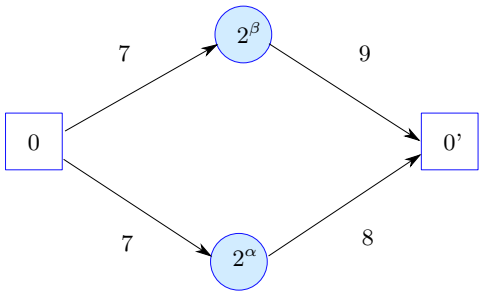

Figure 5 **The solution with $\mathcal{T}^s(\boldsymbol{\xi}) = 15$ is feasible**

Finally, in the example of Figure 6, the robot takes five units of time to reach the rendezvous customer 2 while the bike takes seven units of time. In the second operation, the bike takes 8 units of time to reach the final depot after leaving the rendezvous customer 2 while the robot only takes five units of time. Thus, the value of $\tilde{\mathcal{T}}^s(\boldsymbol{\xi})$ is 15. Since $\tilde{\mathcal{T}}^s(\boldsymbol{\xi}) \leq \zeta$, we construct the support graph associated with this solution as depicted in Figure 7, and we compute the shortest path from 0 to

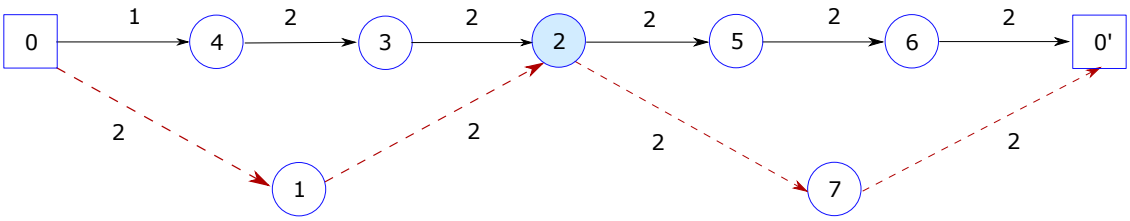

Figure 6 **A solution with two operations and $\tilde{\mathcal{T}}^s(\boldsymbol{\xi}) = 15$**

$0'$ with a value of 16, which proves the solution to be infeasible and has to be cut off. The following inequality cuts off this infeasible solution

$$x^b_{04} + x^b_{43} + x^b_{32} + x^b_{25} + x^b_{56} + x^b_{60'} + x^r_{01} + x^r_{12} + x^r_{27} + x^r_{70'} \leq 9$$

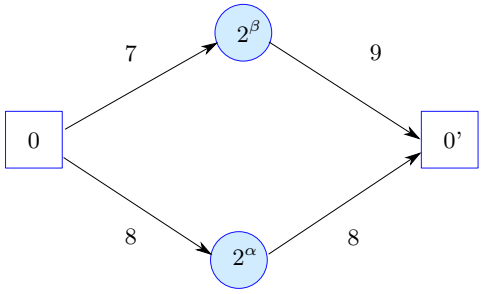

Figure 7 **The solution with $\mathcal{T}^s(\boldsymbol{\xi}) = 16$ is infeasible**

To summarize, formulation F_2 features objective function (1a) and constraints (1b)–(1i), (1p)–(1z), (2a)–(2d), and (3). Constraints (3) are exponentially many, so they cannot be enumerated upfront. However, formulation F_2 can be solved with a general-purpose MILP solver by starting with F_2 without any constraints (3) and then separate constraints (3) (as lazy cuts) on the integer solutions found in the search tree that violate the maximum working time constraint.

4.3. Valid Inequalities for F_1 and F_2

The linear relaxation of both formulations F_1 and F_2 can be tightened by adding the following two sets of valid inequalities, which build upon the well-known Generalized Subtour Elimination Constraints (GSEC), see, e.g., Fischetti, Salazar González, and Toth (1998)

$$\sum_{(i,j)\in\mathcal{A}\,:\,i\in\mathcal{S},\,j\in\mathcal{V}\setminus S} x^b_{ij} \geq y^b_k + y^s_k \qquad \mathcal{S}\subseteq\mathcal{N}\,:\,|\mathcal{S}|\geq 2,\,k\in\mathcal{S} \tag{5a}$$

$$\sum_{(i,j)\in\mathcal{A}^r\,:\,i\in\mathcal{S},\,j\in\mathcal{V}^r\setminus\mathcal{S}} x^r_{ij} \geq y^r_k + y^s_k \qquad \mathcal{S}\subseteq\mathcal{N}^r\,:\,|\mathcal{S}|\geq 2,\,k\in\mathcal{S} \tag{5b}$$

Constraints (5a) state that, for each subset of customers $\mathcal{S}\subseteq\mathcal{N}$ of cardinality at least two and each customer $k\in\mathcal{S}$, if the bike serves customer k (i.e., $y^b_k + y^s_k$ equals 1), then the bike must traverse at least one of the arcs emanating from $\mathcal{S}$ and terminating at $\mathcal{V}\setminus\mathcal{S}$. Constraints (5b) are similar

to constraints (5a) but apply to the customers visited by the robot. In the following, we refer to constraints (5a)–(5b) as GSECs.

Separation of GSECs (5). The number of GSECs (5) is exponential in the size of $\mathcal{N}$, so it is impossible to enumerate all of them for TSPBR instances of medium or even small size. Nevertheless, we can dynamically separate them in a branch-and-bound tree. Violated inequalities can be identified by solving a Max-Flow Problem (MFP).

Let $\overline{\boldsymbol{\xi}}$ denote an optimal solution of the linear relaxation of one of the two formulations F_1 or F_2. To separate constraints (5a), we create an auxiliary graph $\overline{\mathcal{G}}(\overline{\boldsymbol{\xi}}) = (\mathcal{V}, \overline{\mathcal{A}}(\overline{\boldsymbol{\xi}}))$ induced by solution $\overline{\boldsymbol{\xi}}$. Arc (i,j) belongs to $\overline{\mathcal{A}}(\overline{\boldsymbol{\xi}})$ if $\overline{x}^b_{ij} > 0$, and $\overline{x}^b_{ij}$ represents the capacity of arc (i,j) in the MFP. For each customer $i \in \mathcal{N}$, we solve a MFP on graph $\overline{\mathcal{G}}$, where 0 is the source and i is the sink of the flow. Then, we consider the partition $(\mathcal{S}, \mathcal{V} \setminus \mathcal{S})$ induced by the min-cut corresponding to the optimal max-flow, such that $0 \in \mathcal{V} \setminus \mathcal{S}$ and $i \in \mathcal{S}$. If the value of such min-cut is less than $\overline{y}^b_k + \overline{y}^s_k$, for some $k \in \mathcal{S}$, then a violated GSEC has been identified. Constraints (5b) can be separated similarly.

5. Heuristic Solutions for the TSPBR

In this section, we present a *Genetic Algorithm* (GA) to obtain high-quality solutions of the TSPBR. GAs are adaptive methods inspired by the natural evolution of biological organisms. An initial population of individuals, also called *chromosomes*, evolves through generations until a termination criterion is reached. The termination criterion can be based on quality and can be set as a maximum number of iterations or a time limit. New individuals (children) are generated from individuals belonging to the current generation (parents) by means of genetic operators (crossover and mutation). The principles of the genetic procedures were firstly formalized by Holland (1975) and have been successfully used in different contexts (Moscato and Cotta 2010). In the following, we present the different components of our GA.

5.1. Main Components of the GA Algorithm

5.1.1. Individual representation. As often done in the context of routing problems, an individual is a permutation of the locations that need to be visited (see, e.g., Lacomme, Prins, and Ramdane-Chérif (2001), Prins (2004), Vidal et al. (2012)).

5.1.2. Evaluation of individuals. Each individual is evaluated by means of the *AdSplit* procedure (see Section 5.2 for a detailed description). The role of *AdSplit* is to cut (split) the permutation in different bike and robot legs as well as to select the rendezvous customers. The procedure *AdSplit* is inspired by the split procedure used to convert a permutation of customers in solutions in the context of vehicle routing problems (see, e.g., Beasley (1983), Prins (2004)).

5.1.3. Initial population. The population is initialized with n_c randomly generated sequences. The *AdSplit* algorithm is then applied to each sequence to obtain a (non-necessarily feasible) TSPBR solution. The local search procedure described in Section 5.1.6 is finally applied to optimize these solutions (Vidal et al. 2012). These solutions are represented as sequences of delivery points obtained by concatenating bike and robot legs as they appear in the solution. These new sequences are then included in the population $\mathcal{P}$ along with the corresponding modified cost (see Section 5.1.6), which is also used as part of the computation of the biased fitness of the individual (see Section 5.1.8).

5.1.4. Children generation. To generate a child, two chromosomes are randomly drawn from the population, and the one with the lower biased fitness is selected to be one of the parents. The procedure is repeated twice, once for each parent (Prins 2004).

5.1.5. Children education. Each newly generated child undergoes, first, the *AdSplit* procedure to obtain a TSPBR solution, and, then, the local search for further improvements (Vidal et al. 2012). By concatenating the bike and robot legs as they appear in such a solution, a sequence of delivery points is obtained. This sequence is added to the population $\mathcal{P}$, and its cost is stored.

5.1.6. Local search. The local search procedure is based on the intra- and inter-leg relocation, intra- and inter-leg exchange, and the 2-opt operators. By exploiting the objective function of the TSPBR, the local search is implemented as follows. First, a solution $\boldsymbol{\xi}$ is evaluated with a modified cost, $\bar{c}(\boldsymbol{\xi})$, computed as $\bar{c}(\boldsymbol{\xi}) = c(\boldsymbol{\xi}) + \epsilon \cdot \max\{T(\boldsymbol{\xi}) - \zeta, 0\}$, where $T(\boldsymbol{\xi})$ is the total working time of $\boldsymbol{\xi}$ and $c(\boldsymbol{\xi})$ its cost, that is, the number of customers served by the bike.

A move is accepted if it reduces the modified cost of a solution. Notice that if $T(\boldsymbol{\xi}) \leq \zeta$, a move is accepted only if it reduces the number of customers served by the bike, which is impossible for intra-leg movements, or inter-leg relocations that relocate a customer from a robot leg to a bike leg. On the other hand, if $T(\boldsymbol{\xi}) > \zeta$, an intra-leg move or an inter-leg relocation that relocates a customer from a robot leg to a bike leg may be accepted if it reduces $T(\boldsymbol{\xi})$. This approach takes advantage of each iteration of the *AdSplit* procedure, even if the procedure provides an infeasible solution.

5.1.7. Crossover. The crossover operator used to determine a child is the classical order-crossover (OX) (Oliver, Smith, and Holland 1987).

5.1.8. Survivor strategy. Once n_g chromosomes have been added to the population (that is, $|\mathcal{P}| = n_c + n_g$), the best n_c individuals in terms of the value of their *biased fitness* are kept in the population while the other n_g are discarded. The biased fitness takes into account the quality of the individual (based on the cost of the corresponding solution) and the diversification contribution provided to the population. In particular, the diversification contribution of each individual is

evaluated with respect to the n_{close} closest chromosomes in the population. Finally, the best n_e individuals are always guaranteed to survive to the next population. This approach is proposed in Vidal et al. (2012) to which the reader is referred.

5.1.9. Termination criteria. The GA terminates if it has not improved the best-known solution for the last n_{stay} iterations or if the computational time limit has been reached. An iteration consists of children's generation and education.

5.1.10. Overall heuristic description. We first generate n_c individuals. Each individual undergoes the *AdSplit* and the local search procedures. Then, as long as the termination criteria are not met, we generate and educate children. When the population reaches the size of $n_c + n_g$, the survivor strategy is applied. When the termination criteria are met, the best solution found is returned. We refer to this procedure as $\mathcal{H}$.

5.2. Procedure *AdSplit*

The procedure *AdSplit* obtains a (non-necessarily feasible) TSPBR solution from a permutation of the nodes $\mathcal{V}$ such as $\boldsymbol{\sigma} = (\sigma_0 = 0, \sigma_1, \ldots, \sigma_n, \sigma_{n+1} = 0')$ that starts with 0, ends with $0'$, and features all customers $\mathcal{N}$ in between. *AdSplit* consists of two main steps (i.e., the definition of the support graph and the resolution of a constrained shortest path problem) that are detailed and illustrated through the TSPBR instance described in Example 1 below.

Along with the notation used so far, we need some further notation. Given two values i, j such that $0 \leq i < j \leq n+1$, let us define $\mathcal{N}^r_{ij}(\boldsymbol{\sigma}) = \{\sigma_k \in \mathcal{N}^r \,|\, k = i+1, \ldots, j-1\}$ and $\mathcal{N}^b_{ij}(\boldsymbol{\sigma}) = \{\sigma_k \in \mathcal{N}^b \,|\, k = i+1, \ldots, j-1\}$. Finally, let us define $\mathcal{N}_{ij}(\boldsymbol{\sigma}) = \mathcal{N}^r_{ij}(\boldsymbol{\sigma}) \cup \mathcal{N}^b_{ij}(\boldsymbol{\sigma})$.

EXAMPLE 1. We consider a TSPBR instance with six customers, where $\mathcal{N}^r = \{1, 2, 4, 5\}$ and $\mathcal{N}^b = \{3, 6\}$. The robot has three containers: two small ones and a large one. Customers 1 and 2 required small parcels whereas the parcels of customers 4 and 5 are large. The bike and the robot travel at the same speed, and travel times are reported in Table 1. The maximum working time is $\zeta = 12$, and the time for a replenishment is $\eta = 1$. The service time in each node is also one for each customer.

	1	2	3	4	5	6	0'
0	2	1	2	1	1		
1		2	2	1			2
2			1	2	1		3
3				1	2		
4					1	1	1
5						4	1
6							1

Table 1 Travel times of Example 1 – empty cells correspond to travel times not used in the example

5.2.1. Step1 – Definition of the support graph. To a permutation $\boldsymbol{\sigma}$, we associate an acyclic support multi-graph $\mathcal{G}_{\boldsymbol{\sigma}} = (\mathcal{V}_{\boldsymbol{\sigma}}, \mathcal{A}_{\boldsymbol{\sigma}})$. The node set $\mathcal{V}_{\boldsymbol{\sigma}}$ contains two nodes for each customer where the bike and the robot may have to synchronize plus the two copies of the depot, i.e., $\mathcal{V}_{\boldsymbol{\sigma}} = \{\sigma_i^\alpha, \sigma_i^\beta \,|\, \sigma_i \in \mathcal{N}^r,\, \mathcal{N}_{0i}^r(\boldsymbol{\sigma}) \neq \varnothing,\, \mathcal{N}_{i0'}^r(\boldsymbol{\sigma}) \neq \varnothing\} \cup \{\sigma_0^\alpha = 0, \sigma_{0'}^\alpha = 0'\}$. The arc set $\mathcal{A}_{\boldsymbol{\sigma}}$ features an arc $(\sigma_i^\gamma, \sigma_j^\delta)_{\sigma_k\sigma_\ell}$ (with $\sigma_i^\gamma, \sigma_j^\delta \in \mathcal{V}_{\boldsymbol{\sigma}}$ such that $i < j$, $\gamma, \delta \in \{\alpha, \beta\}$, and $\mathcal{N}_{ij}^r(\boldsymbol{\sigma}) \neq \varnothing$) for each feasible operation $\mathcal{O}_{\sigma_i^\gamma \sigma_j^\delta}^{\sigma_k \sigma_\ell}$ with $i < k \leq \ell < j$ such that $\sigma_k, \sigma_\ell \in \mathcal{N}_{ij}^r(\boldsymbol{\sigma})$, satisfying the following conditions:

C1) operation $\mathcal{O}_{\sigma_i^\gamma \sigma_j^\delta}^{\sigma_k \sigma_\ell}$ starts from σ_i, ends at σ_j, the bike and the robot synchronize at these two nodes, and σ_i and σ_j are served before (after, resp.) the replenishment if $\gamma, \delta = \alpha$ ($\gamma, \delta = \beta$, resp.);

C2) customers σ_k, σ_ℓ as well as all robot customers of the set $\mathcal{N}_{k\ell}^r(\boldsymbol{\sigma})$ are served by the robot in the order in which they appear in permutation $\boldsymbol{\sigma}$;

C3) all customers of the sets $\mathcal{N}_{ik}(\boldsymbol{\sigma})$, $\mathcal{N}_{\ell j}(\boldsymbol{\sigma})$, and $\mathcal{N}_{k\ell}^b(\boldsymbol{\sigma})$ are served by the bike in the order in which they appear in permutation $\boldsymbol{\sigma}$;

C4) the parcels of the customers assigned to the robot can be feasibly accommodated in the robot containers; notice that this feasibility check can easily be done by storing parcels in the smallest feasible container available;

C5) the operation is not dominated.

To better clarify how the support graph $\mathcal{G}_{\boldsymbol{\sigma}}$ is defined, let us refer to Example 1, and assume we consider permutation $\boldsymbol{\sigma} = (0, 1, 2, 3, 4, 5, 6, 0')$. The node set $\mathcal{V}_{\boldsymbol{\sigma}}$ is defined as $\mathcal{V}_{\boldsymbol{\sigma}} = \{0^\alpha, 2^\alpha, 2^\beta, 4^\alpha, 4^\beta, 0'^\alpha\}$. Notice that $\mathcal{V}_{\boldsymbol{\sigma}}$ does not contain any node corresponding to customers 1 and 5 as $\mathcal{N}_{01}^r(\boldsymbol{\sigma}) = \varnothing$ and $\mathcal{N}_{50'}^r(\boldsymbol{\sigma}) = \varnothing$. This decision of ruling out TSPBR solutions where the bike and the robot synchronize at the first or the last robot customer of a permutation stems from the observation that it is unlikely that an optimal TSPBR solution features an operation consisting of a robot leg that does not serve any robot customer – as a matter of fact, the TSPBR always features an optimal solution that does not include any robot leg without customers served by the robot if the travel time matrices $\boldsymbol{t}^b$ and $\boldsymbol{t}^r$ satisfy the triangle inequalities.

Table 2 reports all the arcs $(\sigma_i^\gamma, \sigma_j^\delta)_{\sigma_k\sigma_\ell}$ that satisfy conditions C1-C4. For each arc $(\sigma_i^\gamma, \sigma_j^\delta)_{\sigma_k\sigma_\ell}$, the table reports the indices k, ℓ, the robot and the bike leg of operation $\mathcal{O}_{\sigma_i^\gamma \sigma_j^\delta}^{\sigma_k \sigma_\ell}$, the corresponding cost c^o (equal to the number of customers served by the bike, including the last node of the leg, and excluding the first node of the leg), the robot travel time (t^r), the bike travel time (t^b), the total operation working time ($t^o = \max\{t^r, t^b\}$), and if the operation is dominated or not. The robot and the bike travel times include the replenishment time at the operation's start node.

The dominance (i.e., condition C5) between two arcs $(\sigma_{i_1}^{\gamma_1}, \sigma_{j_1}^{\delta_1})_{\sigma_{k_1}\sigma_{\ell_1}}$ and $(\sigma_{i_2}^{\gamma_2}, \sigma_{j_2}^{\delta_2})_{\sigma_{k_2}\sigma_{\ell_2}}$ of cost c_1^o, c_2^o, resp., and working time t_1^o, t_2^o, resp., is straightforward: if $\sigma_{i_1}^{\gamma_1} = \sigma_{i_2}^{\gamma_2}$, $\sigma_{j_1}^{\delta_1} = \sigma_{j_2}^{\delta_2}$, $c_1^o \leq c_2^o$, and

Table 2 **Arcs of the support graph $\mathcal{G}_{\boldsymbol{\sigma}}$ corresponding to Example 1 and permutation $\boldsymbol{\sigma} = (0, 1, 2, 3, 4, 5, 6, 0')$**

Arc	k	ℓ	Robot Leg	Bike Leg	c^o	t^r	t^b	t^o	Dominated
$(0^\alpha, 2^\alpha)_{11}$	1	1	$0^\alpha \to 1 \to 2^\alpha$	$0^\alpha \to 2^\alpha$	1	6	3	6	no
$(0^\alpha, 2^\beta)_{11}$	1	1	$0^\alpha \to 1 \to 2^\beta$	$0^\alpha \to 2^\beta$	1	6	2	6	no
$(0^\alpha, 4^\alpha)_{11}$	1	1	$0^\alpha \to 1 \to 4^\alpha$	$0^\alpha \to 2 \to 3 \to 4^\alpha$	3	5	7	7	no
$(0^\alpha, 4^\alpha)_{12}$	1	2	$0^\alpha \to 1 \to 2 \to 4^\alpha$	$0^\alpha \to 3 \to 4^\alpha$	2	9	6	9	no
$(0^\alpha, 4^\alpha)_{22}$	2	2	$0^\alpha \to 2 \to 4^\alpha$	$0^\alpha \to 1 \to 3 \to 4^\alpha$	3	5	9	9	yes (by $\mathcal{O}^{11}_{0^\alpha 4^\alpha}$)
$(0^\alpha, 4^\beta)_{11}$	1	1	$0^\alpha \to 1 \to 4^\beta$	$0^\alpha \to 2 \to 3 \to 4^\beta$	3	5	6	6	no
$(0^\alpha, 4^\beta)_{12}$	1	2	$0^\alpha \to 1 \to 2 \to 4^\beta$	$0^\alpha \to 3 \to 4^\beta$	2	9	5	9	no
$(0^\alpha, 4^\beta)_{22}$	2	2	$0^\alpha \to 2 \to 4^\beta$	$0^\alpha \to 1 \to 3 \to 4^\beta$	3	5	8	8	yes (by $\mathcal{O}^{11}_{0^\alpha 4^\beta}$)
$(0^\alpha, 0'^\alpha)_{11}$	1	1	$0^\alpha \to 1 \to 0'^\alpha$	$0^\alpha \to 2 \to 3 \to 4 \to 5 \to 6 \to 0'^\alpha$	5	6	15	15	yes (by $\mathcal{O}^{14}_{0^\alpha 0'^\alpha}$)
$(0^\alpha, 0'^\alpha)_{12}$	1	2	$0^\alpha \to 1 \to 2 \to 0'^\alpha$	$0^\alpha \to 3 \to 4 \to 5 \to 6 \to 0'^\alpha$	4	10	14	14	yes (by $\mathcal{O}^{14}_{0^\alpha 0'^\alpha}$)
$(0^\alpha, 0'^\alpha)_{14}$	1	4	$0^\alpha \to 1 \to 2 \to 4 \to 0'^\alpha$	$0^\alpha \to 3 \to 5 \to 6 \to 0'^\alpha$	3	11	13	13	no
$(0^\alpha, 0'^\alpha)_{22}$	2	2	$0^\alpha \to 2 \to 0'^\alpha$	$0^\alpha \to 1 \to 3 \to 4 \to 5 \to 6 \to 0'^\alpha$	5	6	17	17	yes (by $\mathcal{O}^{14}_{0^\alpha 0'^\alpha}$)
$(0^\alpha, 0'^\alpha)_{24}$	2	4	$0^\alpha \to 2 \to 4 \to 0'^\alpha$	$0^\alpha \to 1 \to 3 \to 5 \to 6 \to 0'^\alpha$	4	7	16	16	yes (by $\mathcal{O}^{14}_{0^\alpha 0'^\alpha}$)
$(0^\alpha, 0'^\alpha)_{44}$	4	4	$0^\alpha \to 4 \to 0'^\alpha$	$0^\alpha \to 1 \to 2 \to 3 \to 5 \to 6 \to 0'^\alpha$	5	4	18	18	yes (by $\mathcal{O}^{14}_{0^\alpha 0'^\alpha}$)
$(0^\alpha, 0'^\alpha)_{55}$	5	5	$0^\alpha \to 5 \to 0'^\alpha$	$0^\alpha \to 1 \to 2 \to 3 \to 4 \to 6 \to 0'^\alpha$	5	4	14	14	yes (by $\mathcal{O}^{14}_{0^\alpha 0'^\alpha}$)
$(2^\alpha, 0'^\alpha)_{44}$	4	4	$2^\alpha \to 4 \to 0'^\alpha$	$2^\alpha \to 3 \to 5 \to 6 \to 0'^\alpha$	3	5	12	12	yes (by $\mathcal{O}^{55}_{2^\alpha 0'^\alpha}$)
$(2^\alpha, 0'^\alpha)_{55}$	5	5	$2^\alpha \to 5 \to 0'^\alpha$	$2^\alpha \to 3 \to 4 \to 6 \to 0'^\alpha$	3	4	8	8	no
$(2^\beta, 0'^\alpha)_{44}$	4	4	$2^\beta \to 4 \to 0'^\alpha$	$2^\beta \to 3 \to 5 \to 6 \to 0'^\alpha$	3	5	13	13	yes (by $\mathcal{O}^{55}_{2^\beta 0'^\alpha}$)
$(2^\beta, 0'^\alpha)_{55}$	5	5	$2^\beta \to 5 \to 0'^\alpha$	$2^\beta \to 3 \to 4 \to 6 \to 0'^\alpha$	3	4	9	9	no
$(4^\alpha, 0'^\alpha)_{55}$	5	5	$4^\alpha \to 5 \to 0'^\alpha$	$4^\alpha \to 6 \to 0'^\alpha$	1	4	4	4	no
$(4^\beta, 0'^\alpha)_{55}$	5	5	$4^\beta \to 5 \to 0'^\alpha$	$4^\beta \to 6 \to 0'^\alpha$	1	4	5	5	no

$t^o_1 \leq t^o_2$ (with at least one of the two inequalities strictly satisfied), then operation $(\sigma^{\gamma_2}_{i_2}, \sigma^{\delta_2}_{j_2})_{\sigma_{k_2}\sigma_{\ell_2}}$ is dominated. Therefore, the arc set $\mathcal{A}_{\boldsymbol{\sigma}}$ consists of eleven arcs.

The support graph $\mathcal{G}_{\boldsymbol{\sigma}}$ is depicted in Figure 8, where empty circles represent bike customers and filled circles represent robot customers. Each arc starting from a node σ^γ_i and ending in a node σ^δ_j corresponds to an operation $\mathcal{O}^{\sigma_k\sigma_\ell}_{\sigma^\gamma_i\sigma^\delta_j}$, $i < k \leq \ell < j$ for which it is reported the operation cost c^o, the operation time t^o, and the values of k and ℓ in the format $(c^o, t^o)_{k,\ell}$. The initial and final depots are represented by squares.

The computational complexity of creating the support graph $\mathcal{G}_{\boldsymbol{\sigma}}$ is $O(n^4)$.

5.2.2. Step 2 – Resolution of a constrained shortest path problem on $\mathcal{G}_{\boldsymbol{\sigma}}$. The goal of the second step is to find the best TSPBR solution that can be derived from the support graph $\mathcal{G}_{\boldsymbol{\sigma}}$. To select the best sequence of operations and thus the synchronization points, we solve a shortest path problem with a resource constraint on the acyclic multi-graph $\mathcal{G}_{\boldsymbol{\sigma}}$. The objective is to minimize the number of customers served with the bike and the resource (i.e., the constraint) is the maximum working time of the solution. To this end, we need states indexed by the total working time and the last visited customer, which means we need up to $2|\mathcal{N}^r|\zeta$ states. As each state is extended at most $|\mathcal{N}^r|n$ times (corresponding to the maximum number of operations starting from a node), the computational complexity of solving this constrained shortest path problem is $O(n^3\zeta)$.

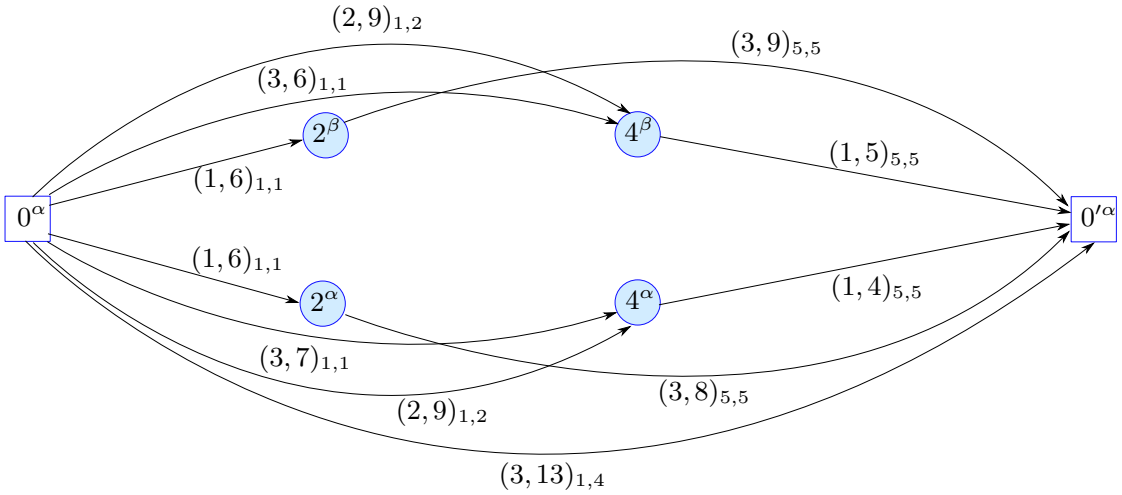

Figure 8 **Support graph $\mathcal{G}_\sigma$ corresponding to permutation $\sigma = (0,1,2,3,4,5,6,0')$**

Arcs	c^s	t^s	Feasible	$\overline{c}^s$
$(0^\alpha,2^\alpha)_{11}-(2^\alpha,0'^\alpha)_{55}$	4	14	no	$4+2\epsilon$
$(0^\alpha,2^\beta)_{11}-(2^\beta,0'^\alpha)_{55}$	4	15	no	$4+3\epsilon$
$(0^\alpha,4^\alpha)_{11}-(4^\alpha,0'^\alpha)_{55}$	4	11	yes	4
$(0^\alpha,4^\alpha)_{12}-(4^\alpha,0'^\alpha)_{55}$	3	13	no	$3+\epsilon$
$(0^\alpha,4^\beta)_{11}-(4^\beta,0'^\alpha)_{55}$	4	11	yes	4
$(0^\alpha,4^\beta)_{12}-(4^\beta,0'^\alpha)_{55}$	3	14	no	$3+2\epsilon$
$(0^\alpha,0'^\alpha)_{14}$	3	13	no	$3+\epsilon$

Table 3 **Set of (non-necessarily feasible) TSPBR solutions derived from the support graph $\mathcal{G}_\sigma$ of Figure 8**

Table 3 indicates all the TSPBR solutions explored in this step when considering the support graph $\mathcal{G}_\sigma$ of Figure 8. Table 3 reports, for each solution, the corresponding sequence of arcs, the cost (c^s), the total working time (t^s), if the solution is feasible, and the modified cost ($\overline{c}^s$). The modified cost is computed as $\overline{c}^s = c^s + \epsilon \cdot \max\{t^s - \zeta, 0\}$, where $\epsilon \geq 0$ is a parameter of the algorithm, which allows penalizing the infeasibility of the solutions.

The *AdSplit* procedure returns a solution with the lowest modified cost, which depends on the value of parameter ϵ. In the example, if $\epsilon < 1$, *AdSplit* returns one of the two (infeasible) solutions of cost 3 and duration 13, i.e., $(0^\alpha,4^\alpha)_{12}-(4^\alpha,0'^\alpha)_{55}$ and $(0^\alpha,0'^\alpha)_{14}$. On the other hand, if $\epsilon > 1$, *AdSplit* returns one of the two (feasible) solutions of cost 4 and duration 11, i.e., $(0^\alpha,4^\alpha)_{11}-(4^\alpha,0'^\alpha)_{55}$ and $(0^\alpha,4^\beta)_{11}-(4^\beta,0'^\alpha)_{55}$, both of which corresponds to robot legs $0\to1\to4$, $4\to5\to0'$, and bike legs $0\to2\to3\to4$, $4\to6\to0'$. If $\epsilon = 1$, *AdSplit* returns one of these four solutions.

Table 3 also shows that properly ordering the service and replenishment at rendezvous customers, to take advantage of the bike's idle time, can affect the total working time of a solution. For example, by comparing the solutions with synchronization at customer 2, i.e., $(0^\alpha,2^\alpha)_{11}-(2^\alpha,0'^\alpha)_{55}$ and $(0^\alpha,2^\beta)_{11}-(2^\beta,0'^\alpha)_{55}$, we can see that the former solution (where the bike serves customer 2 before the replenishment) has a lower working time than the latter (i.e., 15 vs 14).

6. Computational Experiments and Results

6.1. Data Description and Instances Generation

The company that inspired our study on the TSPBR (i.e., JD.com) has provided us with a data set of information that has allowed us to generate a set of realistic instances to test our algorithms. In particular, they have provided us with information about the customers and their orders, the robot and its containers, and the delivery costs. This information has allowed us to generate a set of benchmark instances as follows.

6.1.1. Customers and their orders. The collaboration with JD.com enabled us to access detailed information about $\approx 740\,000$ fulfilled orders in January 2022. For each order, they have provided us with the location of the corresponding customer (specifically, its latitude and longitude), the volume of the parcel delivered, the realized service time, and a code that identifies the deliveryman who performed the delivery.

6.1.2. Robots. JD.com has identified three types of robots that can be used for deliveries. The first type of robot features 24 containers: 12 small, 8 medium, and 4 large. The second type of robot features 12 containers: 6 medium and 6 large. The third and last type of robot features 8 large containers only. Figure 9 shows the three types of robots and a graphical representation of the containers on each side of the robot.

Figure 9 The three types of robots considered by JD.com: Type 1 (left), Type 2 (middle), and Type 3 (right)

6.1.3. Delivery costs. JD.com has provided us with an assessment of the delivery costs. In particular, they have estimated that the cost to serve a customer with the bike is 7 yuan whereas the cost to serve a customer with the robot is 1 yuan. Notice that these costs are consistent with

the assumption (see Section 3) that it is more expensive to serve a customer with the bike than the robot. Along with these two costs, JD.com has also estimated that they incur a fixed cost of 100 yuan per day to deploy the bike for deliveries and 150 yuan per day to deploy the robot. These fixed costs include, for example, maintenance and insurance costs, depreciation charges, and battery charging fees. The bike fixed cost also includes the biker's fixed daily salary, social welfare, paid holidays, etc. Being these costs fixed, they are not involved in the process of solving the TSPBR, but they are helpful in drawing some conclusions about the economic impact of deploying the robot along with the bike.

6.1.4. Generation of the test instances. To derive the graph of a single TSPBR instance, we randomly selected one of the codes associated with the deliverymen provided by JD.com and the customers served on one of the days of the available data. From the customer locations and the depot location (also provided by JD.com), expressed in the form of latitude and longitude, we could derive the travel time matrices by first computing geographical distances and then assuming, as advised by JD.com, a travel speed of 6 m/sec (i.e., 21.6 km/hour) for the bike and 5 m/sec (i.e., 18 km/hour) for the robot - travel times are expressed in seconds and rounded up if a fractional value was obtained. The replenishment time η is set equal to 600 seconds. We have also explicitly considered the customer service time provided by JD.com for these delivery tasks. To select the subset of customers that must be served by the bike, we randomly picked 20% of the customers. The compatibility of the remaining 80% of the customers with the robot containers is established based on the volume of the corresponding parcels and the volume of the robot containers. Finally, to obtain a valid upper bound ω on the required time to serve all customers, we assume to only have the bike available for deliveries and solve the corresponding TSP on the resulting instance with a time limit of one hour. We ended up with 100 graphs featuring between 35 and 60 nodes. We then considered each of these graphs in nine different parameter settings, one for each robot type and three different values of maximum working time $\zeta \in \{0.65, 0.80, 0.95\} \cdot \omega$, corresponding to a 35%, 20% and 5% reduction of working time, respectively. Therefore, with such configurations, the set of benchmark instances we considered consists of 900 instances.

6.2. Computational Results

In this section, we report on the computational results achieved by solving the TSPBR with the following algorithms:

- $\mathcal{F}_1$ corresponds to F_1 solved with CPLEX;
- $\mathcal{F}_2$ corresponds to F_2 solved with CPLEX with constraints (3) separated as lazy cuts;
- $\mathcal{F}_1^+$ and $\mathcal{F}_2^+$ indicate $\mathcal{F}_1$ and $\mathcal{F}_2$ with the additional GSECs (5) separated as user cuts;
- $\mathcal{H}$ corresponds to the genetic algorithm described in Section 5;

- $\mathcal{F}_1^{\mathcal{H}}$, $\mathcal{F}_2^{\mathcal{H}}$, $\mathcal{F}_1^{\mathcal{H},+}$, and $\mathcal{F}_2^{\mathcal{H},+}$ indicate algorithms $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_1^+$, and $\mathcal{F}_2^+$, resp., in which the model is warm-started with the best solution found by the genetic algorithm $\mathcal{H}$.

All the experiments reported in this section have been performed on a computer equipped with a 2.20 GHz AMD Ryzen Threadripper Pro 3955wx 16-Core CPU and 64 GB of RAM, running a 64-bit Linux operating system. The algorithms were coded in C++, and the source codes were compiled with gcc 9.4.0 and -O3 optimization flag. Version 20.1.0 of CPLEX was used, and the default parameter setting has always been used. In all experiments, a time limit of one hour was imposed on each algorithm to solve each instance.

6.2.1. Computational performance of $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_1^+$, and $\mathcal{F}_2^+$. In this section, we report on the computational performance of $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_1^+$, and $\mathcal{F}_2^+$, and compare their effectiveness to solve the TSPBR to optimality. We used the whole set of 900 instances to test the four algorithms with a time limit of one hour on each run.

The results are summarized in Table 4. The top, middle, and bottom part of the table reports the results when setting $\zeta = 0.95/0.80/0.65 \cdot \omega$, respectively. The first two columns indicate the number of nodes ($\mathcal{V}$) and the number of instances (#) considered in the corresponding row. For each of the four algorithms, Table 4 reports the number of instances solved to proven optimality (opt), the average percentage gap between the final upper bounds and the cost of the best-known solutions ($\mathsf{gap_u}$), the average percentage gap between the final lower bounds and the cost of the best-known solutions ($\mathsf{gap_l}$), and the average computing time in seconds (cpu). The percentage gaps of an instance, achieved by each of the algorithms, are computed as $gap_u = 100 \cdot \frac{ub - bks}{bks}$ and $gap_l = 100 \cdot \frac{bks - lb}{bks}$, where ub is the final best upper bound, lb is the final best lower bound, and bks is the cost of the best-known solution computed by any of the algorithms.

Table 5 shows that algorithm $\mathcal{F}_2^+$ can solve 713 of the 900 instances to optimality and outperforms the other three algorithms in terms of instances solved to optimality, final gaps, and average computing times. We can also observe that instances with the maximum working time equal to $0.65 \cdot \omega$ are, on average, more difficult to solve to optimality with any of the algorithms: fewer instances can be closed, final gaps are higher, and the average computing time increases. Moreover, the higher the number of customers is, the more difficult the instances are.

When comparing the results of $\mathcal{F}_1$ and $\mathcal{F}_2$, we can see that they achieve similar results even though the final upper bounds provided by $\mathcal{F}_2$ are, on average, significantly better. Furthermore, Table 4 shows that separating GSEC (5) is usually beneficial; indeed, $\mathcal{F}_1^+$ and $\mathcal{F}_2^+$ can solve 3 and 18 instances more than $\mathcal{F}_1$ and $\mathcal{F}_2$, resp., in lower average computing times.

Table 4 Computational comparison between $\mathcal{F}_1$, $\mathcal{F}_1^+$, $\mathcal{F}_2$, and $\mathcal{F}_2^+$

			$\mathcal{F}_1$				$\mathcal{F}_1^+$				$\mathcal{F}_2$				$\mathcal{F}_2^+$			
	$\lvert\mathcal{V}\rvert$	#	opt	gap_u	gap_l	cpu	opt	gap_u	gap_l	cpu	opt	gap_u	gap_l	cpu	opt	gap_u	gap_l	cpu
$\zeta = 0.95 \cdot \omega$	35	60	60	0.00	0.00	11	60	0.00	0.00	24	60	0.00	0.00	15	60	0.00	0.00	12
	40	60	60	0.00	0.00	96	60	0.00	0.00	84	60	0.00	0.00	122	60	0.00	0.00	55
	45	60	60	0.00	0.00	232	59	0.19	0.00	227	58	0.24	0.15	275	60	0.00	0.00	240
	50	45	44	0.25	0.00	309	43	2.96	0.25	312	45	0.00	0.00	269	45	0.00	0.00	207
	55	45	37	6.79	1.09	1137	36	10.41	0.94	1364	39	2.76	0.87	1125	41	9.64	0.49	1063
	60	30	23	10.32	1.34	1438	26	18.19	0.96	1182	23	16.97	1.31	1208	27	19.92	1.02	1098
		300	284	2.06	0.30	428	284	3.86	0.27	437	285	2.11	0.29	412	293	3.44	0.18	362
$\zeta = 0.80 \cdot \omega$	35	60	59	0.00	0.19	96	60	0.00	0.00	102	59	0.00	0.19	145	60	0.00	0.00	71
	40	60	57	0.19	0.35	361	58	0.00	0.39	379	57	0.14	0.37	437	58	0.00	0.37	432
	45	60	56	0.27	0.45	824	53	1.16	0.86	895	55	0.23	0.69	815	59	0.00	0.17	766
	50	45	37	3.65	1.46	1266	36	3.73	1.51	1243	38	2.28	1.36	1212	38	4.05	1.36	1168
	55	45	19	31.36	2.86	2398	22	28.83	1.95	2410	26	5.38	1.97	2412	26	12.99	2.03	2124
	60	30	13	39.52	2.97	2730	12	46.66	3.19	2798	14	10.71	3.19	2742	15	18.75	2.71	2392
		300	241	9.12	1.14	1079	241	9.34	1.09	1103	249	1.92	1.07	1097	256	4.12	0.89	987
$\zeta = 0.65 \cdot \omega$	35	60	55	1.54	0.42	554	57	1.78	0.18	530	58	0.47	0.21	386	60	0.00	0.00	474
	40	60	46	2.90	1.49	1098	48	5.93	1.33	1164	48	1.54	1.66	1069	52	0.41	1.44	878
	45	60	24	3.27	8.14	2546	21	3.56	8.26	2742	23	1.54	8.48	2573	23	3.26	8.25	2604
	50	45	15	8.23	8.59	2768	19	5.95	8.30	2495	17	1.26	8.40	2708	17	3.13	8.77	2577
	55	45	9	19.33	14.37	3219	9	21.78	13.96	3264	9	14.03	14.39	3265	6	25.11	14.55	3240
	60	30	6	13.39	13.81	3238	4	40.58	13.87	3324	6	13.94	13.94	3356	6	20.48	13.63	3396
		300	155	5.56	6.84	2061	158	7.89	6.68	2083	161	3.18	6.88	2037	164	4.70	6.80	2003
All		900	680	5.58	2.76	1190	683	6.94	2.68	1208	695	2.33	2.75	1182	713	4.03	2.62	1117

6.2.2. Computational performance of $\mathcal{H}$. We first tuned the parameters of algorithm $\mathcal{H}$. In particular, we considered the following values for the six parameters of the algorithm:

- number of chromosomes kept in the population: $n_c \in \{10, 15, 20, 25, 30\}$;
- number of new chromosomes included in the population: $n_g \in \{n_c, 2n_c, 3n_c\}$;
- number of elite individuals selected to survive and being carried over into the next generation: $n_e \in \{0.2n_c, 0.3n_c, 0.4n_c\}$
- number of close chromosomes to the inspected individual: $n_{close} \in \{0.2n_c, 0.3n_c, 0.4n_c\}$;
- number of iterations without improvement of the best solution: $n_{stay} \in \{1, 3, 5\}$;
- penalty of the working time violation: $\epsilon \in \{10, 15, 20\}$.

Overall, we considered 1215 combinations of parameters, which we tested on 20 instances randomly selected from our testbed of 900 instances. In particular, we selected four instances with $|\mathcal{N}| \in \{35, 40, 45, 50, 55\}$ each. As to the termination criterion, the algorithm $\mathcal{H}$ terminates when either it does not improve the best solution for n_{stay} iterations or the one-hour time limit is reached.

To select the best parameter combination, we first excluded the configurations that could not provide a feasible solution on all 20 instances. Then, we normalized (with a unity-based normalization) the average gaps and run times of the remaining configurations. Finally, we computed a score for

each parameter combination by assigning equal weights to these two criteria. The best configuration turned out being the one with $n_c = 15, n_g = 15, n_e = 3, n_{close} = 4, n_{stay} = 1, \epsilon = 10$. Therefore, we used this configuration in all the tests reported in the remainder of this section.

Then, we tested the heuristic algorithm $\mathcal{H}$ on all 900 generated instances and summarized its performance in Table 5. Each row of the table indicates average values over the subset of instances featuring the number of nodes reported in the first column ($|\mathcal{V}|$). The second column (#) reports the number of instances per row of the table. Then, for each of the three possible values of ζ (i.e., $\zeta \in \{0.65, 0.80, 0.95\} \cdot \omega$), Table 5 reports the number of instances on which $\mathcal{H}$ found an optimal solution (opt), the average percentage gap with respect to the best-known solutions (gap), and the average computing time in seconds (cpu). We can indicate the number of times $\mathcal{H}$ finds an optimal solution because we have the results of $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_1^+$, $\mathcal{F}_2^+$, $\mathcal{F}_1^{\mathcal{H}}$, $\mathcal{F}_2^{\mathcal{H}}$, $\mathcal{F}_1^{\mathcal{H},+}$, and $\mathcal{F}_2^{\mathcal{H},+}$, on which we will report in the next sections. As to the gaps, we calculated them with respect to the best-known solution we found for each instance in the entire computational campaign we conducted (no matter the algorithm that found such solution). In particular, the gap of an instance is computed as $gap = 100 \cdot \frac{ub - bks}{bks}$, where ub is the upper bound returned by $\mathcal{H}$ and bks is the cost of the best-known solution. The average values reported in columns gap and cpu are computed over all instances in the corresponding row.

Table 5 **Summary of the performance of genetic algorithm $\mathcal{H}$ over all 900 test instances**

		$\zeta = 0.95 \cdot \omega$			$\zeta = 0.80 \cdot \omega$			$\zeta = 0.65 \cdot \omega$		
$\lvert\mathcal{V}\rvert$	#	opt	gap	cpu	opt	gap	cpu	opt	gap	cpu
35	60	60	0.00	8	59	0.17	9	39	5.78	10
40	60	58	0.44	17	53	1.14	19	40	3.12	22
45	60	55	1.06	29	47	1.88	32	21	4.75	190
50	45	42	0.62	46	37	1.11	53	17	4.69	197
55	45	37	2.02	84	29	2.41	101	16	4.05	438
60	30	28	0.21	114	21	1.41	138	7	2.61	591
All	300	280	0.72	42	246	1.31	49	140	4.30	199

Table 5 shows that $\mathcal{H}$ can find an optimal solution on 666 of the 900 instances with an average computing time of less than 100 seconds and, on average, the best solution returned is within 2.11% from the best-known solutions. We can also observe that instances with a tighter working time (i.e., $\zeta = 0.65 \cdot \omega$) are generally more difficult: $\mathcal{H}$ can find fewer optimal solutions, takes more time, and has higher final gaps.

6.2.3. Effect of warm-starting $\mathcal{F}_1$, $\mathcal{F}_1^+$, $\mathcal{F}_2$, and $\mathcal{F}_2^+$ with the best solution obtained by $\mathcal{H}$. Table 4 shows that $\mathcal{F}_1$, $\mathcal{F}_1^+$, $\mathcal{F}_2$, and $\mathcal{F}_2^+$ struggle at finding high-quality primal solutions; as a matter of fact, the best algorithm (i.e., $\mathcal{F}_2^+$) yields upper bounds that are, on average, 4.03%

far from the cost of the best-known solutions. For this reason, we tested $\mathcal{F}_1$, $\mathcal{F}_1^+$, $\mathcal{F}_2$, and $\mathcal{F}_2^+$ by warm-starting them with the best primal solution obtained by $\mathcal{H}$.

The results achieved are summarized in Table 6, which features the same columns as Table 4. As before, we imposed a time limit of one hour on each algorithm to solve each instance, but this time limit includes the computing time taken by $\mathcal{H}$. For ease of comparison, the last row of the table indicates the overall results achieved by $\mathcal{F}_1$, $\mathcal{F}_1^+$, $\mathcal{F}_2$, and $\mathcal{F}_2^+$ taken from Table 4.

Table 6 Computational comparison between $\mathcal{F}_1^{\mathcal{H}}$, $\mathcal{F}_1^{\mathcal{H},+}$, $\mathcal{F}_2^{\mathcal{H}}$, and $\mathcal{F}_2^{\mathcal{H},+}$

			$\mathcal{F}_1^{\mathcal{H}}$				$\mathcal{F}_1^{\mathcal{H},+}$				$\mathcal{F}_2^{\mathcal{H}}$				$\mathcal{F}_2^{\mathcal{H},+}$			
	$\lvert\mathcal{V}\rvert$	#	opt	gap_u	gap_l	cpu	opt	gap_u	gap_l	cpu	opt	gap_u	gap_l	cpu	opt	gap_u	gap_l	cpu
$\zeta = 0.95 \cdot \omega$	35	60	60	0.00	0.00	11	60	0.00	0.00	14	60	0.00	0.00	14	60	0	0.00	13
	40	60	59	0.28	0.00	87	60	0.00	0.00	37	60	0.00	0.00	38	60	0	0.00	51
	45	60	60	0.00	0.00	137	60	0.00	0.00	111	60	0.00	0.00	83	60	0	0.00	96
	50	45	45	0.00	0.00	98	45	0.00	0.00	98	45	0.00	0.00	52	45	0	0.00	50
	55	45	44	0.20	0.09	461	44	0.20	0.00	458	44	0.22	0.00	326	45	0	0.00	449
	60	30	28	0.26	0.63	362	29	0.00	0.33	304	29	0.00	0.33	263	29	0	0.33	277
		300	296	0.11	0.08	167	298	0.03	0.03	146	298	0.03	0.03	110	299	0	0.03	134
$\zeta = 0.80 \cdot \omega$	35	60	60	0.00	0.24	84	60	0.00	0.00	73	60	0.00	0.00	32	60	0.00	0.00	32
	40	60	60	0.00	0.41	318	60	0.00	0.12	293	60	0.00	0.32	297	60	0.00	0.12	248
	45	60	59	0.13	0.15	342	59	0.13	0.18	342	59	0.13	0.33	444	60	0.00	0.00	365
	50	45	39	0.76	1.43	932	40	0.56	1.01	823	43	0.00	0.63	701	43	0.00	0.68	576
	55	45	36	1.24	1.59	1499	38	0.74	1.61	1480	38	0.95	1.34	1472	41	0.13	1.03	1287
	60	30	24	1.05	1.46	1207	27	0.21	1.25	1193	27	0.21	1.32	1123	26	0.42	1.08	1196
		300	278	0.43	0.76	634	284	0.24	0.58	606	287	0.19	0.56	593	290	0.06	0.39	528
$\zeta = 0.65 \cdot \omega$	35	60	57	0.60	0.39	459	57	0.60	0.24	426	59	0.22	0.11	430	59	0.12	0.00	510
	40	60	53	1.11	0.92	835	54	0.99	0.76	769	50	1.71	1.06	855	55	0.00	0.78	706
	45	60	33	1.79	7.33	2332	33	1.77	7.47	2405	33	1.29	7.43	2297	32	1.76	7.87	2509
	50	45	22	2.23	8.12	2385	22	2.57	8.07	2381	25	2.63	8.40	2465	27	2.24	7.67	2124
	55	45	20	3.06	13.63	2600	21	2.10	13.40	2581	20	0.70	13.85	2627	20	3.48	13.48	2520
	60	30	11	4.70	13.08	2759	12	3.60	13.62	2954	10	2.85	13.39	2828	11	1.90	13.13	2838
		300	196	1.96	6.30	1749	199	1.73	6.28	1760	197	1.43	6.40	1763	204	1.42	6.22	1725
All		900	770	0.79	2.41	850	781	0.67	2.30	837	782	0.55	2.33	822	793	0.49	2.12	799
			vs $\mathcal{F}_1$				vs $\mathcal{F}_1^+$				vs $\mathcal{F}_2$				vs $\mathcal{F}_2^+$			
			680	5.58	2.76	1190	683	6.94	2.68	1208	695	2.33	2.75	1182	713	4.03	2.62	1117

Table 6 shows that all the algorithms benefit from the warm-start: they closed more instances, and achieved lower final gaps, in lower average computing times. Even though the performance of the warm-started algorithms is quite similar, we can see that $\mathcal{F}_2^+$ still outperforms the other algorithms.

To easily compare the performance of the algorithms, we visualize their performance in the performance profile of Figure 10. This performance profile reports the computing time in seconds,

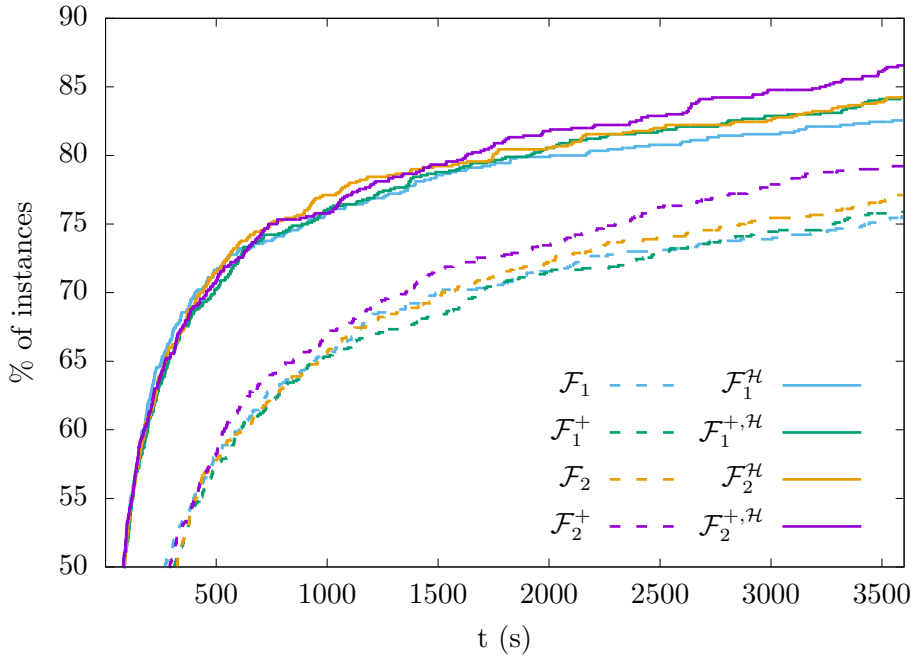

Figure 10 **Performance profile of the algorithms $\mathcal{F}_1$, $\mathcal{F}_1^+$, $\mathcal{F}_2$, $\mathcal{F}_2^+$, $\mathcal{F}_1^{\mathcal{H}}$, $\mathcal{F}_1^{\mathcal{H},+}$, $\mathcal{F}_2^{\mathcal{H}}$, and $\mathcal{F}_2^{\mathcal{H},+}$: percentage of instances solved to optimality within given computing times**

in the horizontal axis, and the cumulative percentage of instances being solved to optimality within a given CPU time, in the vertical axis.

Figure 10 shows that $\mathcal{F}_2^{\mathcal{H},+}$ enhanced with GSECs (5) and warm-started with the best solution provided by $\mathcal{H}$ achieves the best performances. Moreover, warm-starting the algorithms allows solving to proven optimality almost 70% of the instances within 300 seconds of computing time.

6.3. Managerial Insights

To the best of our knowledge, JD.com has not conducted any pilot project to assess the benefits of last-mile deliveries with a bike aided by a robot yet. Therefore, in this section, we show some potential benefits by assessing the impact of introducing self-driving robots in daily distribution.

6.3.1. Results on a sample instance. For a clear comparison of optimal TSPBR solutions in different scenarios (without the self-driving robot or with the robot of one of the three types considered), we selected a sample instance with $|\mathcal{V}| = 50$ from the benchmark set. For a quantitative analysis of these optimal solutions, JD.com estimated the purchasing cost of a robot to be 200 000 yuan, and this cost is amortized over four years. Therefore, by assuming that the robot can be deployed in 330 work days per year, the daily depreciation charge is about 150 yuan per work day.

The main features of optimal solutions under the 10 scenarios are summarized in Table 7, which indicates the number of customers served by the bike and the robot, the number of replenishments (including the initial one at the depot), the total costs (including the assumed 150 yuan of

depreciation charge), and the total working time. The total cost equals the sum of the service cost by the bike (i.e., 7×# customers served by bike + daily fixed cost to deploy a bike of 100) and by the robot (i.e., 1×# customers served by robot + daily depreciation of 150).

Table 7 Comparison of the optimal solutions on the sample 50-node instance under different scenarios

		$\zeta = 0.95 \cdot \omega$			$\zeta = 0.80 \cdot \omega$			$\zeta = 0.65 \cdot \omega$		
	No Robot	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
#customers served by bike	48	7	9	10	10	11	13	16	17	18
#customers served by robot	0	41	39	38	38	37	35	32	31	30
#replenishments	0	2	4	5	2	3	5	3	3	3
Costs (incl. depreciation)	436	340	352	358	358	364	376	394	400	406
Working time	413	374	371	377	328	329	327	260	265	263

Table 7 clearly shows that there can be a significant reduction of costs whenever the robot is deployed no matter which type of robot is used. Moreover, these cost reductions can be attained with an $8\% \sim 37\%$ saving of working time taken by the bike when the robot is not deployed.

6.3.2. Expected cost savings. Table 8 summarizes the average cost savings, over all the benchmark instances, when deploying a robot of each of the three types and setting the maximum working time to $\{0.95, 0.80, 0.65\} \cdot \omega$ minutes. Average cost savings are computed in percentage considering the cost incurred using the bike only as a base – notice that the cost of serving all customers with the bike only for an instance with n customers is $7n + 100$.

Table 8 Average percentage of cost savings, over all benchmark instances, by using robots compared to using the bike only

	$\zeta = 0.95 \cdot \omega$			$\zeta = 0.80 \cdot \omega$			$\zeta = 0.65 \cdot \omega$		
$\lvert\mathcal{V}\rvert$	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
35	3.44	2.36	0.82	1.72	0.36	-1.09	-3.08	-4.89	-6.71
40	10.90	9.51	8.20	8.11	6.64	5.16	3.20	1.72	0.41
45	15.86	14.36	12.72	12.94	11.37	10.02	6.81	3.82	-0.60
50	20.28	18.62	16.79	17.98	16.15	14.13	11.83	8.53	3.39
55	26.67	24.12	21.91	23.10	20.38	19.36	16.73	11.30	3.06
60	27.63	25.85	23.72	25.61	23.12	20.87	19.45	14.82	5.34
	15.85	14.24	12.52	13.28	11.47	9.93	7.61	4.58	0.12

From Table 8, we can observe that the average cost savings are in the range $-7\% \sim 28\%$, depending on the time-saving reduction target and the deployed robot type. Higher cost savings can be expected when the number of customers to serve increases and if the robot features more containers (see, e.g., type-1 and type-2 robots compared to type-3 robots). No matter the type of robot used, cost savings are expected in instances with 50 or more nodes.

Table 8 also indicates that, by introducing the self-driving robot, working time savings can be significant while still decreasing total costs. Indeed, by reducing the working-time restriction by 35% (e.g., from 8 hours to about 5 hours), the average cost savings can be as much as 7.61% with Type 1 robots; if the robot is deployed to help busy deliverymen (for example, bikes with more than 50 customers per day), cost savings can reach 19%. Besides, if the working time-saving requirement is not so ambitious, for example, a 20% reduction, the average cost saving will be $13.28\%, 11.47\%$, and 9.93% for the three types of robots, resp., which is still promising to most of the e-commerce companies. In particular, this is crucial for JD.com as the number of their parcel delivery requests is increasing at an annual rate of 30% (Wang 2022b). These time savings can be valuable for JD.com as they could (*a*) allow bikers to take extra breaks or earn more by serving additional customers, (*b*) better cope with spikes in their sales and subsequent deliveries on special days such as Chinese Black Friday (on June 18, August 18, and November 11) when daily sales may double (e.g., Wang 2022a), and (*c*) in the long term, reallocate their workforce to other departments or tasks and deal with increases in sales and deliveries.

6.3.3. Sensitivity to cost values. Possible cost savings resulting from adopting SDRs in a TSPBR setting depend on the expected costs for serving customers with the bike and the robot, i.e., c^b and c^r. Based on the expectations of JD.com, we have previously assumed $c^b = 7$ and $c^r = 1$. We now show what happens if c^b and c^r vary. In particular, we consider nine combinations of values with $c^b \in \{5, 7, 10\}$ and $c^r \in \{0.5, 1.0, 1.5\}$. For each of these combinations and for each of the 100 graphs previously described, we compute the cost of the best-known TSPBR solution when $\zeta = 0.8 \cdot \omega$ for any of the three robot types. Figure 11 presents the box-plot distributions of cost-savings (in percentage), compared to the bike-only solution, using type-1 robot (white bars), type-2 robot (green bars), and type-3 robot (red bars).

Figure 11 suggests that: (*a*) the benefits of deploying robots increase with the ratio between c^b and c^r (e.g., if $c^b = 10$ and $c^r = 0.5$, deploying type-1 robots brings up to 50% of cost savings), and (*b*) if $c^b = 5$, there may not be any cost savings in most cases, especially if type-2 and type-3 robots are deployed.

7. Conclusions

Motivated by the challenges faced by the Chinese e-commerce giant JD.com in last-mile delivery, we have addressed a delivery problem where a bike and a self-driving robot work in tandem to deliver parcels to customers in urban areas. We called this new problem the Traveling Salesman Problem with Bike-and-Robot (TSPBR). The decisions entailed by the TSPBR are (*i*) partitioning the customers to serve between the bike and the robot, (*ii*) deciding upon the locations where the two vehicles synchronize so that the robot is replenished, and (*iii*) routing the two vehicles.

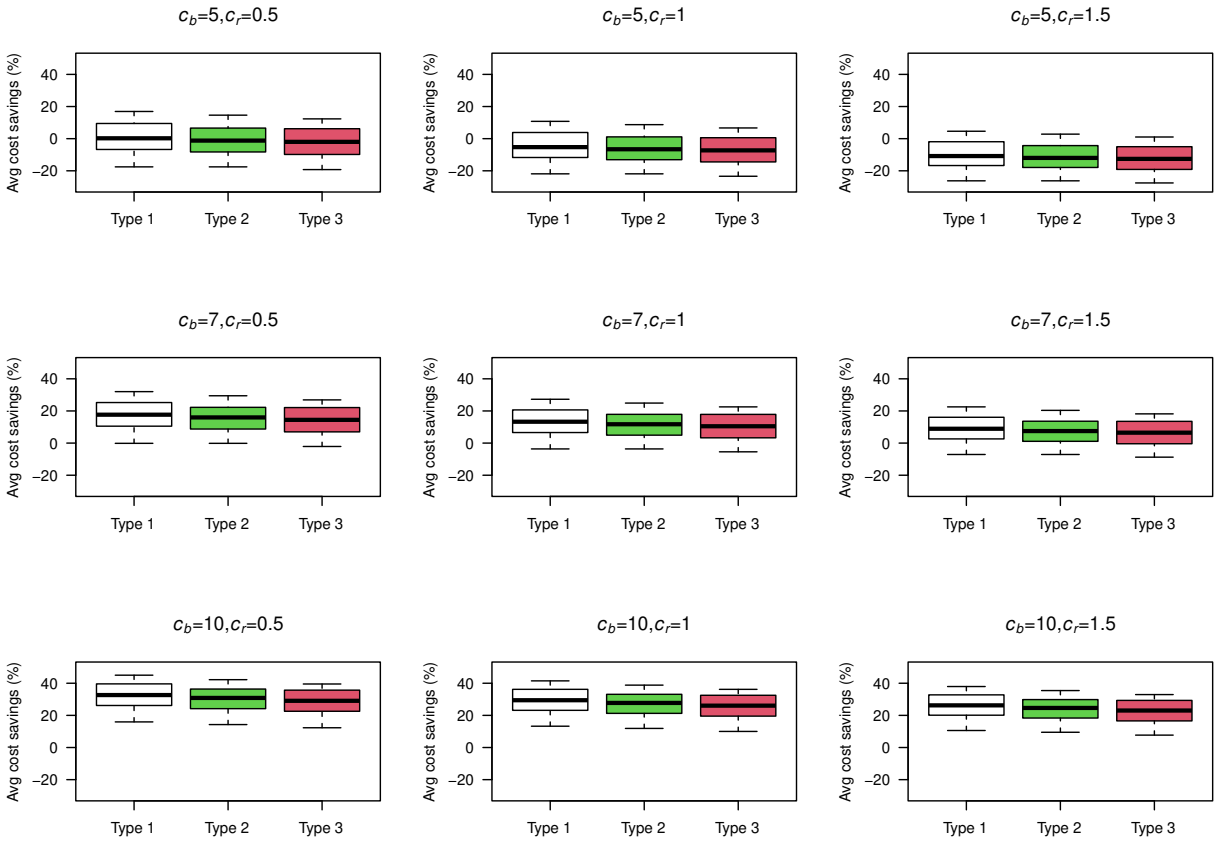

Figure 11 **Cost savings distributions assuming different combinations of cost values ($c_b \in \{5, 7, 10\}$ and $c_r \in \{0.5, 1, 1.5\}$), when $\zeta = 0.8 \cdot \omega$, with the three robot types (1-white, 2-green, 3-red)**

The main challenge in mathematically modeling the TSPBR is the needed spatial and temporal synchronization of the two vehicles.

We have introduced two mixed-integer linear programming formulations that model the synchronization constraints with different approaches. In the first formulation, synchronization constraints are ensured through a polynomial number of constraints. In the second formulation, synchronization is guaranteed with an exponential number of constraints generated on the fly. To strengthen the linear relaxation of both formulations, we have also introduced a set of valid inequalities derived from the well-known generalized subtour elimination constraints.

To have an algorithm that can scale up and solve large TSPBR instances, we have also proposed a genetic algorithm. As often done in the TSP literature, chromosomes are represented as permutations of customers. To efficiently explore large neighborhoods, we have devised a dynamic programming-based procedure.

We have tested the formulations and the genetic algorithm on a set of benchmark instances based on real data provided by JD.com. The formulations can often find optimal solutions for

instances with up to 60 nodes in some ten minutes of computing time. The genetic algorithm can yield high-quality solutions, within $\sim 3\%$ to optimality, in a few minutes of computing time. Finally, we have shown that significant cost savings can be achieved by deploying the robot along with the bike rather than assigning all deliveries to the bike alone.

We hope that our study can inspire other researchers and practitioners to explore the potential benefits of adopting robots in last-mile deliveries, which is still a partly unexplored research area. We envision several interesting ways to extend the TSPBR problem setting we consider: multiple bikes and robots can be investigated, different types of collaboration among the vehicles can be adopted, and customer-oriented constraints could also be embedded in the model. From a methodological perspective, all these features bring an extra layer of complexity to the TSPBR that would require an effort to develop new solution methods that can address real-size instances.

References

Agatz N, Bouman P, Schmidt M, 2018 *Optimization Approaches for the Traveling Salesman Problem with Drone. Transportation Science* 52(4):965–981.

Alfandari L, Ljubić I, da Silva MdM, 2022 *A Tailored Benders Decomposition Approach for Last-mile Delivery with Autonomous Robots. European Journal of Operational Research* 299(2):510–525.

Archetti C, Bertazzi L, 2021 *Recent Challenges in Routing and Inventory Routing: E-commerce and Last-mile Delivery. Networks* 77(2):255–268.

Bakach I, Campbell AM, Ehmke JF, 2021 *A Two-tier Urban Delivery Network with Robot-Based Deliveries. Networks* 78(4):461–483.

Beasley J, 1983 *Route First–Cluster Second Methods for Vehicle Routing. Omega* 11(4):403–408.

Boccia M, Masone A, Sforza A, Sterle C, 2021 *A Column-and-Row Generation Approach for the Flying Sidekick Travelling Salesman Problem. Transportation Research Part C: Emerging Technologies* 124:102913.

Bouman P, Agatz N, Schmidt M, 2018 *Dynamic Programming Approaches for the Traveling Salesman Problem with Drone. Networks* 72(4):528–542.

Boysen N, Schwerdfeger S, Weidinger F, 2018 *Scheduling Last-mile Deliveries with Truck-based Autonomous Robots. European Journal of Operational Research* 271(3):1085–1099.

Chen C, Demir E, Huang Y, 2021 *An Adaptive Large Neighborhood Search Heuristic for the Vehicle Routing Problem with Time Windows and Delivery Robots. European Journal of Operational Research* 294:1164–1180.

Chen C, Demir E, Huang Y, Qiu R, 2021 *The Adoption of Self-Driving Delivery Robots in Last Mile Logistics. Transportation Research Part E* 146:102214.

de Freitas J, Penna P, 2020 *A Variable Neighborhood Search for Flying Sidekick Traveling Salesman Problem. International Transactions in Operational Research* 27(1):267–290.

Dell'Amico M, Montemanni R, Novellani S, 2021 *Algorithms based on Branch and Bound for the Flying Sidekick Traveling Salesman Problem. Omega* 104:102493.

Dell'Amico M, Montemanni R, Novellani S, 2022 *Exact Models for the Flying Sidekick Traveling Salesman Problem. International Transactions in Operational Research* 29(3):1360–1393.

Drexl M, 2012 *Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. Transportation Science* 46(3):297–316.

El-Adle A, Ghoniem A, Haouari M, 2021 *Parcel Delivery by Vehicle and Drone. Journal of the Operational Research Society* 72(2):398–416.

Es Yurek E, Ozmutlu H, 2018 *A Decomposition-based Iterative Optimization Algorithm for Traveling Salesman Problem with Drone. Transportation Research Part C: Emerging Technologies* 91:249–262.

Fischetti M, Salazar González JJ, Toth P, 1998 *Solving the Orienteering Problem through Branch-and-Cut. INFORMS Journal on Computing* 10(2):133–148.

Ha Q, Deville Y, Pham Q, Hà M, 2020 *A Hybrid Genetic Algorithm for the Traveling Salesman Problem with Drone. Journal of Heuristics* 26(2):219–247.

Holland JH, 1975 *Adaptation in Natural and Artificial Systems: An Introductory analysis with Applications to Biology, Control, and Artificial Intelligence* (University of Michigan Press).

Lacomme P, Prins C, Ramdane-Chérif W, 2001 *A Genetic Algorithm for the Capacitated Arc Routing Problem and its Extensions.* Boers E, ed., *Applications of Evolutionary Computing. EvoWorkshops 2001. Lecture Notes in Computer Science*, volume 2037, 473–483 (Springer, Berlin, Heidelberg).

Liu Z, Li X, Khojandi A, 2022 *The Flying Sidekick Traveling Salesman Problem with Stochastic Travel Time: A Reinforcement Learning Approach. Transportation Research Part E: Logistics and Transportation Review* 164:102816.

Macrina G, Di Puglia Pugliese L, Guerriero F, Laporte G, 2020 *Drone-aided Routing: A Literature Review. Transportation Research Part C* 120:102762.

Moscato P, Cotta C, 2010 *A Modern Introduction to Memetic Algorithms.* Gendreau M, Potvin JY, eds., *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, chapter 6 (Boston: Springer).

Murray CC, Chu AG, 2015 *The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-assisted Parcel Delivery. Transportation Research Part C: Emerging Technologies* 54:86–109.

Oliver I, Smith D, Holland J, 1987 *A Study of Permutation Crossover Operators on the Traveling Salesman Problem. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 224–230.

Ostermeier M, Heimfarth A, Hübner A, 2022 *Cost-optimal Truck-and-Robot Routing for Last-mile Delivery. Networks* 79(3):364–389.

Otto A, Agatz N, Campbell J, Golden B, Pesch E, 2018 *Optimization Approaches for Civil Applications of Unmanned Aerial Vehicles (UAVs) or Aerial Drones: A Survey. Networks* 72:411–458.

Poikonen S, Golden B, Wasil E, 2019 *A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone. INFORMS Journal on Computing* 31(2):335–346.

Prins C, 2004 *A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. Computers & Operations Research* 31(12):1985–2022.

Roberti R, Ruthmair M, 2021 *Exact Methods for the Traveling Salesman Problem with Drone. Transportation Science* 55(2):315–335.

Schermer D, Moeini M, Wendt O, 2020 *A Branch-and-Cut Approach and Alternative Formulations for the Traveling Salesman Problem with Drone. Networks* 76(2):164–186.

Scherr YO, Saavedra BAN, Hewitt M, Mattfeld DC, 2019 *Service Network Design with Mixed Autonomous Fleets. Transportation Research Part E: Logistics and Transportation Review* 124:40–55.

Simoni MD, Kutanoglu E, Claudel CG, 2020 *Optimization and Analysis of a Robot-assisted Last Mile Delivery System. Transportation Research Part E* 142:102049.

Soares R, Marques A, Amorim P, Parragh SN, 2023 *Synchronisation in vehicle routing: classification schema, modelling framework and literature review. European Journal of Operational Research* .

Srinivas S, Ramachandiran S, Rajendran S, 2022 *Autonomous Robot-driven Deliveries: A Review of Recent Developments and Future Directions. Transportation Research Part E: Logistics and Transportation Review* 165:102834.

Statista, 2021a *eCommerce Report 2021.* `https://www.statista.com/study/42335/ecommerce-report/`, accessed on November 21st, 2022.

Statista, 2021b *Leading Aspects that Global Shoppers Would Change about the Delivery of Products Purchased Online as of April 2021.* `https://www.statista.com/statistics/1274950/global-online-shoppers-wished-changes-on-delivery/`, accessed on November 21st, 2022.

Vásquez S, Angulo G, Klapp M, 2021 *An Exact Solution Method for the TSP with Drone Based on Decomposition. Computers and Operations Research* 127:105127.

Vidal T, Crainic TG, Gendreau M, Rei W, 2012 *A Hybrid Genetic Algorithm for Multi-depot and Periodic Vehicle Routing Problems. Operations Research* 60(3):611–624.

Wang Y, 2022a *JD.com's Record-breaking 2022 Singles' Day Grand Promotion Reflects Robust Consumption Vitality.* `https://jdcorporateblog.com/jd-coms-record-breaking-2022-singles-day-grand-promotion-reflects-robust-consumption-vitality/`, accessed on August 1st, 2023.

Wang Y, 2022b *The First Chinese Retailer Among World Top 10: JD.com.* `https://jdcorporateblog.com/the-first-chinese-retailer-among-world-top-10-jd-com/`, accessed on August 1st, 2023.

Yu S, Puchinger J, Sun S, 2020 *Two-echelon Urban Deliveries using Autonomous Vehicles. Transportation Research Part E: Logistics and Transportation Review* 141:102018.

Yu S, Puchinger J, Sun S, 2021 *Van-based Robot Hybrid Pickup and Delivery Routing Problem. European Journal of Operational Research* 298(3):894–914.