Branch-and-Cut for the Active-Passive Vehicle Routing Problem

Christian Tilk*,[b], Michael Forbes*,[a]

[a] *School of Mathematics and Physics, Faculty of Science, The University of Queensland, St Lucia, Brisbane, Australia*
[b] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

Abstract

This paper studies the active-passive vehicle-routing problem (APVRP). The APVRP covers a range of logistics applications where pickup-and-delivery requests necessitate a joint operation of active vehicles (e.g., trucks) and passive vehicles (e.g., loading devices such as containers). It supports a flexible coupling and decoupling of active and passive vehicles at customer locations in order to achieve a high utilization of both resources. This flexibility raises the need to synchronize the operations and the movements of active and passive vehicles in time and space. The contribution of the paper is twofold. First, we present a branch-and-cut algorithm for the exact solution of the APVRP that is based on Benders decomposition. Second, our approach can be generalized to deal with other vehicle-routing problems with timing aspects and synchronization constraints. Especially for the more complicated cases in which completion time or duration of routes is part of the objective, we show how stronger optimality cuts can be defined by identifying minimal responsible subset. Computational experiments show that the proposed algorithm outperforms the previous state-of-the-art for the APVRP and compute optimal solutions for more than 70 previously unsolved benchmark instances.

Key words: Routing, synchronization, branch-and-cut, Benders decomposition, truck and trailer

1. Introduction

Many applications in the area of transport logistics involve problems in which the execution of transport requests calls for a joint operation of *active vehicles* such as trucks, tractors, and *passive vehicles*, e.g., trailers, semitrailers, and containers. While active vehicles can move on their own from one location to another, passive vehicles require an active vehicle for being repositioned. In the literature on vehicle-routing problems (VRPs, see Irnich *et al.* (2014) for an overview), the distinction of active and passive vehicles is usually ignored, and operations are planned for active vehicles only (cf. the survey by Lahyani *et al.* (2015)). Active and passive vehicles are often paired to fixed units when modeling real-world problems. While this fixed active-passive assignment eases the solution of the VRP, it may hinder an effective utilization of the resources. If, for example, a manned truck (active vehicle) and an empty container (passive vehicle) are considered a unit in operations planning, the truck and its driver have to wait at a customer location while the container

*Corresponding author.
Email addresses: tilk@uni-mainz.de (Christian Tilk), m.forbes@uq.edu.au (Michael Forbes)

is being loaded. To overcome this issue and to support a more flexible use of such resources, Meisel and Kopfer (2014) introduced the *active-passive vehicle routing problem* (APVRP), in which an explicit distinction between active and passive vehicles is made. Given a set of pickup-and-delivery requests, the APVRP enables a flexible modeling of complex transport operations in which an active vehicle carries a passive vehicle to some pickup location, drops it off there, and leaves this location to transport other passive vehicles elsewhere. Later, when the passive vehicle has been loaded, the same or some other active vehicle returns to the customer, picks up the container, and carries it to its delivery location. This flexibility introduces multiple interdependencies between vehicles and raises the need to synchronize the operations and the movements of active and passive vehicles in time and space. According to the survey by Drexl (2012), such a combination of synchronization requirements is rarely addressed in the VRP literature.

To solve the APVRP, we use a Benders decomposition approach. Benders decomposition (Benders, 1962) is a partitioning method applicable to mixed-integer programs. It decomposes the original problem formulation into two simpler ones: an integer master problem and a linear subproblem. Recently, Rahmaniani *et al.* (2017) present a state-of-the-art survey on Benders decomposition approaches, emphasizing its use in combinatorial optimization.

The advantage of using Benders decomposition for VRP variants with timing aspects relies on the following observation: Formulations of VRP variants with timing aspects often utilizes arc-flow variables x_{ij} and node-time variables T_i that are bounded by the time windows. Mostly, these variable are coupled with the help of MTZ-constraints using a big-M Method. This usually results in a large mixed-integer model with a weak LP-relaxation. Moreover, the MIP solver carries the burden of all MTZ-constraints and bounded T variables at all branch-and-bound nodes, while they become relevant only when the corresponding x_{ij} variable takes a value of 1 (cf. Codato and Fischetti, 2006). Codato and Fischetti (2006) explained that you can get rid of the T variables using Benders decomposition but also that the resulting cuts are very weak and still depending on the Big-M values. To overcome this issue, they introduce *combinatorial Benders cuts* that are modelled only on the set of x_{ij} variables that violates timing aspects.

The contribution of the paper at-hand is twofold. First, we present an exact solution approach for the APVRP that is based on combinatorial Benders cuts and introduce a new compact formulation that constitutes the basis for our decomposition approach. Second, our approach can be generalized to deal with other VRP variants with timing aspect and synchronization constraints, especially for the more complicated cases in which completion time or duration of routes is part of the objective. For this purpose, we show how stronger optimality cuts can be derived from the solution of Benders subproblem by identifying minimal responsible subsets and how heuristic cut generation based on individual routes can be applied.

The remainder of this paper is structured as follows: Section 2 briefly reviews the literature on the APVRP and related problems. The formal problem description and the definition of appropriate network structures are provided in Section 3. Based on the network representation, a new compact formulation of the APVRP is presented in Section 4. Section 5 introduces our Benders decomposition approach and a simple solution algorithm. In Section 6, improvements of the simple algorithm are presented and the improved algorithm is computationally evaluated in Section 7. Section 8 summarizes the paper and discusses potential avenues for further research.

2. APVRP-related Literature

There are manifold real-world logistics applications of vehicle routing problems with active and passive vehicles that have attracted the attention of the research community. However, most of them are modelled with a fixed assignment of active and passive vehicle. Typical applications are found in routing problems where trucks pull trailers or swap bodies (see, e.g., Cheung *et al.*, 2008; Drexl, 2013). The most common problem in this area is the truck-and-trailer routing problem (TTRP, Chao, 2002), in which the assignment of active and passive vehicles is fixed but the passive vehicle can be decoupled for loading or because a street cannot be traversed by an active and a passive vehicle together. There are several applications for the TTRP, e.g., distribution of goods to grocery stores (Semet and Taillard, 1993), postal mail delivery (Bodin and Levy, 2000), the movement of empty and loaded containers for a logistics company (Tan *et al.*, 2006), and raw milk collection (Rothenbächer *et al.*, 2018). However, the features of the APVRP, which include simultaneous operations planning of active and passive vehicles together with the possibility to couple and decouple them flexibly on their routes, are not supported by the TTRP but could be exploited in all mentioned applications.

Literature considering a flexible assignment of active and passive vehicles is scarce. Meisel and Kopfer (2014) introduces the APVRP, provide mixed-integer programming (MIP) formulations, a branch-and-cut algorithm, an adaptive large neighborhood search(ALNS) metaheuristic, and benchmark instances. Recently, Tilk *et al.* (2018) presented a branch-and-price approach for the resolution of the APVRP and reported optimal solutions for instances with up to 76 tasks. There are some related problems that support a flexible coupling and decoupling of passive vehicles. Smilowitz (2006) models drayage operations that require the movement of loaded and empty equipment between rail yards, shippers, consignees and equipment yards. The problem is solved with a branch-and-price algorithm. Drexl (2014) studies the VRP with trailers and transshipments which supports, besides the flexible coupling, also load exchanges between all vehicles. Two branch-and-cut algorithms are presented, but only very small instances can be solved. Soares *et al.* (2019) introduce the full truck-load pickup and delivery problem with multiple vehicle synchronisation that is motivated by a real-life application in the biomass supply chain where it is necessary to simultaneously perform chipping and transportation operations at the forest roadside. They present a compact formulation of the problem that is used in a fix-and-optimise heuristic.

3. Problem Description and Network Definition

The APVRP can be formally described as follows: Let A be a set of classes of active vehicles and let K_a be the number of vehicles of class $a \in A$. All active vehicles are initially based at the same start depot o and end their routes at the same end depot d. Let P be a set of passive vehicles, each passive vehicle $p \in P$ is initially located at a specific start position o_p and must be brought to a specific end position d_p. Let R be a set of pickup-and-delivery requests, each request $r \in R$ consists of transporting a loading unit from a pickup location ℓ_r^+ to a delivery location ℓ_r^-. To fulfill a request r, an active vehicle must carry a passive vehicle to ℓ_r^+ for loading. Each passive vehicle can load only one request at a time, and each active vehicle can transport only one passive vehicle at a time. Hence, the loaded passive vehicle must then be transported directly to ℓ_r^- for unloading. Afterwards, the empty passive vehicle must be carried away from ℓ_r^-.

The set of locations is defined as $L := \{o, d\} \cup \{o_p, d_p : p \in P\} \cup \{\ell_r^+, \ell_r^- : r \in R\}$. The travel time between two location $i, j \in L$ is given by t_{ij} and the travel distance by c_{ij}. Moreover, s_r^+

indicates the time necessary to load an empty passive vehicle with request r and s_r^- indicate the time to unload r. Times for coupling and decoupling of passive vehicles to or from active vehicles are assumed to be zero (but could easily be incorporated into the model). Picking up a loaded request r at its pickup location can be finished no earlier than at time a_r. Waiting is allowed. Unloading a request r at its delivery location must be finished no later than b_r. The overall planning horizon is $[0, t^{max}]$. Requests that cannot be fulfilled imply a penalty.

Additionally, there exists compatibility relationships between the requests and the active and the passive vehicles. P^r denotes the set of passive vehicles that can be used to perform request r. Likewise, R^p denotes the set of requests that can be performed with passive vehicle p. P^a indicates the set of passive vehicles that can be coupled with active vehicles of class a, and A^p is the set of classes of active vehicles compatible with passive vehicle p. The objective is to minimize a weighted sum of the total distance traveled, the total completion time of the routes, and the number of unserved requests. The respective weights are $\alpha, \beta, \gamma \in \mathbb{R}_+$.

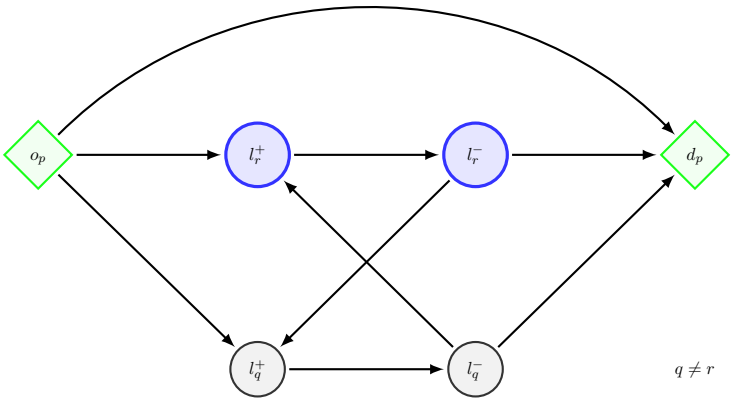

Figure 1: Passive Network - A passive vehicle moves between physical location

We model the APVRP as an optimization problem over two sets of graphs. The first set contains one graph for each passive vehicle modelling the movement of passive vehicles between physical locations. The route of a passive vehicle $p \in P$ can be described as an o_p-d_p-path in the graph $G^p = (V^p, E^p)$ with node set $V^p := \{o_p, d_p\} \cup \{l_r^+, l_r^- : r \in R^p\}$ and arc set $E^p := \{(o_p, d_p)\}$ $\cup\{(o_p, l_r^+), (l_r^+, l_r^-), (l_r^-, l_q^+), (l_r^-, d_p) : \ \forall\ r, q \in R^p \text{ with } r \neq q\}$ as depicted in Figure 1. The travel distances and travel times for an arc $(i, j) \in E^p$ are then given by the distance and time between the physical locations i and j, namely c_{ij} and t_{ij}. A passive vehicle fulfills a request if its route contains the visit of locations ℓ_r^+ and ℓ_r^-.

The second set of graphs contains one graph for each class of active vehicles $a \in A$ modelling the movement of active vehicles carrying a passive vehicle from one physical location to another. From the point of view of an active vehicle, the fulfillment of a request r comprises three transport *tasks* for the same passive vehicle p:

(i) carry an empty passive vehicle from its current location i to the pickup location of request r,
(ii) direct transport of a loaded passive vehicle from the pickup to the delivery location of r, and
(iii) carrying away the emptied passive vehicle from the delivery location to its next destination j.

Note that these three tasks for a request can be performed by up to three different active vehicles. In addition, there are two additional tasks for each passive vehicle p that need to be fulfilled, namely,

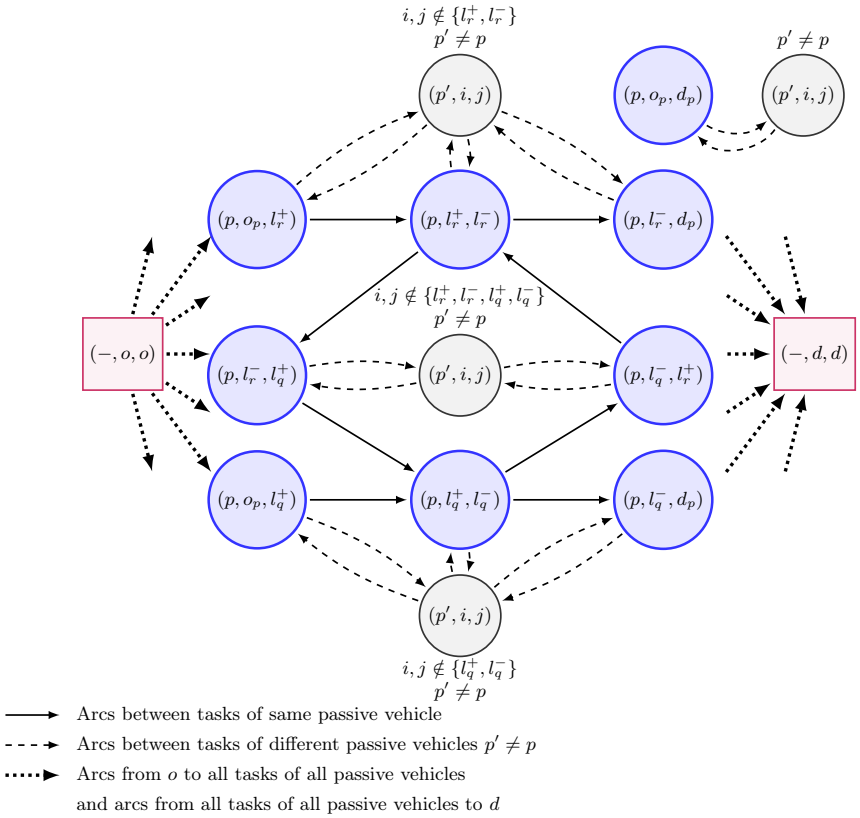

Figure 2: Active Network: An active vehicle moves between tasks to carry passive vehicles

carry the passive vehicle from o_p to the first location in its route and deliver it to d_p at the end of its route. Now, the route of an active vehicle $a \in A$ can be described as a sequence of tasks. Let $\Gamma := \cup_{a \in A} \Gamma^a$ be the set of all possible tasks, where $\Gamma^a := \{(p, i, j) : p \in P^a, (i, j) \in E^p\} \cup \{\tau_o, \tau_d\}$ is the set of all possible tasks for a class of active vehicles a with $\tau_o := (-, o, o)$ and $\tau_d := (-, d, d)$ modelling the start and end of routes. For the remainder of this paper, we define $\tau, \tau' \in \Gamma$ as $\tau := (p, i, j)$ and $\tau' := (p', i', j')$. Moreover, i is called the start location of task τ and j is called the end location of task τ. Hence, the journey of an active vehicle $a \in A$ can be defined as an τ_o-τ_d-path in the graph $G^a = (\Gamma^a, E^a)$ with node set Γ^a and arc set E^a as depicted in Figure 2. With this network definition, the active vehicles move in the line of tasks defined by the routes of passive vehicles, e.g., an active vehicle using an arc from (i, j, p) to (i', j', p') picks up the passive vehicle p from i, delivers it to j, then travels to i' to pick up the passive vehicle p' and delivers it to j'. (Note that $p = p'$ if and only if $j = i'$.) The travel distances and times of an arc $(\tau, \tau') \in E^a$ are given by the travel distance and time between the end location of task τ and the start location of task τ', namely $c_{ji'}$ and $t_{ji'}$, respectively. Moreover, each task has a time window as described in Table 1 according to the specifications given by (Meisel and Kopfer, 2014).

A feasible solution to the APVRP is a set of routes for the passive and a set of scheduled routes for the active vehicles that fulfills the following properties:

1. Each passive vehicle starts its journey at o_p and ends it at d_p.
2. For all $r \in R$ the arc (ℓ_r^+, ℓ_r^-) is used by at most one passive vehicle. Together with the structure of the passive network, this implies for all $r \in R$ that if arc (ℓ_r^+, ℓ_r^-) is used by a

Task τ	Earliest time a_τ	Latest time b_τ
(p, o_p, d_p)	t_{o,o_p}	$t^{max} - t_{d_p,p}$
(p, o_p, ℓ_r^+)	t_{o,o_p}	$b_r - s_r^+ - s_r^- - t_{\ell_r^+,\ell_r^-}$
(p, ℓ_r^-, d_p)	$a_r^p + s_r^- + t_{\ell_r^+,\ell_r^-}$	$t^{max} - t_{d_p,p}$
(p, ℓ_r^+, ℓ_r^-)	a_r^p	$b_r - s_r^-$
(p, ℓ_r^-, ℓ_q^+)	$a_r^p + s^- + t_{\ell_r^+,\ell_r^-}$	$b_q - s_q^+ - s_q^- - t_{\ell_q^+,\ell_q^-}$

Table 1: Time Windows of Tasks with $a_r^p := \max\{a_r, t_{oo_p} + t_{o_p\ell_r^+} + s^+\}$

specific passive vehicle p then (i) the passive vehicle p must also use exactly one arc from the set $\{(i, \ell_r^+) : i \in V^p\}$ and exactly one arc from the set $\{(\ell_r^-, j) : j \in V^p\}$ and (ii) all other passive vehicle $p' \neq p$ can not use any arc from the set $\{(i, \ell_r^+) : i \in V^{p'}\} \cup \{(\ell_r^-, j) : j \in V^{p'}\}$.

3. Each route of an active vehicle a starts at τ_o and ends at τ_d.
4. For each $a \in A$ there are at most K_a routes of active vehicles a.
5. All tasks that are present in the route of the passive vehicles are operated exactly once by one active vehicle.
6. The operation of all tasks in the route of an active vehicle are started in their time window.
7. For all $r \in R$, it holds $T_{i\ell_r^+} + t_{i\ell_r^+} + s_r^+ \leq T_{\ell_r^+\ell_r^-}$ and $T_{\ell_r^+\ell_r^-} + t_{\ell_r^+\ell_r^-} + s_r^- \leq T_{\ell_r^- j}$, where $T_{i\ell_r^+}$, $T_{\ell_r^+\ell_r^-}$, and $T_{\ell_r^- ,j}$ are the start times of the single tasks possible selected from the sets $\{(p, i, \ell_r^+) : p \in P^r,\ (i, \ell_r^+) \in E^p\}$, $\{(p, \ell_r^+, \ell_r^-) : p \in P^r,\ (\ell_r^+, \ell_r^-) \in E^p\}$, and $\{(p, \ell_r^-, j) : p \in P^r,\ (\ell_r^-, j) \in E^p\}$ (see point 2 above). These inequalities ensure the temporal synchronization of tasks within and between the routes of active vehicles.

4. Compact Formulation

In order to employ Benders decomposition, we introduce a new compact formulation for the APVRP based on the networks presented in the previous section. The formulation uses the following types of variables:

- Binary variables U_r indicating whether or not request $r \in R$ remains unfulfilled,
- binary variables X_{ij}^p measuring the flow of passive vehicle $p \in P$ along arc $(i, j) \in E^p$,
- binary variables $Y_{\tau\tau'}^a$ measuring the flow of active vehicles of class $a \in A$ along arc $(\tau, \tau') \in E^a$,
- and continuous variables T_τ indicating the start time of task τ.

The formulation reads as follows:

$$z_{CM} = \min \quad \alpha\Big(\sum_{p\in P}\sum_{(i,j)\in E^p} c_{ij}X_{ij}^p + \sum_{a\in A}\sum_{(\tau,\tau')\in E^a} c_{ji'}Y_{\tau\tau'}^a\Big) + \beta\sum_{a\in A}\sum_{\tau\in\Gamma^a}(T_\tau + t_{ij} + t_{jd})Y_{\tau\tau_d}^a + \gamma\sum_{r\in R}U_r \tag{1a}$$

$$\text{s.t.} \quad U_r + \sum_{p\in P^r} X_{l_r^+ l_r^-}^p = 1 \qquad r \in R \tag{1b}$$

$$\sum_{r\in R^p} X_{o_p l_r^+}^p + X_{o_p d_p}^p = 1 \qquad p \in P \tag{1c}$$

$$\sum_{(i,j)\in E^p} X_{ij}^p = \sum_{(h,i)\in E^p} X_{hi}^p \qquad p \in P \quad i \in L \setminus \{o_p, d_p\} \tag{1d}$$

$$\sum_{\tau\in\Gamma^a} Y_{\tau_o\tau}^a \leq K_a \qquad a \in A \tag{1e}$$

$$\sum_{\tau' \in \Gamma^a} Y^a_{\tau\tau'} = \sum_{\tau' \in \Gamma^a} Y^a_{\tau'\tau} \qquad a \in A \quad \tau \in \Gamma^a \setminus \{\tau_o, \tau_d\} \tag{1f}$$

$$\sum_{a \in A} \sum_{\tau' \in \Gamma^a} Y^a_{\tau'\tau} = X^p_{ij} \qquad p \in P \quad (i,j) \in E^p \tag{1g}$$

$$T_\tau + t_{ij} + t_{ji'} \leq T_{\tau'} + t^{max}(1 - \sum_{a \in A^p \cap A^{p'}} Y^a_{\tau\tau'}) \qquad (\tau, \tau') \in \bigcup_{a \in A} E^a \quad \tau' \neq \tau_d \tag{1h}$$

$$T_{(p,i,\ell^+_r)} + t_{i,\ell^+_r} + s^+_r \leq T_{(p,\ell^+_r,\ell^-_r)} + t^{max}(1 - X^p_{i\ell^+_r}) \qquad r \in R \quad p \in P^r \quad (i, \ell^+_r) \in E^p \tag{1i}$$

$$T_{(p,\ell^+_r,\ell^-_r)} + t_{\ell^+_r,\ell^-_r} + s^-_r \leq T_{(p,\ell^-_r,j)} + t^{max}(1 - X^p_{\ell^-_r j}) \qquad r \in R \quad p \in P^r \quad (\ell^-_r, j) \in E^p \tag{1j}$$

$$e_\tau X^p_{ij} \leq T_\tau \leq (l_\tau - t_{ij}) X^p_{ij} \qquad \tau \in \bigcup_{a \in A} \Gamma^a \tag{1k}$$

$$X^p_{ij} \in \{0,1\} \qquad p \in P \quad (i,j) \in E^p \tag{1l}$$

$$Y^a_{\tau\tau'} \in \{0,1\} \qquad a \in A \quad (\tau, \tau') \in E^a \tag{1m}$$

$$U_r \in \{0,1\} \qquad r \in R \tag{1n}$$

(1a) is the objective function minimizing the weighted sum of the travel costs and completion times of the routes as well as the penalties paid for unfulfilled requests. Constraints (1b) ensures that each request is either performed exactly once or the penalty has to be paid for leaving the request unfulfilled. Constraints (1c) guarantees that each passive vehicle is used. It ensures that the journey of each passive vehicle either starts with a request or that it is directly carried to its destination location. Constraints (1d) are flow conservation constraints for the journey of passive vehicles. The number of active vehicles of each class is limited by (1e). Flow conservation for each class of active vehicles is ensured by constraints (1f). Constraints (1g) couples the flow of the active and passive vehicles. It ensures that if a passive vehicle serves task $\tau = (p, i, j)$, an active vehicle must carry it. Constraints (1h) ensure time feasibility regarding the sequence of tasks in the routes of all active vehicles. The time synchronization of the passive vehicles' flow regarding the order of fulfilling a request is ensured by Constraints (1i) and (1j). The variable domains are given by (1k) – (1n). Note that the objective is non-linear since a product of variables is used to compute the completion time. On the one hand, linearizing is possible, e.g., by considering each active vehicle individually. On the other hand, it is not necessary for our Benders decomposition approach since the non-linear part vanishes in the Benders subproblem (see Section 5).

5. Benders Decomposition

To solve the APVRP with Benders decomposition, we use combinatorial Benders cuts (see, Codato and Fischetti (2006)). Benders decomposition is a partitioning method applicable to mixed-integer programs (Benders, 1962). It separates the original problem into two simpler ones: an integer master problem and a linear subproblem. A simple algorithm using combinatorial Benders cuts can be constructed as follows: The algorithm starts with solving the master problem via branch-and-bound. During the solution process, each integer solution found in the branch-and-bound tree is sent to the subproblem for examination. The subproblem decides first, if the solution is feasible and second, if the objective value of the master problem takes the appropriate value. If the solution is infeasible, a *feasibility cut* is generated and passed to the master to cut the solution off. Similarly, if the objective value is incorrect an *optimality cut* is generated and passed to the master to cut the solution off. Otherwise, the solution is identified as feasible and branch-and-bound continues. The interaction between master and subproblem is depicted in Figure 3.

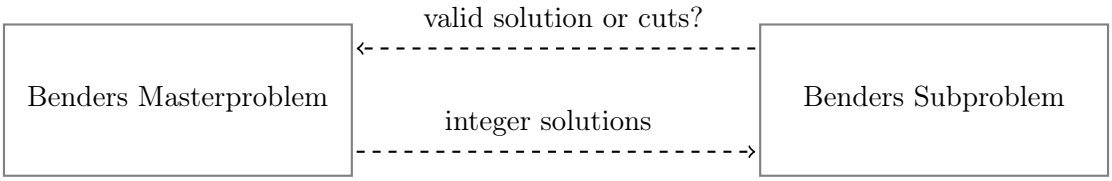

Figure 3: Interaction between Benders master and subproblem

Formulation 1 is decomposed such that the routing part is handled in the master problem and the timing aspects are managed in the subproblem. Since the end times of the routes of active vehicles are important for computing the objective value, we introduce new variables θ_τ^a for each class of active vehicles a and each task τ modelling the arrival time at the end depot d for an active vehicle of class a if this active vehicle performs task τ directly before going to the end depot, i.e., $Y^a_{\tau\tau_d} = 1$. Note that there must be exactly one variable $Y^a_{\tau\tau_d}$ with value 1 for each active vehicle used since each task τ can be performed at most once. Hence, the master problem reads as follows:

$$z_{BM} = \min \quad \alpha\Big(\sum_{p\in P}\sum_{(i,j)\in E^p} c_{ij}X^p_{ij} + \sum_{a\in A}\sum_{(\tau,\tau')\in E^a} c_{ji'}Y^a_{\tau\tau'}\Big) + \beta\sum_{a\in A}\sum_{\tau\in\Gamma^a}\theta^a_\tau + \gamma\sum_{r\in R}U_r \tag{2a}$$

s.t.(1b)–(1g) and (1l)–(1n)

(2a) is the objective function formulated in the new variables θ_τ^a. Note that this reformulation implies that time infeasible or non-elementary routes of passive and active vehicles become possible. Moreover, the timing component in the objective function is currently zero. To tackle these two aspects, we introduce our Benders subproblem and show how optimality and feasibility cuts can be generated.

Note first that if Formulation 2 turns out to be infeasible for a specific instance then also Formulation 1 is infeasible for that instance. Otherwise, let $(\bar{S}, \bar{\theta})$ with $\bar{S} = (\bar{X}, \bar{Y}, \bar{U})$ be the optimal solution of Formulation 2 with solution value $\bar{z}_{BM}$. Now, we have to check if there exists an assignment $\bar{T}$ of the T variables such that $(\bar{S}, \bar{T})$ satisfy Constraints (1h)-(1k) and $\bar{z}_{BM} = z^{CM}(\bar{S}, \bar{T})$. Since Constraints (1k) enforce that $T_\tau = 0$ for all $\tau \in \Gamma$ with $\bar{X}^p_{ij} = 0$, the Benders subproblem decomposes to:

$$z_{BS} = \min \quad \sum_{\tau: Y_{\tau\tau_d}\in\bar{Y}} T_\tau + t_{ij} + t_{jd} \tag{3a}$$

$$T_\tau + t_{ij} + t_{ji'} \le T_{\tau'} \qquad \sum_{a\in A^p\cap A^{p'}} \bar{Y}^a_{\tau\tau'} = 1 \quad \tau' \ne \tau_d \tag{3b}$$

$$T_{(p,i,\ell_r^+)} + t_{i\ell_r^+} + s_r^+ \le T_{(p,\ell_r^+,\ell_r^-)} \qquad \bar{X}^p_{i,\ell_r^+} = 1 \tag{3c}$$

$$T_{(p,\ell_r^+,\ell_r^-)} + t_{\ell_r^+\ell_r^-} + s_r^- \le T_{(p,\ell_r^-,j)} \qquad \bar{X}^p_{\ell_r^-,j} = 1 \tag{3d}$$

$$a'_\tau \le T_\tau \le b'_\tau - t_{ij} \qquad \bar{X}^p_{ij} = 1 \tag{3e}$$

The objective (3a) minimizes the sum of the completion time of all routes. Constraints (3b) ensure time feasibility regarding the sequence of tasks in the routes of all active vehicles. Time synchronization of the passive vehicles' flow is ensured by constraints (3c) and (3d). The variable domains are given by (3e).

On the one hand, if the linear system 3 is infeasible then there does not exist any combination $\bar{T}$ of the T variables such that $(\bar{S}, \bar{T})$ is a feasible solution to the APVRP. Hence, we need to generate a feasibility cut to exclude solution $\bar{S}$ from the feasible region of Formulation 2. Let $E^{\bar{X}} := \{(p,i,j) : \bar{X}^p_{ij} = 1\}$ be the set of all executed tasks in $\bar{S}$ and let $E^{\bar{Y}} := \{(\tau,\tau') : \sum_{a \in A^p \cap A^{p'}} \bar{Y}^a_{\tau\tau'} = 1\}$ be the set of all tasks that are consecutively served by an active vehicle. The general form of Benders feasibility cut is given by:

$$\sum_{(p,i,j) \in E^{\bar{X}}} X^p_{ij} + \sum_{(\tau,\tau') \in E^{\bar{Y}}} \sum_{a \in A^p \cap A^{p'}} Y^a_{\tau\tau'} \leq |E^{\bar{X}}| + |E^{\bar{Y}}| - 1 \tag{4}$$

However, this cut can be strengthened since the infeasibility may rely only on some of the X^p_{ij} and $Y^a_{\tau\tau'}$ variables that occur in constraint (4). As explained in Codato and Fischetti (2006), a stronger cut can be derived by computing a *minimal (or irreducible) infeasible subsystem (MIS)* $(E^{X^*} \cup E^{Y^*}) \subset (E^{\bar{X}} \cup E^{\bar{Y}})$ and defining the cut on that subset. The MIS can be any inclusion-minimal set of row indices of Formulation 3 such that the resulting linear subsystem is infeasible.

On the other hand, if the linear system 3 has a feasible solution $\bar{T}$ with solution value $\bar{z}_{BS}$, then $(\bar{S}, \bar{T})$ is a feasible solution to Formulation 1 with solution value $z_{UB} := \bar{z}_{BM} + \beta(\bar{z}_{BS} - \sum_{a \in A} \sum_{\tau \in \Gamma^a} \bar{\theta}^a_\tau)$. If $\bar{z}_{BS} = \sum_{a \in A} \sum_{\tau \in \Gamma^a} \bar{\theta}^a_\tau$, the solution $\bar{S}$ has the correct objective value and no cut is passed to the Benders master problem. Otherwise, z_{UB} is an valid upper and we need to generate an optimality cut forcing the solution $\bar{S}$ to take the value z_{UB} in the Benders master problem. The general form of Benders optimality cut is given by:

$$\sum_{a \in A} \sum_{\tau : (\tau,\tau_d) \in E^{\bar{Y}}} \theta^a_\tau \geq \bar{z}_{BS}(1 - |\bar{Y}| + \sum_{(\tau,\tau') \in E^{\bar{Y}}} \sum_{a \in A^p \cap A^{p'}} Y^a_{\tau\tau'}). \tag{5}$$

Similarly to the MIS for feasibility cuts, we can strengthen optimality cuts by defining a *minimal responsible subset (MRS)* $E^{Y^*} \subset E^{\bar{Y}}$ and define the cut on that subset. The idea to use MIS and MRS is based on the seminal work of Hooker (2000) who derives Benders cuts from minimal sets of inconsistencies which are heuristically computed using greedy algorithms. However, a MRS can also be computed exactly. Any optimal solution of Formulation 3 defines a MRS which contains only those pairs of tasks that corresponds to a non-zero dual variable plus the set $\{(\tau,\tau_d) : (\tau,\tau_d) \in E^{\bar{Y}}\}$. Note that the approach of using a MRS will work in general for all combinatorial Benders decomposition algorithms. The proof is simple: If we remove the constraints in the subproblem corresponding to dual variables which are zero, then the original primal and dual solutions are still primal and dual feasible for the new problem. Moreover, the solution value for the reduced primal and dual problems remains the same and thus the solutions are still optimal. Hence, the cut is valid with just the corresponding reduced set of binary variables.

Note that, instead of adding only one inequality per solution of the Benders subproblem, we can compute several in the following manner: If the Benders subproblem is feasible, we can generate at most one optimality cut. Otherwise, we generate a MIS and the corresponding feasibility cut, remove all constraints corresponding to this MIS from the Benders subproblem and solve it again. This procedure is iteratively repeated until the Benders subproblem is feasible, then we may generate an optimality cut as a last step.

6. Improved Algorithm

In this section, we introduce two techniques that can be used to improve the simple algorithm introduced in the previous section. First, we describe several inequalities based on the feasibility

and ending time of individual routes of active and passive vehicle in Section 6.1. Second, we show how the active vehicle index can be eliminated from tasks to significantly reduce the number of variables in Section 6.2. Last, we summarize the overall solution algorithm in Section 6.3.

6.1. Valid Inequalities

To speed-up the solution procedure, we use several heuristic procedures to check for individual routes of passive and active vehicles of a solution $\bar{S}$ if they are infeasible or should enforce larger θ-values. Only if all heuristics fail to find a feasibility cut, we solve Formulation 3. Note that testing the route of passive vehicle p for feasibility corresponds to checking constraints (1i), (1j), and (1k) for p. Similarly, testing the route of an active vehicle a_1 for feasibility corresponds to checking constraints (1h) and (1k) for that specific vehicle a_1 (after decomposing the variables for classes of active vehicles in routes for individual vehicles). On the one hand, there are two types of infeasible routes: A route may be a cycle (not starting at the depot) or a route may be infeasible due to the time window constraints of the involved requests. Both kinds of infeasibility can be cut off by inequalities of type (4). On the other hand, if a single route of an active vehicle is feasible, we can check if its individual ending time (without possibly required synchronization between routes) is smaller or equal to the value of the corresponding θ-variable and generate an optimality cut if it is larger. Next, we give a more detailed description of the different kinds of single route inequalities that we use and how they are separated.

Cycle Inequalities. Checking a solution $(\bar{S}, \bar{\theta})$ of Formulation 2 for cycles on the X variables or Y variables can be done by inspection. Cycle inequalities are strengthened using the tournament constraints (Ascheuer *et al.*, 2000). Let $C := (l_1, l_2, \ldots, l_n, l_1)$ be the sequence of locations forming a cycle on the X variables, the corresponding feasibility cut is given by:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{\substack{p: \\ (l_i, l_j) \in E^p}} X^p_{l_i l_j} \leq n - 1 \tag{6}$$

Likewise, let $C := (\tau_1, \tau_2, \ldots, \tau_n, \tau_1)$ be the sequence of tasks forming a cycle on the Y variables, the corresponding cut is given by:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{\substack{a: \\ (\tau_i, \tau_j) \in E^a}} Y^a_{\tau_i \tau_j} \leq n - 1 \tag{7}$$

Infeasible Chain Inequalities. To check if an active or passive route respects the time windows of the involved requests, we compute the earliest arrival time at each vertex in the route using resource extension functions (REFs, see Irnich, 2008) and check if time windows are violated. If the route is infeasible, we can try to shorten it by removing arcs at the start and the end of the route as long as the infeasibility remains. More precisely, we use forward labeling to identify the first infeasible position i and cut all nodes after that position off. Then, we start backward labeling from node i to identify the first infeasible position backwards and cut all nodes ahead of this position off. The remaining chain is the smallest with respect to that infeasibility. Certainly, a route may have several chains that causes infeasibility, another chain, if one exists, can be found by inverting the order of forward and backward labeling.

For the identified chains, we construct tournament simple fork inequalities (Ropke *et al.*, 2007) by adding all other arcs to the chain that can replace the first or last arc while infeasibility is kept.

Let $P := (l_1, l_2, \ldots, l_n, l_{n+1})$ be the sequence of locations forming an infeasible chain on the X variables and let $S := \{j \in L : (j, l_2, \ldots, l_n, l_{n+1}) \text{ is an infeasible chain}\}$ be the set of locations that can replace the first location in the chain while infeasibility is kept. The tournament simple infork inequality for infeasible chains of passive vehicles is given by:

$$\sum_{i=2}^{n} \sum_{j=i+1}^{n+1} \sum_{\substack{p: \\ (l_i, l_j) \in E^p}} X^p_{l_i l_j} + \sum_{k \in S} \sum_{\substack{p: \\ (k, l_2) \in E^p}} X^p_{k l_2} \leq n - 1 \tag{8}$$

Analogous, we can define tournament simple outfork inequalities by replacing the last location in the chain while infeasibility is kept. Let $T := \{j \in L : (l_1, l_2, \ldots, l_n, j) \text{ is an infeasible chain}\}$ be the set of locations that can replace the last location in the chain while infeasibility is kept. The tournament simple outfork inequality for infeasible chains of passive vehicles is given by:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{\substack{p: \\ (l_i, l_j) \in E^p}} X^p_{l_i l_j} + \sum_{k \in T} \sum_{\substack{p: \\ (l_n, k) \in E^p}} X^p_{l_n k} \leq n - 1 \tag{9}$$

Additionally, if there does not exist any permutation Π such that the chain $(l_1, \Pi(l_2), \ldots, \Pi(l_n), l_{n+1})$ is feasible, we can use a lifted version of the tournament inequalities (Ascheuer *et al.*, 2000):

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n+1} \sum_{\substack{p: \\ (l_i, l_j) \in E^p}} X^p_{l_i l_j} + \sum_{j=2}^{n-1} \sum_{i=j+1}^{n} \sum_{\substack{p: \\ (l_i, l_j) \in E^p}} X^p_{l_i l_j} \leq n - 1 \tag{10}$$

Moreover, we can further lift these inequalities if there exists no permutation including l_1 or l_{n+1} or both such that the permuted chain is feasible.

In the same manner, we can check if the individual routes of active vehicles respect the time windows of the involved tasks. The tournament simple infork inequality for a sequence of tasks $P := (\tau_1, \tau_2, \ldots, \tau_n, \tau_{n+1})$ forming an infeasible chain on the Y variables and the set $S := \{j \in L : (j, \tau_2, \ldots, \tau_n, \tau_{n+1}) \text{ is an infeasible chain}\}$ is given by:

$$\sum_{i=2}^{n} \sum_{j=i+1}^{n+1} \sum_{\substack{a: \\ (\tau_i, \tau_j) \in E^a}} Y^a_{\tau_i \tau_j} + \sum_{k \in S} \sum_{\substack{a: \\ (k, \tau_2) \in E^a}} Y^a_{k \tau_2} \leq n - 1 \tag{11}$$

The tournament simple outfork inequalities for active vehicles can be constructed in the same manner. Similarly, the lifted tournament inequalities, if there does not exist any permutation Π such that the chain $(\tau_1, \Pi(\tau_2), \ldots, \Pi(\tau_n), \tau_{n+1})$ is feasible, is given by:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n+1} \sum_{\substack{p: \\ (\tau_i, \tau_j) \in E^a}} Y^a_{\tau_i \tau_j} + \sum_{j=2}^{n-1} \sum_{i=j+1}^{n} \sum_{\substack{a: \\ (\tau_i, \tau_j) \in E^a}} Y^a_{\tau_i \tau_j} \leq n - 1 \tag{12}$$

Endtime Inequalities. For each feasible route of an active vehicle, we can check if the corresponding θ variable is greater than or equal to the completion time of that route and if not, we generate a feasibility cut to force the θ variables to take the appropriate value. Let $R := (\tau_o, \tau_1, \ldots, \tau_n, \tau_d)$ be a feasible route of an active vehicle and let $T^R_{(i,n)}$ be the completion time of route $(\tau_o, \tau_i, \tau_{i+1}, \ldots, \tau_n, \tau_d)$. $T^R_{(i,n)}$ can be computed using a labeling algorithm. The endtime inequality for route R is given by:

$$\theta^a_{\tau_n} \geq T^R_{(1,n)} Y^a_{\tau_n, \tau_d} + \sum_{i=1}^{n-1} (Y^a_{\tau_i, \tau_{i+1}} - 1)(T^R_{(1,n)} - T^R_{(i+1,n)}) \tag{13}$$

This inequality forces $\theta^a_{\tau_n}$ to take a value at least as large as the completion time of Route R if all Y variables in route R are used. Moreover, for each route that results from removing tasks from R (except τ_n), $\theta^a_{\tau_n}$ is forced to take a value at least as large as a lower bound for the completion time of this route.

6.2. Passive Vehicle Index Reduction

The large number of Y variables in Benders master problem slows down the solution of the Benders master problem. We can reduce the number of Y variables and the number of constraints by removing the index p of the passive vehicle from all tasks $\tau \in \Gamma$ and defining constraints that ensure that only compatible active vehicle can fulfil a task. More precisely, we define $\Gamma^a := \{(i,j) : \exists p \in P^a \text{ with } (i,j) \in E^p\} \cup \{\tau_o, \tau_d\}$ and replace Constraints (1g) by:

$$\sum_{a \in A} \sum_{\tau' \in \Gamma^a} Y^a_{\tau'\tau} = \sum_{p \in P} \sum_{(i,j) \in E^P} X^p_{ij} \qquad (i,j) \in \bigcup_{p \in P} E^p \tag{14a}$$

$$\sum_{a \in A^p} \sum_{\tau' \in \Gamma^a} Y^a_{\tau'\tau} \geq X^p_{ij} \qquad p \in P \quad (i,j) \in E^p \tag{14b}$$

Constraints (14a) enforce that an active vehicle fulfils the transport task if it is assigned to any passive vehicle. With the help of Constraints (14b), it is ensured that only compatible active vehicles can fulfil the transport task.

6.3. Algorithm

This section gives a more detailed description of our solution algorithm. We embedded the combinatorial Benders cuts in a branch-and-cut algorithm in Python 3.6 using Gurobi 8.5 to solve the Benders master and subproblem. Combinatorial Benders cuts are embedded in the algorithm using the "MipNode-Callback" function of Gurobi to separate them at each integer node of the branch-and-bound tree. The corresponding cut callback is given in pseudo-code in Algorithm 1. Note first that the algorithm has two phases: In Phase 1, the Benders master problem is solved without the θ-variables, i.e., disregarding the time-component of the objective function. All optimality cuts that could be generated in the first phase of the algorithm are instead stored in a list. Furthermore, the best solution found with respect to the full objective is also recorded and used as a starting solution in phase 2. After an optimal solution regarding the objective of phase 1 is found, phase 2 is initialized by adding the θ-variables and the problem is solved again regarding the complete objective function. Moreover, to obtain a good starting lower bound for the θ-variables, we generate optimality cuts for all active routes of length three and four and add them together with the optimality cuts stored in phase 1 as lazy constraints. The reason for using two phases is that the first phase is usually solved very quickly and gives an good upper bound for phase 2. Fischetti *et al.* (2016) pointed out that it is very important to start with a good upper bound and with a rich family of Benders cuts, to intensify root-node variable fixing and generation of internal cuts.

We comment on Algorithm 1 in more detail: The input data of the cut callback algorithm are the Benders master problem (BM), the (integer) solution of the current node $(\bar{S}, \bar{\theta})$ and the corresponding solution value z_{BM}, the current phase (*phase*) and the best known upper bound for the second phase (UB_2). The return value is a boolean indicating if the solution $(\bar{S}, \bar{\theta})$ is valid or

Algorithm 1: Cut Callback Routine(BM,$\bar{S}, \bar{\theta}$, z_{BM},*phase*,UB_2)

```
1  if HeuristicCutSeparation(BM,S̄,θ̄,phase) > 0 then
2      return false
3  else
4      Construct and Solve BS(S̄,θ̄) → (feasible,MS,z_BS)
5      if !feasible then
6          Add Feasibility cut(MS) to BM
7          return false
8      else
9          if (UB_2 > (z_BM − Σ_{a∈A} Σ_{τ∈Γ^a} θ̄^a_τ) + z_BS then
10             Set UB_2 := (z_BM − Σ_{a∈A} Σ_{τ∈Γ^a} θ̄^a_τ) + z_BS
11         if phase==1 then
12             return true
13         else
14             if z_BS == Σ_{a∈A} Σ_{τ∈Γ^a} θ̄^a_τ then
15                 return true
16             else
17                 Add Optimality cut(MS) to BM
18                 return false
```

not. In Step 1, the algorithm first checks if violated inequalities for individual routes of passive and active vehicles in $(\bar{S}, \bar{\theta})$ can be generated (see Section 6.1). The corresponding separating procedure is given in pseudo-code in Algorithm 2 and will be discussed later on. If no violated feasibility cut for individual routes of passive and active vehicles can be found, the Benders subproblem (BS) for solution $(\bar{S}, \bar{\theta})$ is constructed and solved. Note that the algorithm proceeds to solve BS even if the heuristic has found violated optimality cuts. Thus, the algorithm can compute an valid upper bound for phase 2 for every feasible solution $(\bar{S}, \bar{\theta})$. The output of BS is a boolean indicating if BS is feasible (*feasible*), a minimal subset (*MS*) that is either a MIS or a MRS (see Section 5), and the solution value (z_{BS}) which is infinity if BS is infeasible. The MIS can be obtained using the "IISCONSTR"-function of Gurobi, the MRS can be obtained as explained in Section 5. If the solution procedure returns infeasible, a feasibility cut (4) is added to BM and the algorithm returns false. Otherwise, the algorithm computes a valid upper bound for phase 2 and checks if it improves the old one. Then, it is examined if solution $(\bar{S}, \bar{\theta})$ takes the correct objective value: In phase 1, the algorithm returns directly true since the time component of the objective is missing. In phase 2, the algorithm checks if the value of the θ-variables equals the objective value of BS and if so, $(\bar{S}, \bar{\theta})$ is a feasible solution with correct objective value. Otherwise, an optimality cut (5) is added to BM and the algorithm returns false.

Next, we comment on Algorithm 2 in more detail. The inputs of the algorithm are the Benders master problem (BM), the (integer) solution of the current node $(\bar{S}, \bar{\theta})$, and the current phase (*phase*). The return value gives the number of added feasibility cuts (*numAdded*). First, the algorithm decomposes the X-variables into individual routes and cycles. For each time-infeasible passive route, an infeasible chain is constructed and two inequalities (infork and outfork) of type (8) and (9) are generated and added to BM. If possible, a lifted tournament inequality of type (10) is also added to BM. For each passive cycle, an inequality of type (6) is generated and added to BM. If any violated inequality was detected, the algorithm terminates since it is not necessary to examine

Algorithm 2: HeuristicCutSeparation(BM,$(\bar{S},\bar{\theta})$,*phase*)

Init: *numAdded*=0
Decompose $\bar{X}$ into single routes $R_1^p \dots R_k^p$ and cycles $C_1^p \dots C_l^p$
for $i = 1$ *to* k **do**
    if R_i^p *is time-feasible* **then**
        continue
    else
        Add chain inequality(R_i^p) to BM
        numAdded++
for $i = 1$ *to* l **do**
    Add cycle inequality(C_i^p) to BM
    numAdded++
if *numAdded>0* **then**
    return *numAdded*
Decompose $\bar{Y}$ into single routes $R_1^a \dots R_n^a$ and cycles $C_1^a \dots C_m^a$
for $i = 1$ *to* n **do**
    if R_i^a *is time-feasible* **then**
        if $\theta(R_i^a) < End(R_i^a)$ **then**
            if *phase==1* **then**
                Store endtime inequality(R_i^a)
            else
                Add endtime inequality(R_i^a) to BM
    else
        Add chain inequality(R_i^a) to BM
        numAdded++
for $i = 1$ *to* m **do**
    numAdded++
    if C_i^a *is time-feasible* **then**
        Add cycle inequality(C_i^a) to BM
    else
        Add chain inequality(C_i^a) to BM
return *numAdded*

the active routes because they will change anyway if some chains of passive variables are excluded from the feasible region. Otherwise, the algorithm proceeds to decompose the Y-variables into individual routes and cycle. Afterwards, each individual active route is checked for time-feasibility: For time-feasible active routes, the algorithm checks if the corresponding θ-variable takes at least the value of the completion time of that route. If not, an optimality cut of type (13) is generated and either stored or directly added to BM depending on the phase the algorithm is currently in. Note that the algorithm does not increase the parameter *numAdded* here since only feasibility cuts are counted. Thus, the correct objective value of solution $(\bar{S},\bar{\theta})$ can be computed to potential improve the overall upper bound UB_2 in Algorithm 1. If the route turns out to be time-infeasible, an infeasible chain is constructed and two inequalities (infork and outfork) of type (11) are generated and added to BM. If possible, we also add a lifted tournament inequality of type (12). Finally, the algorithm deals with active cycles. For cycles containing an time-infeasible chain, two infeasible chain inequalities (infork and outfork) of type (11) and, if possible, an inequality of type (12) are

generated and added to BM. Cycle inequalities of type (7) are only generated for time-feasible cycles. The reason for generating infeasible chain inequalities rather than cycle inequalities is that they are stronger in the sense that more routes are affected by them since chains are usually smaller than cycles regarding the number of involved variables.

7. Computational Results

The results reported in this section were obtained using a high-performance computing system running Linux. Each job was assigned a maximum of 4 cores running at 2.4GHz each, and 32GB of RAM. The algorithm was implemented in Python 3.6 and Gurobi 8.5 was used to solve Benders master and subproblem. For all experiments, we set a CPU time limit of two hours for each of the two phases of our branch-and-cut algorithm.

We tested our algorithm on the 30 Benchmark instances of Meisel and Kopfer (2014) with 10 requests and the following characteristics: The instances have 2 identical active vehicles, 4 passive vehicles, and a planning horizon of 2,500 time units. Locations are randomly placed in a 100×100 area, distances and travel times are set to the Euclidean distance, time windows are of width 1,000 with random start times, and service times for loading and unloading range between 50 and 100 time units. Each passive vehicle is compatible with two active vehicles; each request is compatible with three passive vehicles. Because their huge planning horizon is difficult to handle with an exact approach, Tilk *et al.* (2018) introduce a set of 160 new instances that feature a more coarse time discretisation with a 1000 unit planning horizon. The set can be divided in the classes A and B. Class A instances have ten requests, two identical active vehicles, and four passive vehicles while class B instances have 20 requests, two classes of active vehicle with 2 identical vehicles each, and eight passive vehicles. In both classes, service times vary between 25 and 50. Moreover, both classes can be divided in 4 groups that feature time window widths of 25,50,100, and 200. All other characteristics of the new instances are the same as in the one of Meisel and Kopfer (2014).

Table 2: Comparison with the branch-and-price approach by Tilk *et al.* (2018)

instance class	Our Approach						Branch-and-price					
	No. solved	Time [s]			Gap [%]		No. solved	Time [s]			Gap [%]	
		Min	Avg	Max	Avg	Max		Min	Avg	Max	Avg	Max
A25	20	1	4	7	0.0	0.0	20	36	191	931	0.0	0.0
A50	20	1	4	8	0.0	0.0	20	43	362	2485	0.0	0.0
A100	20	1	4	7	0.0	0.0	19	38	1076	7200	0.1	1.1
A200	20	2	16	72	0.0	0.0	14	109	3466	7200	0.4	2.7
B25	20	70	357	1378	0.0	0.0	17	131	3594	7200	0.2	0.9
B50	20	101	703	4633	0.0	0.0	8	282	5419	7200	0.6	1.6
B100	20	139	936	5033	0.0	0.0	2	2814	6882	7200	1.5	2.8
B200	9	294	5039	7282	1.3	5.2	0	7200	7200	7200	3.1	5.1
MeiselKopfer	26	12	1429	7211	0.6	5.3	1	1645	7002	7200	2.6*	5.8*

*: Two instances are excluded in the average and maximum gap since the LP-Relaxation was not solved

Table 2 compares our results with the branch-and-price of Tilk *et al.* (2018). For these comparison, we use a hierarchical objective of first fulfilling as many requests as possible, then minimizing traveled distance, and then minimizing route completion time, i.e., $\alpha = 10$, $\beta = 1$, and $\gamma = 10000$.

The table contains, for both algorithms and each instance class, the number of instances solved to proven optimality, the minimum, average, and maximum runtime in seconds as well as the average and maximum gap between lower bound and best known solution at the end of the optimization.

The table shows that our algorithm outperforms the branch-and-price algorithm: 74 more instances can be solved to proven optimality and the runtime decreases by more than 75 % on average. Moreover, for all but one of the instances not solved to optimality, we provide new best known solution values that can be found in the detailed tables in the Appendix. Note that our approach could have a runtime of 4 hours (2 hours for each of the two phases) but the detailed results show that phase 1 takes 8 seconds on average with a maximum of 237 seconds. This also shows that the APVRP is much easier to solve if completion time is not a part of the objective.

8. Conclusion

This paper has investigated the exact solution of the APVRP by means of a branch-and-cut method based on Benders decomposition. The problem supports a flexible coupling for the operations and the movement of active and passive vehicles to achieve an efficient resource utilization and high-quality transport solutions. We have introduced a new compact formulation that build the basis for our Benders decomposition approach. Moreover, we showed how minimal responsible subsets can be derived from the solution of Benders subproblem and then be used to derive stronger optimality cuts. These lifted optimality cuts and the heuristic cut generation based on individual routes can accelerate the solution process. Both techniques can be adapted for general VRP variants that includes a timing component in the objective function. Computational experiments show that our method outperforms the previous state-of-the-art and is even capable of solving to optimality the original instances of Meisel and Kopfer (2014) that were designed for heuristics.

Due to the success of our approach, the next step will be adapting the algorithm to other VRP variants to clarify which aspects of VRP variants make the algorithm work well and were are its limitation. For the standard VRPTW, pretests have shown that the approach does not work quite well. Our intuition is that the more restricted the routes are due to resource consumption and the smaller the number of routes are, the better the approach will work in general.

Regarding algorithmic components, it could be of interest to solve Benders subproblem with a more sophisticated approach than just using a MIP-solver since it can be formulated as a system of differences for many VRP variants. This is of interest especially for those VRP variants with a more difficult subproblem. However, for the APVRP Benchmark instances, Gurobi solves Benders subproblem very quickly.

Acknowledgement

This research was partially supported by the Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/9-2. This support is gratefully acknowledged.

References

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, **36**(2), 69–79.

Benders, J. (1962). Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, **4**, 238–252.

Bodin, L. and Levy, L. (2000). Scheduling of local delivery carrier routes for the United States Postal Service. In M. Dror, editor, *Arc Routing: Theory, Solutions, and Applications*, chapter 11, pages 419–442. Kluwer, Boston.

Chao, I. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, **29**, 33–51.

Cheung, R. K., Shi, N., Powell, W.-B., and Simao, H. P. (2008). An attribute-decision model for cross-border drayage problem. *Transportation Research Part E: Logistics and Transportation Review*, **44**(2), 217–234.

Codato, G. and Fischetti, M. (2006). Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, **54**(4), 756–766.

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

Drexl, M. (2013). Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, **227**(2), 275–283.

Drexl, M. (2014). Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. *Networks*, **63**(1), 119–133.

Fischetti, M., Ljubić, I., and Sinnl, M. (2016). Benders decomposition without separability: A computational study for capacitated facility location problems. *European Journal of Operational Research*, **253**(3), 557–569.

Hooker, J. (2000). *Logic-based methods for optimization: combining optimization and constraint satisfaction*. John Wiley & Sons.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S., Toth, P., and Vigo, D. (2014). *The Family of Vehicle Routing Problems*, chapter 1, pages 1–33. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Lahyani, R., Khemakhem, M., and Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, **241**(1), 1–14.

Meisel, F. and Kopfer, H. (2014). Synchronized routing of active and passive means of transport. *OR Spectrum*, **36**(2), 297–322.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, **259**(3), 801–817.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, **49**(4), 258–272.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2018). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*. Forthcoming.

Semet, F. and Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, **41**(4), 469–488.

Smilowitz, K. (2006). Multi-resource routing with flexible tasks: An application in drayage operations. *IIE Transactions*, **38**(7), 555–568.

Soares, R., Marques, A., Amorim, P., and Rasinmäki, J. (2019). Multiple vehicle synchronisation in a full truckload pickup and delivery problem: A case-study in the biomass supply chain. *European Journal of Operational Research*, **277**(1), 174 – 194.

Tan, K., Chew, Y., and Lee, L. (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, **172**(3), 855–885.

Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., and Meisel, F. (2018). Branch-and-price-and-cut for the active-passive vehicle-routing problem. *Transportation Science*, **52**(2), 300–319. DOI: 10.1287/trsc.2016.0730.

Appendix

This section contains detailed computational results for all instances. For all experiments,a hierarchical objective of first fulfilling as many requests as possible, then minimizing traveled distance, and then minimizing route completion time was used (i.e., $\alpha = 10$, $\beta = 1$, and $\gamma = 10,000$). A hard CPU time limit of 2 hours for each phase of the algorithm was set. Tables 3 - 11 present the results obtained. The first column indicates the name of the instances, the second gives the value of the best known feasible solution computed by any of the various settings used in preliminary tests. The subsequent two columns show the value of the lower bound at the end of the optimization and the corresponding optimality gap in percent. Columns five and six indicate the CPU time

used in Phase 1 and the overall CPU time, respectively. The next two columns give the number of feasibility and optimality cuts generated during the course of the algorithm and the last column indicates the number of branch-and-bound nodes solved.

Table 3: Results on MeiselKopfer Instances

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
A01	13098	13098	0	11.3	207.8	3257	630	25581
A02	13067	13067	0	14.3	223.9	3319	1533	28059
A03	15383	14790	3.9	1.3	7201.3	11111	2428	289729
A04	14115	14115	0	20	143.4	2242	1014	14168
A05	16030	16030	0	3.3	664.3	7043	1282	109697
A06	10358	10358	0	2.5	11.8	233	86	783
A07	13848	13125	5.2	11.3	7211.4	15986	2924	236660
A08	15705	15705	0	0.6	268.7	3865	279	63081
A09	14430	13934	3.4	4.8	7205.1	18632	2629	217391
A10	16030	16030	0	5.5	319.7	4251	762	52493
A11	16132	16132	0	4	415	6809	601	54435
A12	14463	14463	0	3	32.4	931	128	8572
A13	15443	15443	0	32.9	462.6	5620	959	65222
A14	15057	15057	0	32.6	2982.8	20843	3837	248499
A15	12199	12199	0	25.9	139.5	2064	671	11702
A16	15983	15134	5.3	5.5	7205.6	18238	2709	150675
A17	15104	15104	0	3.1	36	932	180	7049
A18	11857	11857	0	1.2	32.4	846	100	4548
A19	15210	15210	0	5.5	197.1	4492	813	29604
A20	15023	15023	0	4.5	154.9	2551	350	25547
A21	14300	14300	0	7.2	241.8	4289	432	40048
A22	14302	14302	0	6.7	462.8	5499	1077	50483
A23	16121	16121	0	2.2	248.4	4463	482	47322
A24	15999	15999	0	15.5	470.7	4845	1114	58078
A25	18671	18671	0	12.4	2428.9	20101	1234	220014
A26	14344	14344	0	10.2	675.2	7623	1419	87184
A27	14170	14170	0	6.7	1941	14143	2017	145574
A28	13752	13752	0	17.2	316.5	4966	476	63026
A29	14246	14246	0	236.5	867.2	4623	4814	29055
A30	14876	14876	0	15.8	93.2	2629	1306	10133

Table 4: Results on 10-Request Instances with Time Window Flexibility 25

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
A01TW25	7721	7721	0	1.7	3.5	36	24	198
A02TW25	7577	7577	0	0.5	2.2	20	9	206
A03TW25	8198	8198	0	0.5	1.5	9	12	1
A04TW25	8114	8114	0	2.3	5.5	119	135	308
A05TW25	8861	8861	0	2.5	3.8	23	40	60
A06TW25	7233	7233	0	0.3	2.1	5	9	1
A07TW25	7767	7767	0	0.6	1.2	13	19	1
A08TW25	8168	8168	0	1.3	1.9	28	35	1
A09TW25	7653	7653	0	1.8	4.6	28	32	299
A10TW25	9583	9583	0	2.4	3.8	26	35	99
A11TW25	9237	9237	0	1.4	3.3	48	62	52
A12TW25	7720	7720	0	0.7	1.6	1	5	1
A13TW25	8791	8791	0	1.9	4.1	33	44	54
A14TW25	7632	7632	0	2.8	7.1	60	30	489
A15TW25	7539	7539	0	2.3	7.3	75	71	2125
A16TW25	7814	7814	0	1.8	4.7	53	45	389
A17TW25	8603	8603	0	1.5	5.2	46	48	342
A18TW25	6943	6943	0	0.3	1.4	6	3	22
A19TW25	9231	9231	0	3	5	102	81	262
A20TW25	8665	8665	0	3.2	4.1	90	95	1

Table 5: Results on 10-Request Instances with Time Window Flexibility 50

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
A01TW50	7671	7671	0	2.2	5	74	79	147
A02TW50	7527	7527	0	1.8	3.7	41	14	333
A03TW50	8165	8165	0	1.8	2.5	12	20	1
A04TW50	7972	7972	0	1.4	3.2	62	70	135
A05TW50	8662	8662	0	0.6	1.4	11	10	1
A06TW50	7183	7183	0	0.3	1.6	4	8	23
A07TW50	7742	7742	0	0.7	1.4	17	26	1
A08TW50	8131	8131	0	1.8	2.4	17	19	1
A09TW50	7448	7448	0	1.4	3.1	10	19	27
A10TW50	9489	9489	0	1.7	4.8	133	149	242
A11TW50	9124	9124	0	1.2	3.4	42	30	149
A12TW50	7670	7670	0	1.3	2.9	18	25	67
A13TW50	8741	8741	0	1.6	4.1	74	79	170
A14TW50	7453	7453	0	2.7	5.4	39	45	156
A15TW50	7454	7454	0	1.6	5.4	63	64	1567
A16TW50	7520	7520	0	1.4	2.3	18	26	1
A17TW50	8499	8499	0	1.8	3.9	25	32	89
A18TW50	6898	6898	0	0.3	1	5	3	1
A19TW50	9119	9119	0	3.8	7.2	256	269	809
A20TW50	8633	8633	0	4.7	7.5	98	96	36

Table 6: Results on 10-Request Instances with Time Window Flexibility 100

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
A01TW100	7562	7562	0	3.2	6.3	75	66	529
A02TW100	7477	7477	0	2.3	5.4	94	62	673
A03TW100	8140	8140	0	1	2.2	17	15	22
A04TW100	7852	7852	0	1.9	5	72	89	352
A05TW100	8537	8537	0	2	3.6	38	64	47
A06TW100	7122	7122	0	0.4	1.6	20	14	105
A07TW100	7448	7448	0	1	2.4	22	29	84
A08TW100	8050	8050	0	0.5	1	2	6	1
A09TW100	7287	7287	0	0.3	2	9	12	17
A10TW100	9054	9054	0	2.5	5.1	40	44	146
A11TW100	8993	8993	0	1.4	4.3	65	63	314
A12TW100	7626	7626	0	2	3.5	35	54	133
A13TW100	8533	8533	0	3	7.3	95	119	108
A14TW100	7400	7400	0	2.6	5.1	57	65	415
A15TW100	7368	7368	0	1.7	5	99	93	2131
A16TW100	7354	7354	0	2	3.6	23	32	49
A17TW100	8335	8335	0	1.2	4.1	22	19	113
A18TW100	6812	6812	0	0.7	2.2	14	15	58
A19TW100	8418	8418	0	3.1	7.4	131	95	305
A20TW100	8490	8490	0	2.8	5.6	102	85	337

Table 7: Results on 10-Request Instances with Time Window Flexibility 200

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
A01TW200	7081	7081	0	6.9	72.1	992	302	19763
A02TW200	7063	7063	0	6.9	24.4	435	301	2777
A03TW200	7565	7565	0	2.5	4.6	47	33	138
A04TW200	7534	7534	0	5.4	22.1	368	272	4086
A05TW200	7851	7851	0	2	3.8	33	12	86
A06TW200	6389	6389	0	0.6	4.3	63	16	356
A07TW200	7067	7067	0	3.4	14.7	177	118	1175
A08TW200	7759	7759	0	0.4	1.6	12	4	1
A09TW200	7158	7158	0	0.6	1.7	8	10	1
A10TW200	8141	8141	0	3.2	5.6	118	126	74
A11TW200	8472	8472	0	2.2	11.6	142	60	1236
A12TW200	7495	7495	0	0.4	2.4	47	49	295
A13TW200	8114	8114	0	5.3	26.3	281	224	1525
A14TW200	7154	7154	0	3.6	10.6	118	92	824
A15TW200	6804	6804	0	1	5.7	43	50	447
A16TW200	7109	7109	0	5.7	25.8	224	205	1559
A17TW200	7829	7829	0	6.5	12.2	120	106	508
A18TW200	6582	6582	0	3.7	10.8	167	70	870
A19TW200	8072	8072	0	8.5	49	796	456	12111
A20TW200	7842	7842	0	2.3	5.1	42	34	114

Table 8: Results on 20-Request Instances with Time Window Flexibility 25

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
B01TW25	16106	16106	0	3.2	428.5	268	136	222667
B02TW25	17617	17617	0	4.4	342.6	271	207	188612
B03TW25	15170	15170	0	5.6	694.6	250	106	370035
B04TW25	18085	18085	0	8.6	106.7	178	336	38015
B05TW25	17816	17816	0	4.2	583.1	564	309	263107
B06TW25	16743	16743	0	4.3	268.5	249	106	117072
B07TW25	16921	16921	0	1.2	96	192	33	27819
B08TW25	18508	18508	0	5.2	279.1	337	137	125787
B09TW25	18933	18933	0	3.4	69.9	97	169	20109
B10TW25	16382	16382	0	2.1	239.4	282	66	110168
B11TW25	17014	17014	0	6.3	92.1	134	169	28649
B12TW25	18028	18028	0	7.1	620.8	383	251	331890
B13TW25	16332	16332	0	2	162.3	183	67	72791
B14TW25	16721	16721	0	3.6	163.8	162	79	72053
B15TW25	18031	18031	0	7.1	1378.4	492	214	740603
B16TW25	16596	16596	0	3.8	124.8	198	153	45917
B17TW25	15587	15587	0	5.3	225.1	207	170	100635
B18TW25	17685	17685	0	4.2	92.8	140	59	33686
B19TW25	15804	15804	0	4	1050.6	461	175	517716
B20TW25	19398	19398	0	4	113.7	146	202	45723

Table 9: Results on 20-Request Instances with Time Window Flexibility 50

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
B01TW50	16004	16004	0	5.8	1250.5	482	232	618087
B02TW50	16838	16838	0	4.7	251.8	224	209	133413
B03TW50	15031	15031	0	6.7	838.5	276	119	462515
B04TW50	17564	17564	0	7.6	119.9	187	238	40467
B05TW50	17548	17548	0	7.1	573.1	725	426	219528
B06TW50	16691	16691	0	5.8	692.7	401	243	326707
B07TW50	16596	16596	0	1.3	126.7	187	33	36496
B08TW50	18139	18139	0	4.8	195.3	342	189	64313
B09TW50	18667	18667	0	3.8	100.8	113	141	39159
B10TW50	16330	16330	0	4.8	352.7	300	87	169241
B11TW50	16963	16963	0	4.4	198.9	200	109	100330
B12TW50	17952	17952	0	22	968.1	462	166	562948
B13TW50	16112	16112	0	2	140.9	215	96	57634
B14TW50	16519	16519	0	4.5	153.3	188	83	63918
B15TW50	17881	17881	0	17.2	4632.5	739	600	2160127
B16TW50	16318	16318	0	2.3	202	205	66	105139
B17TW50	15326	15326	0	6	573.4	284	162	272352
B18TW50	17509	17509	0	6.6	140.4	184	77	53605
B19TW50	15752	15752	0	7.3	2345.8	486	194	1385673
B20TW50	19169	19169	0	7.1	200	155	175	90348

Table 10: Results on 20-Request Instances with Time Window Flexibility 100

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
B01TW100	15792	15792	0	6.5	1089.4	545	347	593383
B02TW100	16591	16591	0	7.7	901.6	397	377	514036
B03TW100	14744	14744	0	2.2	551.3	382	128	290225
B04TW100	17372	17372	0	5.9	449.5	307	162	253978
B05TW100	17311	17311	0	7.3	914	794	360	450810
B06TW100	16545	16545	0	4.8	1264.1	552	166	614396
B07TW100	16496	16496	0	2.9	196.6	389	62	77417
B08TW100	17842	17842	0	10.3	942.9	733	395	405175
B09TW100	18546	18546	0	5	362.5	225	283	195958
B10TW100	16135	16135	0	5.9	2079.3	515	178	971398
B11TW100	16848	16848	0	4.6	278	240	137	142904
B12TW100	17339	17339	0	7.8	384.4	528	359	168671
B13TW100	15970	15970	0	5.5	357.4	376	168	188896
B14TW100	16250	16250	0	2.4	175.8	275	148	65110
B15TW100	17466	17466	0	10.9	5033.4	1024	346	2395957
B16TW100	15866	15866	0	3.1	138.6	199	129	63047
B17TW100	15228	15228	0	5.9	660.8	446	264	329397
B18TW100	17366	17366	0	5.1	248.6	324	126	120352
B19TW100	15626	15626	0	7.5	2218.7	575	258	1259231
B20TW100	18809	18809	0	5.9	468	273	212	224925

Table 11: Results on 20-Request Instances with Time Window Flexibility 200

instance	UB	LB	Gap [%]	Time [s]		No. cuts		No. nodes
				Phase 1	Total	Feas	Opt	
B01TW200	15145	15145	0.0	11.2	3257.3	937	481	1260260
B02TW200	16169	15856	1.9	25.6	7225.6	1597	959	1716160
B03TW200	14338	13946	2.7	7.2	7207.2	2073	505	1914973
B04TW200	16409	16409	0.0	3.9	1129.8	704	150	477937
B05TW200	16352	15793	3.4	81.6	7281.6	2502	1048	929236
B06TW200	16160	16160	0.0	11.5	6994.4	2329	932	2032689
B07TW200	15857	15857	0.0	4.8	1789.3	2874	339	490138
B08TW200	17508	17077	2.5	63.1	7263.1	3030	788	1571662
B09TW200	17586	17586	0.0	6.3	1662.1	539	282	769010
B10TW200	15354	14557	5.2	30	7230	4067	1096	1114467
B11TW200	16116	16031	0.5	12.6	7212.6	2009	1106	1902322
B12TW200	16649	16186	2.8	51.8	7251.8	2209	862	1519020
B13TW200	15631	15351	1.8	15	7215	2323	802	1836481
B14TW200	15605	15605	0.0	10.6	1998.5	932	483	757903
B15TW200	16439	15939	3.0	33.1	7233.1	2452	1003	1595776
B16TW200	15283	15164	0.8	12.3	7212.3	1200	908	2341372
B17TW200	14487	14487	0.0	6.9	294.3	718	167	99966
B18TW200	16771	16771	0.0	10.6	3090.8	796	277	1315616
B19TW200	15120	14842	1.8	7.3	7207.3	1560	393	2374325
B20TW200	17639	17639	0.0	7.7	1024.3	534	380	391013