TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Wirtschaftswissenschaften
Lehrstuhl für Operations Management

The Vehicle Routing Problem with Worker and Vehicle Synchronization: Metaheuristic and Branch-and-Price Approaches

Dipl.-Kfm. Univ. Martin Fink

Vollständiger Abdruck der von der Fakultät für Wirtschaftswissenschaften der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Martin Grunow
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Rainer Kolisch
	2.	Prof. Guy Desaulniers, PhD École Polytechnique de Montréal, Canada

Die Dissertation wurde am 09.08.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Wirtschaftswissenschaften am 15.09.2016 angenommen.

Acknowledgments

First of all, I would like to thank Prof. Dr. Rainer Kolisch for giving me the opportunity to conduct research in the field of vehicle routing problems applied to the fascinating aviation sector and allowing me to craft this dissertation at his chair of Operations Management. In addition, I want to thank Dr. Markus Frey and Dr. Ferdinand Kiermaier for advising, supporting and backing me during the whole dissertation process. Also, I would like to express my gratitude to both my second examiner Prof. Guy Desaulniers and to Prof. François Soumis for warmly welcoming me to Montréal and for their counsel and support during the second part of this dissertation. I also thank Prof. Dr. Martin Grunow, the chair of the examination committee. Special thanks go to my colleagues at the chair of Operations Management for establishing a great work environment with valuable discussions and many a good jest: Claus Brech, Alexander Döge, Dr. Thomas Fliedner, Dr. Markus Frey, Dr. Daniel Gartner, Gina Lambert, Dr. Ferdinand Kiermaier, Christian Ruf, Dr. Sebastian Schiffels and Dr. Martin Tritschler.

Moreover, I am very appreciative for having a host of friends, for their constant support and advice. Finally, I want to express my utmost gratitude to my girlfriend Romy Zschoche, to my sister Anja Herkelmann and to my parents Josef and Gerda Fink - I am deeply grateful for having you in my life.

Munich, July 2016

Abstract

English: With air traffic growing rapidly, the efficient planning of all ground handling processes is vital for airports and airlines to guarantee punctual flight operations. Based on the problem setting of a ground handling company at Munich International Airport, a new problem is introduced, coined the vehicle routing problem with worker and vehicle synchronization (VRPWVS). The VRPWVS belongs to the class of vehicle routing problems with multiple synchronization constraints (VRPMSs) and deals with routing workers to jobs while meeting each job's time window and incorporating each job's specific skill requirement. As large airports span vast distances and for security regulations, workers use vehicles to travel between jobs making movement synchronization in time and space necessary. In addition to movement, the VRPWVS is best characterized by the new concept of active load synchronization. As the VRPWVS covers all types of synchronization constraints, it is called a VRPMS archetype.

Examining the VRPWVS from an application perspective, both a mathematical model as well as hybrid metaheuristic with a tabu search, guided local search and large neighborhood search are proposed. The mathematical model can easily be transformed to a heuristic, which allows it to solve medium sized problems. The metaheuristic finds good results in very short time making it viable for a real-world rolling planning horizon application.

Investigating the VRPWVS from a theoretical perspective, two multi-commodity flow formulations based on time-space networks to efficiently model all synchronization types are presented. Additionally, a branch-and-price heuristic employing a novel fixing strategy is developed and it is shown that the heuristic finds near-optimal solutions in short time.

Deutsch: Die effiziente Planung von Bodenabfertigungsprozessen ist sowohl für Fluggesellschaften als auch -häfen von großer Bedeutung um einen reibungslosen Flugbetrieb gewährleisten zu können, gerade in Zeiten massiven Luftverkehrswachstums. Diese Dissertation betrachtet die Modellierung und Optimierung von operativen Bodenabfertigungprozessen als Tourenplanungsproblem mit Arbeiter und Fahrzeug Synchronisierung (VRPWVS) angelehnt an die Situation am Internationalen Flughafen München. Das VRPWVS gehört zur Klasse der Tourenplanungsprobleme mit mehreren Synchronisierungsnebenbedigungen (VRPMS). Es umfasst die Zuordnung von Arbeitern zu Bodenabfertigungsaufträgen und berücksichtigt dabei Zeitfenster und Qualifikationsanforderungen des jeweiligen Auftrags. Aufgrund von Sicherheitsvorschriften und großen Distanzen zwischen Gates an internationalen Flughäfen nutzen Arbeiter Fahrzeuge zur Fortbewegung, was deren zeitliche und örtliche Synchronisierung erforderlich macht. Neben der genannten Bewegungssynchronisierung lässt sich das VRPWVS am besten durch Synchronisierung von aktiver Ladung wie etwa Arbeitern beschreiben. Weil es alle Synchronisierungstypen abdeckt, ist das VRPWVS ein VRPMS Archetyp. In dieser Arbeit wird das VRPWVS aus einer Anwendungs- sowie einer theoretischen Perspektive heraus betrachtet.

Aus Anwendungssicht wird ein gemischt-ganzzahliges Problem definiert und eine Metaheuristik bestehen aus einer Tabu Suche, Guided Local Search und großer Nachbarschaftssuche entwickelt. Das gemischt-ganzzahlige Problem kann in eine Heuristik umgewandelt werden um mittelgroße Probleminstanzen zu lösen. Die Metaheurstik findet gute Lösungen in sehr kurzer Zeit und ist daher prädestiniert für eine Implementierung an Flughäfen.

Aus theoretischer Perspektive werden zwei Flussprobleme definiert welche auf zeitdiskreten Netzwerken basieren und alle Synchronisierungstypen effizient abbilden. Darüber hinaus wird ein Branch-and-Price Algorithmus entworfen, welcher sowohl auf Variablen verzweigt als auch eine neue Strategie zum Fixieren von Variablen umfasst. In einer experimentellen Studie wird gezeigt, dass der Algorithmus Lösungen nahe der optimalen Lösung in kurzer Zeit findet.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Global air traffic – both cargo and passenger – is growing at 5% annually according to Boeing Commercial Airplanes (2013). Airports are the key infrastructure on the ground to support aviation growth in the air and are highly likely to become a major bottleneck for future air transportation growth. Eurocontrol (2012) reports that by 2035 airports may not be able to satisfy up to 12% of demand, which leaves them with two viable options: first, to increase capacity by building new infrastructure such as terminals, aprons or runways; second, to better utilize current infrastructure by optimizing processes and operations such as ground handling.

Ground handling planning is conducted on strategic, tactical and operational levels (Clausen (2011), Herbers (2005)). Strategic ground handling planning is comprised of demand modeling and workforce sizing (see e.g. Herbers (2006)). Tactical ground handling planning takes the size and composition of the workforce as an input and creates concrete rosters and even shifts for workers encompassing holidays, days off or work regulations (see Kiermaier et al. (2016a)). Operational or real-time ground handling planning has two dimensions. First, making adjustments to tactical rosters and shifts (see Kiermaier et al. (2016b)). Second, assigning and routing workers to jobs, thus determining jobs' start and end times and aiming to minimize delays in ground handling processes. The assignment of workers to ground handling jobs is conducted in a rolling-planning horizon fashion to respond to flight schedule disruptions caused by e.g. bad weather or aircraft technical problems. Ground handling jobs include general services for aircraft, e.g. the unloading and loading of baggage and cargo as well as cleaning and refueling the airplane. Table 1.1 summarizes the three levels.

Table 1.1: Levels of ground handling planning

Level	**Planning horizon**	**Typical activities**
Strategic	Months to years	Demand modeling, workforce design
Tactical	Weeks to months	Rostering or shift design
Operational	Minutes to hours or days	Shift adjustments, task assignment

1.2 Scientific scope and contribution

This dissertation focuses on the operational or real-time assignment of workers to jobs inspired by the setting of Munich International Airport. In this work, the term operational ground handling planning therefore only refers to real-time worker and vehicle routing and job scheduling.

Once an aircraft arrives at an airport, workers with different skill levels are assigned to the airplane's jobs. For example, unloading baggage from a large aircraft like the Boeing 747 is conducted by three to four workers. The number of assigned workers influences a job's processing time. Once a job is completed, workers either move on to another job at the same or at another parking position, or return to the depot, a facility where ground handling workers start and end their shifts and wait until their next job assignment. Due to airports' vast dimensions, and for security reasons, workers use vehicles to travel between different parking positions. A worker either drives a vehicle or rides on a vehicle as a passenger. Hence, when assigning workers to jobs the resulting worker routes have to be synchronized with the vehicle routes. A good indicator of the efficiency of operational ground handling planning is a minimum number and magnitude of job delays as these delays may cause flight schedule disruptions, thus reducing the service level of an airport and airline and resulting in low customer satisfaction. To put a figure on it, the latest estimate from Cook et al. (2004) of the average flight delay cost is 72 Euro per minute, which can be taken as a conservative estimate and is likely higher today.

The outlined operational ground handling planning can be formalized as a vehicle routing problem (VRP) with multiple synchronization constraints (VRPMS). See Drexl (2012) for a more detailed introduction to VRPMSs. This class of rich VRPs is of high relevance for real-world applications and has only recently attracted the attention of the research community. The most complex problem type within the class of VRPMSs is named a VRPMS archetype: a VRP covering all synchronization constraints. Drexl (2007) proposed the first archetypal VRPMS, the VRP with trailers and transshipments (VRPTT). The VRPTT is motivated by milk collection in Bavaria an features two major extensions to the classical VRP: first, it encompasses active and passive vehicles. Active vehicles such as trucks can move on their own, whereas passive vehicles such as trailers, cannot move autonomously. Second, it considers so-called transshipment locations, where trailers can be stationed and decoupled from the truck or where load can be exchanged. In addition, each passive vehicle can be coupled with each active vehicle.

This dissertation makes several contributions: it introduces a second archetype, the VRP with worker and vehicle synchronization (VRPWVS), and while leveraging data from Munich International Airport, examines the VRPWVS from both an application and a theoretical perspective as outlined in the next two sections.

1.2.1 Application perspective

This section models the operational ground handling planning problem as the VRPWVS that extends the well-known VRP with time windows (VRPTW) which has been proven to be NP-hard (see Lenstra and Rinnooy Kan (1981)), making the VRPWVS also an NP-hard problem. The goal of the VRPWVS is to determine worker routes, job start times and processing times as well as to synchronize the worker routes with vehicle routes so that the delay of all jobs is minimized. It encompasses workers' skills and jobs' skill requirements and only assigns a worker to a job if the minimum required qualification level is met. The VRPWVS also includes vehicles' various capacities which indicate the number of workers that can be seated on the vehicle. To model movement synchronization, so-called transports are used, a combination of workers and vehicles that jointly travel between locations in specific time windows.

The well-known concept of load synchronization (see Drexl (2012)) is extended, which states that the amount of load transferred between two units, such as vehicles, needs to be synchronized, i.e. no load gets lost. Extending load synchronization means distinguishing between passive loads (e.g. milk or post) and active loads (e.g. workers, robots). Whereas a passive load is only transported from location to location and transferred between units, an active load is transported across locations to work on jobs and may have an influence on a job's processing time. For example, the more workers are assigned to refueling an aircraft, the quicker the job is completed. Moreover, active load can be characterized by unique features such as skills.

To solve the VRPWVS, a powerful hybrid metaheuristic is presented consisting of a tabu search (see Glover (1986)), guided local search (see Faroe et al. (2003) or Tsang and Voudouris (1997)) and large neighborhood search (see Shaw (1998), Franceschi et al. (2006)). It is the first efficient metaheuristic tailored to a VRPMS archetype. Moreover, the hybrid metaheuristic is suitable for a real-world rolling horizon application at airports because it produces good results in very short run times. The outlined application perspective chapter is based upon Fink et al. (2016b).

1.2.2 Theoretical perspective

In addition to the application perspective, this dissertation also tackles the VRPWVS from a more theoretical angle. To do so, the VRPWVS is slightly adjusted by excluding skills and different vehicle capacities. The emerging model is named abstract VRPWVS (AVRPWVS) and is still a VRPMS archetype. Two multi-commodity flow formulations based on non-trivial time-space networks are developed that efficiently model all five synchronization types.

The AVRPWVS is solved using a branch-and-price heuristic (BAPH), which employs column generation. For a general introduction to column generation, see Barnhart et al. (1998), Desrosiers and Lübbecke (2005) or Baldacci et al. (2012). The BAPH entails both branching and fixing: first, it branches on the number of employed vehicles and on jobs. Once these variables are integral, it starts fixing sums of so-called modified routes which represents a novel way of fixing columns. The presented BAPH is the first branch-and-price approach for an archetype of VRPMSs, and the BAPH can easily be extended to an exact branch-and-cut-and-price algorithm. The described theoretical perspective chapter is based upon Fink et al. (2016a).

1.3 Dissertation overview

This dissertation presents research from an airport application perspective in Chapter 2 and from a theoretical perspective in Chapter 3. Chapter 4 summarizes the findings.

Chapter 2

Synchronized Worker and Vehicle Synchronization for Ground Handling at Airports

2.1 Introduction

Global passenger and air-cargo traffic is growing at 5% p.a. (see Boeing Commercial Airplanes (2013)), increasing the need to efficiently plan all flight related operations at airports to avoid flight delays. Disruptions of the flight schedule not only lead to reduced service quality for the airport and the airline, but also result in significant penalty costs. Cook et al. (2004) estimate the average delay cost of a flight at 72 Euro per minute. According to Rapajic (2009), the main reason for flight delays is poor planning of the ground handling operations rather than external conditions like bad weather or technical problems with the airplanes.

Ground handling operations include general services for airplanes, e.g. the unloading and loading of baggage and cargo as well as cleaning and refueling of the airplane. The services (henceforth denoted as jobs) required for an aircraft need to be conducted in a given sequence, e.g. unloading of baggage has to be finished before new baggage can be loaded. Furthermore, to guarantee that each flight meets its schedule, jobs have to be finished within given time windows. As the ground handling market is very competitive and service quality is of prime importance, ground handlers seek to process aircraft with no delay subject to the scheduled workforce.

Once an airplane arrives at the airport, a dispatcher assigns workers with a mix of skills to the plane's jobs. For example, unloading baggage from a large aircraft like the Boeing 747 is conducted by three to four workers. One worker with a higher skill level navigates the baggage conveyor belt vehicle while two or three workers with lower skill levels unload the baggage. The number of assigned workers influences a job's processing time. Once a job is finished, a worker is either assigned to another job at the same parking position or proceeds to another aircraft to be processed at a different parking position. Due to airports' vast dimensions, and for security reasons, workers use vehicles to travel between different parking positions. A worker either drives a vehicle or rides on a vehicle as a passenger. Hence, when assigning workers to jobs the resulting worker routes have to be synchronized with the vehicle routes.

An airport is a dynamic and volatile environment. During the course of a day, flight delays or cancellations can occur requiring a replanning of flights' schedules and ground handling. Currently, the dispatcher assigns workers to an airplane's job once the plane arrives. As the ground handling workforce composition is based on a regular flight schedule (see Kiermaier et al. (2016a)), the current assignment procedure often fails to compensate for irregularities in the flight schedule, leading to a shortage of workers. As taking into account workers' travel distances and vehicle tours while considering subsequent jobs is a highly complex task for human dispatchers, workers often arrive later than planned at the plane's parking position to process jobs potentially leading to flight delays.

This paper makes three major contributions. First, we model the ground handling planning problem at airports as a rich vehicle routing problem with multiple synchronization constraints (VRPMS) and propose a mixed integer linear problem (MILP), the vehicle routing problem with worker and vehicle synchronization (VRPWVS). Second, we introduce a new type of synchronization, which we denote as "active load synchronization" extending the well-established concept of load synchronization. Classical VRPMSs deal with "passive load synchronization" where passive load, such as milk-containers or parcels, is synchronized with vehicles transporting the goods from one location to another (see Drexl (2012)). In contrast, ground handling planning deals with active load, i.e. workers, that actively influence the duration of a job. The connection between the number of workers assigned to a job and its duration also have a direct impact on the routing decisions for workers and vehicles. Generalizing the vehicle routing problem with time windows (VRPTW), the VRPWVS is NP-hard which makes the MILP intractable to solve real-world instances. Therefore, our third contribution consists of proposing a hybrid metaheuristic which is tailored to a real world application. In a computational study, we compare the hybrid metaheuristic with the MILP solved by the standard solver CPLEX, show the improvements achieved by the individual components of the metaheuristic and compare the hybrid procedure with the current planning procedure of the ground handling company.

The paper is structured as follows. In Section 2.2, we define the VRPWVS. A literature review is presented in Section 2.3, before we propose a MILP formulation of the VRPWVS in Section 2.4. A hybrid metaheuristic consisting of a construction procedure with look-ahead and an improvement procedure is outlined in Section 2.5. Subsequently, we present computational results for the MILP and the hybrid metaheuristic (Section 2.6). We recap our findings and suggest areas for further research in Section 2.7.

2.2 Problem description

Each airplane at an airport is associated with a specific parking position location and with a set of ground handling jobs, such as cleaning, refueling or baggage loading. A plane cannot depart before all jobs have been completed. The number of jobs and their processing times depend on the aircraft type, destination and number of arriving or departing passengers. For example, a wide-body aircraft such as the Airbus A380 is equipped with additional space for cargo in its cargo-hold, whereas a narrow-body aircraft like the Boeing B737 offers no extra space for cargo. The loading of baggage and cargo can only start once the unloading process has been completed. Thus, jobs may have precedence relations. To guarantee that an airplane leaves the airport on time, each job has to be started within a given time window which depends on the airplane's on block times.

Each job requires a minimum number of workers in order to be executed. To speed up a job's processing, it is possible to assign additional workers until the job is executed in its shortest possible processing time. Considering the example of unloading baggage from an aircraft, at least one worker is required and the number of workers can be increased by up to two workers to speed up the job. An additional third worker, however, would not lead to a reduction of the job's processing time due to the limited working space in the airplane's cargo-hold. Workers can only be assigned to a job, if the required skill level is met. Skill levels are ordered hierarchically, meaning that a worker with a specific skill level can handle jobs with an equal or lower skill level. For example, loading and unloading of baggage requires only the lowest skill level, whereas refueling an airplane has specific license requirements.

As soon as all workers assigned to a job have arrived at the respective airplane's parking position, the job is started. The processing of a job cannot be interrupted, nor can we change the number of assigned workers during the job's processing. The workers that are available in each time period are determined by the shift plans that define a shift start and end time for each worker. Each worker needs to be located at a depot at the beginning and end of the shift. Once workers have completed a job, they usually travel back to the depot if they are not assigned to a direct successor job.

To move from one position such as a gate or depot to another, workers use vehicles. The number of workers that can be carried by a vehicle is limited by the number of available seats. Once the vehicle arrives at a plane's parking position, workers either embark or disembark. A vehicle can only be moved if at least one worker, located at the same position as the vehicle, drives the vehicle. At the beginning of the day, all vehicles are located in the depot. If the distance between the current location of a worker and his next target location is below a certain threshold, the worker can also walk to the target location.

In summary, operational ground handling planning encompasses the routing of workers to and across parking positions so that all jobs are handled within their given time window. Moreover, to transfer workers from one position to another, worker routes have to be synchronized in time and space with vehicle routes. In order to minimize costs and to improve an airport's service level and hence reputation, efficient ground handling planning should seek to minimize the delay of jobs to enable timely flight operations. Delays of jobs are weighted with the job's associated aircraft size to account for the number of passengers that might suffer from schedule disruptions.

2.3 Literature review

The VRPWVS is an extension of the VRPTW. For a general introduction to VRPs, we refer to Toth and Vigo (2014), Golden et al. (2008), Braekers et al. (2015) and Laporte (2007). In this literature review, we first provide a discussion of VRPMSs before presenting recent literature on the operative planning of ground handling processes.

2.3.1 Related VRPMS literature

During the last couple of years, constructing and analyzing rich VRPs has been a major trend within the VRP research community. Rich VRPs typically extend the vehicle routing problem by additional constraints to model real-world problems in more detail (see for example Nguyen (2014) or Lahyani et al. (2015)). VRPMSs are well known examples of rich VRPs, encompass the VRPWVS and constitute a field with high practical relevance. However, according to Drexl (2012) only few literature exists in this field. To characterize VRPMSs, Drexl (2012) introduces five types of synchronizations: task, operation, movement, load and resource synchronization.

Resource synchronization means that the consumption of a given resource by all units is limited, where the term unit refers to workers or vehicles. For example in the VRPWVS, vehicle capacity is limited when workers want to travel on vehicles from the start depot to an aircraft's parking position. When several units coordinate their operations in time or space, we need operation synchronization. For example, before workers start from the depot to a parking position, they need to embark on a vehicle. Movement synchronization refers to at least two different types of units that require each other to be able to travel from one location to another. For instance, when workers and vehicles travel from the depot to the parking position, they need to synchronize in time and space. Task synchronization prevails when more than one unit is required to start a task and means that each task has to be started at the same time by each unit. In the VRPWVS, each job has to be completed once by a given number of workers. In our paper, we use the terms task and job interchangeably. Load synchronization requires that no load gets lost whenever any type of load is transferred between units and that the right amount of load is delivered to a customer. In the VRPWVS, the amount of workers that switch vehicles at any parking position or depot is synchronized.

The VRPWVS covers all five synchronization types. However, the established concept of load synchronization does not apply exactly to the VRPWVS. Therefore, we propose to extend load synchronization by distinguishing between active and passive load. *Passive load* mostly prevails in pick-up and delivery problems (PDP), such as for example raw milk or post. In contrast, *active load* is transported across locations to work on jobs and may have an influence on a job's processing time. For example, the more workers we assign to unload baggage, the faster the job is finished. In addition, active load can be characterized by unique features such as skills. Thus, assigning an active load unit to one job can lead to an infeasible solution for another job in case both jobs require the same active load unit.

Table 2.1 classifies related literature according to the features time windows (TW), precedence constraints between jobs (PC), multiple modes (MM), workers' skills (SK), active load synchronization (AL), passive load synchronization (PL) and movement synchronization (MS). The table reveals that there is no model formulation covering all features compared to our MILP. Introduced by Drexl (2007), the VRP with trailers and transshipments (VRPTT), a PDP with time windows and multiple synchronization constraints, features passive units (trailers) that require active units (lorries) to jointly collect passive load (raw milk) from customers and deliver it to a depot. Since the VRPWVS just like the VRPTT covers all five synchronization types, it is a VRPMS archetype.

Drexl (2014) provides several MILP formulations of the VRPTT and solves them using branch-and-cut algorithms. Drexl et al. (2013) investigate a combined vehicle and crew routing and scheduling problem in European road transport. A heterogeneous fleet of trucks and drivers need to jointly fulfill a set of pick-up and delivery tasks. As drivers may change trucks only at relay stations, we observe a movement synchronization type that is a combination of "en route" and "at depot" synchronization (indicated by a bracketed check mark in Table 2.1). The authors present a two-step heuristic based on large neighbourhood search to solve the problem. Mail distribution for Australia Post is the application for a vehicle and crew routing and scheduling problem described by Hollis et al. (2006). It is modeled as a PDP with time windows and with multiple depots. To solve real-world problem instances from Australia Post, the authors use a MILP with heuristic column generation. Kim et al. (2010) present a combined manpower scheduling and routing problem for multi-staged services that can be applied to the construction of roads. Each location requires a sequence of jobs that can only be performed by the right team. The authors present a MILP formulation for small instances and a dispatching-based heuristic algorithm to solve larger instances. In contrast to our model, Kim et al. (2010) do not incorporate time windows, which makes obtaining feasible solutions easier.

Table 2.1: Overview of VRPs with movement synchronization

Literature	**Jobs**				**Synchronization**		
	TW	PC	MM	SK	AL	PL	MS
Drexl (2007)	✓	(✓)	-	-	-	✓	✓
Drexl et al. (2013)	✓	(✓)	-	-	-	✓	(✓)
Hollis et al. (2006)	✓	(✓)	-	-	-	✓	✓
Kim et al. (2010)	-	✓	-	✓	(✓)	-	✓
Our model	✓	✓	✓	✓	✓	-	✓

We make two final comments on related VRPMS literature. First, the referenced literature on VRPMSs are PDPs except for Kim et al. (2010). For more information on PDPs, see Hooker and Natraj (1995) or Nowak et al. (2008). It should be noted that in our model, precedence constraints are more restrictive than in a PDP, because precedence in a PDP means merely that supply jobs need to be completed before delivery jobs (see bracketed checkmark). In our case, precedences can exist between each and every job. Second, we consider active load synchronization only as fully implemented if more than one active load unit such as a worker is allowed on a vehicle (see bracketed checkmark). The required coordination to plan one active load unit as for example in Kim et al. (2010) is much smaller than planning several, as in our case.

2.3.2 Related airport ground handling literature

There are only few publications on ground handling at airports in general.

Dohn et al. (2009) model the short-term planning of ground handling tasks using a manpower allocation problem with time windows and job-teaming constraints with the goal to maximize the number of assigned tasks. Teams are routed across jobs, but no supporting vehicles are employed. Therefore, no movement synchronization is required. The authors use Dantzig-Wolfe decomposition to solve data sets from two major European airports. Ip et al. (2013) describe a VRPTW with heterogeneous vehicles to model ground handling planning. It is solved using a genetic algorithm with adaptive search ability as well as with a genetic algorithm with greedy search. Andreatta et al. (2014) consider the allocation of ground handling staff and equipment to aircraft on the apron. They propose a mathematical formulation as well as a sequential heuristic to solve the problem which is to be integrated in a real-world planning system. Padrón et al. (2016) propose a constraint programming bi-objective approach to schedule ground handling vehicles and apply their solution method to real-world data from two Spanish airports. The four mentioned papers differ significantly from our approach because they do not incorporate skills, multiple modes, precedence constraints and movement or active load synchronization. In addition to research papers on ground handling at airports, there are two PhD theses on the topic. Clausen (2011) provides an overview of airport ground staff scheduling and covers related problems such as demand modeling, work-force planning or task scheduling. The author distinguishes between strategic, tactical, operative and real-time planning. He denotes that task scheduling is usually conducted on a real-time basis and most commonly modeled as a VRP. Herbers (2005) also considers ground staff scheduling at airports. He incorporates demand planning, shift planning and rostering. In contrast to Clausen (2011), Herbers (2005) does not cover the short-term planning of ground handling tasks at all.

2.4 Mathematical model

Section 2.4.1 introduces the required notation for the VRPWVS, including the concept of transports on which the model formulation presented in Section 2.4.2 is based. Section 2.4.3 provides an example of the proposed modeling concept.

2.4.1 Notation and model concept

Jobs. The number of jobs, e.g. loading of baggage, that have to be conducted during the course of the planning day is denoted by set $\mathcal{J} = \{1, \dots, n\}$. We define job 0 as a dummy job in the start depot and denote $\mathcal{J}^0 = \mathcal{J} \cup \{0\}$ the set of all jobs including job 0. Similarly, we set job $n+1$ as a dummy job in the end depot so that $\mathcal{J}^{n+1} = \mathcal{J} \cup \{n+1\}$. Each worker or vehicle tour starts at the start depot and ends at the end depot. Finally, we define the set of all jobs as $\mathcal{J}^{0,n+1} = \mathcal{J} \cup \{0, n+1\}$. To avoid flight delays, each job $i \in \mathcal{J}^{0,n+1}$ has to start within its time window $[ES_i, LS_i]$, with ES_i representing the earliest and LS_i the latest start time of job i. Each job $i \in \mathcal{J}^{n+1}$ has a set of possible modes $\mathcal{M}_i = \{m_i^{min}, \dots, m_i^{max}\}$ and is processed in exactly one mode $m \in \mathcal{M}_i$ meaning that m workers are assigned to job i resulting in an associated processing time $p_{i,m}$. All workers assigned to a job can arrive at its position at most τ periods before its start. Similarly, workers assigned to a job need to be picked up not later than τ periods after its completion.

We define the delay of a job i as the difference between its actual start with its earliest possible starting time ES_i. The higher this delay, the higher the penalty cost. Moreover, to consider the size of the aircraft, we penalize each delay with the job-dependent weights w_i.

Some jobs have precedence relations $< i, j >$, meaning that job j can only be started after job i has been completed. The set of precedences is given by set $\mathcal{E} \subset \mathcal{J} \times \mathcal{J}$.

Transports. Workers and vehicles need to be synchronized in time and space to jointly move from job to job, which is commonly named movement synchronization (Drexl (2012)). To model movement synchronization we introduce an aggregated formulation that we call "movement synchronization transport", henceforth simply referred to as "transport". A transport is defined as a combination of several workers and vehicles that team up to move from one parking position to another and is characterized by three features: i) By the departure job i and the arrival job j. ii) By a transport category that defines a relative departure time window in which the transport can depart from job i and a second relative arrival time window in which the transport can arrive at job j. iii) By the number of workers and vehicles that traverse from job i to j with the same transport category. In the following, we outline how transports represent an aggregated formulation of movement synchronization:

i) Transports aggregate workers and vehicles moving from job i to j.

ii) Transports aggregate the relative departure and arrival times of workers and vehicles by using nine different transport categories that we define for each transport in our model. A category is given by a relative departure and arrival job time window. Time windows are relative as they relate to the job's start and finish time. Figure 2.1 illustrates the categories. Categories $0, 1, 2, 3$ are most common and therefore marked in bold (see also computational study in Section 2.6.3). We denote the arrival time of a transport of category c at job i as $a^a_{i,c}$ and the departure time from job i as $a^d_{i,c}$. Given all nine categories $\mathcal{C} = \{0, 1, \cdots, 8\}$, we can define subsets based on the arrival or departure interval. For example, the subset of all transports arriving at a job before its processing is denoted as $\mathcal{C}^a_< = \{0, 1, 5\}$, the transports arriving before and during its processing as $\mathcal{C}^a_{<,-} = \{0, 1, 5, 4, 6, 8\}$. Similarly, we define $\mathcal{C}^a_> = \{2, 3, 7\}$, $\mathcal{C}^d_< = \{1, 3, 4\}$, $\mathcal{C}^d_{<,-} = \{1, 3, 4, 5, 6, 7\}$ and $\mathcal{C}^d_> = \{0, 2, 8\}$. To formalize transports, we introduce the binary decision variable $tr_{i,j,c}$ that equals 1 if a transport of category c goes from i to j, 0 otherwise.

iii) Transports ensure that the number of workers and vehicles on the transport match. Transports can contain both several workers and vehicles. The number of workers must not exceed the transport capacity (sum of the vehicle capacities) and must not be lower than the number of vehicles because each vehicle requires one driver.

It should be noted that the outlined transport concept can exclude certain solutions depending on the number of transport categories. Each job i features only one departure and arrival time for each transport of category c. Therefore, certain travel times from i to j might become impossible. For example, let two transports of the same category c leave from job i: one transport goes to job j_1, the second to job j_2. The variable for the departure time from job i of category c, $a^d_{i,c}$, will take the value of the latest of the two departure times. The outlined limitation can, however, be easily overcome if for a job i with several transports of the same category c, additional categories are introduced for each outgoing transport of the same category. Hence, determining the number of transport categories results in a trade-off between computational complexity and cutting-off feasible solutions. By limiting the number of transport categories, we can efficiently use the mathematical formulation to generate new heuristic solutions quickly as an element of our solution procedure presented in Section 2.5.

Depart i \ **Arrive at** j	Before start ($<$)	During proc. ($-$)	After end ($>$)
Before start ($<$)	**1**	4	**3**
During proc. ($-$)	5	6	7
After end ($>$)	**0**	8	**2**

Figure 2.1: Transport categories

Workers and vehicles. For workers and vehicles we use the sets $\mathcal{W} = \{1, \ldots, W\}$ and $\mathcal{K} = \{1, \ldots, K\}$. Workers' qualifications are denoted as $\mathcal{Q} = \{1, \ldots, Q\}$, and we use skill and qualification as synonyms throughout the paper. The qualification required for job i and provided by worker w is given by q_i^{i} and q_w^{w}, respectively. We employ hierarchical skills, which means that each skill level P dominates all lower skill levels $P-1, P-2, \ldots, 1$. Thus, when a worker is assigned to a job, $q_i^{\mathrm{i}} \leq q_w^{\mathrm{w}}$ must hold. To move between locations, workers have to embark on a vehicle. The maximum number of workers permitted on vehicle k is given by cap_k.

Given the earliest arrival time $EA_i = ES_i - \tau$ and the latest departure LD_i (defined by latest job finish time plus τ) for each job i, we know that any vehicle can only travel from job i to job j if $EA_i \leq LD_j$ holds, i.e. if the arrival and departure time windows of jobs i and j overlap. We can further tighten this using transport categories. In combination with the set of precedences $\mathcal{E}$, we can define a set of predecessor jobs $Pred_i$ for each job i and a set of successors $Succ_i$. $Succ_i^+$ denotes the set of successors of job i with a positive travel duration $d_{i,j} > 0$.

2.4.2 MILP formulation

Given the notation in the previous section and the problem description in Section 2.2, we present a mixed integer linear program in the following. It is based on the following decision variables:

Job variables

- t_i: Start time of job i
- e_i: End time of job i
- $y_{i,m}$: 1, if job i is performed in mode m, 0 otherwise

Worker variables

- $x_{w,i,j,c}$: 1, if worker w transfers from job i to job j on a transport of category c, 0 otherwise
- $b_{w,i}$: 1, if worker w is assigned to job i, 0 otherwise

- $u^{\mathrm{w}}_{w,i}$: Position of job i in the route of worker w

Vehicle variables

- $v_{k,i,j,c}$: 1, if vehicle k moves from job i to job j on a transport of category c, 0 otherwise
- $u^{\mathrm{k}}_{k,i}$: Position of job i in the route of vehicle k

Transport variables

- $tr_{i,j,c}$: 1, if there exists a transport from job i to job j of category c, 0 otherwise
- $a^{a}_{i,c}$: Arrival time at job i with transport of category c
- $a^{d}_{i,c}$: Departure time from job i with transport of category c

The routing variable $x_{w,i,j,c}$ denotes the routing of worker w, but does not describe a worker's assignment to a job. The assignment is accomplished by binary decision variable $b_{w,i}$. Due to the auxiliary integer variables $u^{\mathrm{w}}_{w,i}$ and $u^{\mathrm{k}}_{k,i}$ for workers and vehicles, subtours are eliminated. The variable stands for the position of job i within the route of vehicle k. For example, $u^{\mathrm{k}}_{k,3} = 2$ means that job 3 is the second job on the route of vehicle k. The model formulation for the vehicle routing problem with worker and vehicle synchronization (VRPWVS) can now be derived as follows.

$$\text{Minimize} \sum_{i \in \mathcal{J}^{n+1}} w_i \cdot (t_i - ES_i) \tag{2.1}$$

subject to

Job constraints:

$$t_i + \sum_{m \in \mathcal{M}_i} p_{i,m} \cdot y_{i,m} = e_i \qquad \forall i \in \mathcal{J}^{n+1} \tag{2.2}$$

$$e_i \leq t_j \qquad \forall\ (i,j) \in \mathcal{E} \tag{2.3}$$

$$\sum_{m \in \mathcal{M}_i} y_{i,m} = 1 \qquad \forall\, i \in \mathcal{J} \tag{2.4}$$

$$\sum_{m \in \mathcal{M}_i} m \cdot y_{i,m} = \sum_{w \in \mathcal{W}: q_i^{\mathrm{i}} \leq q_w^w} b_{w,i} \qquad \forall\, i \in \mathcal{J}^{n+1} \tag{2.5}$$

Worker constraints:

$$b_{w,j} \leq \sum_{i \in \mathcal{J}^0} \sum_{c \in C_<^a} x_{w,i,j,c} \qquad \forall\, w \in \mathcal{W}, j \in \mathcal{J}^{n+1} \tag{2.6}$$

$$b_{w,i} \leq \sum_{j \in \mathcal{J}^{n+1}} \sum_{c \in C_>^d} x_{w,i,j,c} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J}^0 \tag{2.7}$$

$$\sum_{j \in \mathcal{J}} \sum_{c \in \mathcal{C}} x_{w,0,j,c} = \sum_{i \in \mathcal{J}} \sum_{c \in \mathcal{C}} x_{w,i,n+1,c} = 1 \qquad \forall\, w \in \mathcal{W} \tag{2.8}$$

$$\sum_{j \in Pred_i} \sum_{c \in \mathcal{C}} x_{w,j,i,c} = \sum_{j \in Succ_i} \sum_{c \in \mathcal{C}} x_{w,i,j,c} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J} \tag{2.9}$$

$$\sum_{j \in Pred_i} \sum_{c \in \mathcal{C}_-^a} x_{w,j,i,c} \leq \sum_{j \in Succ_i} \sum_{c \in \mathcal{C}_{-,>}^d} x_{w,i,j,c} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J} \tag{2.10}$$

$$\sum_{j \in Pred_i} \sum_{c \in \mathcal{C}_>^a} x_{w,j,i,c} \leq \sum_{j \in Succ_i} \sum_{c \in \mathcal{C}_>^d} x_{w,i,j,c} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J} \tag{2.11}$$

$$u_{w,i}^{\mathrm{w}} - u_{w,j}^{\mathrm{w}} + n \cdot \sum_{c \in \mathcal{C}} x_{w,i,j,c} \leq n - 1 \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J}^0, j \in Succ_i \tag{2.12}$$

$$x_{w,i,j,c} \leq tr_{i,j,c} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J}^0, j \in Succ_i, c \in \mathcal{C} \tag{2.13}$$

$$tr_{i,j,c} \leq \sum_{w \in \mathcal{W}} x_{w,i,j,c} \qquad \forall\, i \in \mathcal{J}^0, j \in Succ_i, c \in \mathcal{C} \tag{2.14}$$

Transport constraints:

$$\sum_{w \in \mathcal{W}} x_{w,i,j,c} \leq \sum_{k \in \mathcal{K}} cap_k \cdot v_{k,i,j,c} \qquad \forall\, i \in \mathcal{J}^0, j \in Succ_i^+, c \in \mathcal{C} \tag{2.15}$$

$$\sum_{k \in \mathcal{K}} v_{k,i,j,c} \leq \sum_{w \in \mathcal{W}} x_{w,i,j,c} \qquad \forall\, i \in \mathcal{J}^0, j \in Succ_i^+, c \in \mathcal{C} \tag{2.16}$$

$$a_{i,c}^d + d_{i,j} - a_{j,c}^a \leq M \cdot (1 - tr_{i,j,c}) \qquad \forall\, i \in \mathcal{J}^0, j \in Succ_i, c \in \mathcal{C} \tag{2.17}$$

$$a_{i,c}^a \leq a_{i,\tilde{c}}^d \qquad \forall i \in \mathcal{J}, c \in \mathcal{C}_<^a, \tilde{c} \in \mathcal{C}_<^d \tag{2.18}$$

$$a_{i,c}^a \leq a_{i,\tilde{c}}^d \qquad \forall i \in \mathcal{J}, c \in \mathcal{C}_-^a, \tilde{c} \in \mathcal{C}_-^d \tag{2.19}$$

$$a^a_{i,c} \leq a^d_{i,\tilde{c}} \qquad \forall i \in \mathcal{J}, c \in \mathcal{C}^a_>, \tilde{c} \in \mathcal{C}^d_> \tag{2.20}$$

$$a^a_{i,c} \leq t_i \qquad \forall\ i \in \mathcal{J}^{n+1}, c \in \mathcal{C}^a_< \tag{2.21}$$

$$t_i \leq a^a_{i,c} \leq e_i \qquad \forall\ i \in \mathcal{J}^{n+1}, c \in \mathcal{C}^a_- \tag{2.22}$$

$$e_i \leq a^a_{i,c} \qquad \forall\ i \in \mathcal{J}^{n+1}, c \in \mathcal{C}^a_> \tag{2.23}$$

$$a^d_{i,c} \leq t_i \qquad \forall\ i \in \mathcal{J}^0, c \in \mathcal{C}^d_< \tag{2.24}$$

$$t_i \leq a^d_{i,c} \leq e_i \qquad \forall\ i \in \mathcal{J}^0, c \in \mathcal{C}^d_- \tag{2.25}$$

$$e_i \leq a^d_{i,c} \qquad \forall\ i \in \mathcal{J}^0, c \in \mathcal{C}^d_> \tag{2.26}$$

Vehicle constraints:

$$v_{k,i,j,c} \leq tr_{i,j,c} \qquad \forall\ k \in \mathcal{K}, i \in \mathcal{J}^0, j \in Succ_i, c \in \mathcal{C} \tag{2.27}$$

$$u^{\mathrm{k}}_{k,i} - u^{\mathrm{k}}_{k,j} + n \cdot \sum_{c \in \mathcal{C}} v_{k,i,j,c} \leq n - 1 \qquad \forall\ k \in \mathcal{K}, i \in \mathcal{J}^0, j \in Succ_i \tag{2.28}$$

$$\sum_{j \in \mathcal{J}\setminus\{0\}} \sum_{c \in \mathcal{C}} v_{k,0,j,c} \leq 1 \qquad \forall\ k \in \mathcal{K} \tag{2.29}$$

$$\sum_{j \in \mathcal{J}\setminus\{0\}} \sum_{c \in \mathcal{C}} v_{k,0,j,c} \leq \sum_{j \in \mathcal{J}\setminus\{n+1\}} \sum_{c \in \mathcal{C}} v_{k,j,n+1,c} \qquad \forall\ k \in \mathcal{K} \tag{2.30}$$

The objective of the model is to minimize the sum of the weighted job delays. The job weight w_i is defined by the aircraft size associated with job i. Constraints can be clustered into job (2.2 - 2.5), worker (2.6 - 2.14), transport (2.15 - 2.26) and vehicle constraints (2.27 - C.3) as well as variable declarations (C.4 - C.14). Note that the variable declarations and part of the vehicle constraints are included only in the appendix as they correspond to the worker routing.

Job constraints (2.2)-(2.5): Constraint (2.2) states that the job end time e_i equals the start time plus the processing time of job i. Constraint (2.3) represents the precedence constraint. (2.4) ensures that each job is processed once in one mode. The correct number of workers with suitable skills needs to be assigned to each job (2.5).

Worker constraints (2.6 - 2.14): Constraint (2.6) ensures that a worker w assigned to job j arrives at the job's location prior to the job's start, i.e. on a transport of category $0, 1$ or 5. Similarly, a worker assigned to i needs to leave after the job is completed using a transport of type $0, 2$ or 8, which is represented by constraint (2.7). Each worker's tour must start and end at the depot (2.8). The flow conservation of workers is defined in constraint (2.9). Workers arriving during the processing of job j (i.e. with a transport of category $4, 6$ or 8) leave the job either during or after its processing (2.10). The same logic applies analogously to workers arriving after a job (2.11). Constraint (2.12) is the classical subtour elimination constraint as introduced by Miller et al. (1960) and adapted by Kara et al. (2004). Constraints (2.13) and (2.14) connect the transport variable $tr_{i,j,c}$ with the worker routing variable $x_{w,i,j,c}$.

Transport constraints (2.15 - 2.26): The number of workers on a transport must be less than or equal to the transport capacity (2.15) and equal to or greater than the number of vehicles to guarantee that there is always at least one driver per vehicle (2.16). Note that, in the above stated model, workers need not synchronize with vehicles but may walk to jobs in close distance to the position of the current job i. The set of jobs for which workers and vehicles need to synchronize starting from job i is given by the set of successors $Succ_i^+$, meaning that workers can walk to jobs at the same parking position. A transport arrives at job j after it left job i plus the travel time needed to go from i to j (2.17). For each job i, the arrival time of a transport needs to be less than or equal to the departure time of transports leaving i. Constraint (2.18) enforces this circumstance for any transport arriving and leaving before the job start t_i. Similar holds for arriving and leaving during a job (2.19) and after a job (2.20). The arrival and departure times for the different transport categories correspond to Figure 2.1 and are given in constraints (2.21) - (2.26).

Vehicle constraints (2.27 - C.3): We use constraint (2.27) to connect the vehicle routing variable $v_{k,i,j,c}$ with the transport routing variable $tr_{i,j,c}$. Constraint (2.28) is the subtour elimination constraint for vehicles. Constraint (2.29) states that a vehicle can but is not required to leave the depot. If it does, constraint (2.30) ensures that the vehicle also returns to the depot. To ensure a proper vehicle routing, we use constraints (C.1) - (C.3) that are similar to the routing of workers. Therefore, they can be found in the appendix, just like the variable declarations.

The VRPWVS is a generalization of the well-known vehicle routing problem with time windows, which has been proven to be NP-hard (Lenstra and Rinnooy Kan (1981)). Thus, the VRPWVS belongs to the class of NP-hard problems. Whereas there exist several exact and heuristic solution procedures to solve medium-sized problem instances for the VRPTW (see, e.g., Desrosiers et al. (1984), Toth and Vigo (2014) or Baldacci et al. (2012)), these procedures are not applicable to the VRPWVS as the latter is a generalization of the VRPTW and as the VRPWVS essentially features two VRPs (one for workers and one for vehicles) that need to be synchronized because a change in one route can also affect the feasibility of another route. Thus, the VRPWVS is more complex than the VRPTW. In addition to the relation to the VRPTW, it should be noted that the VRPWVS has many similarities with the multi-mode resource constrained project scheduling problem such as precedence constraints and multi-mode jobs (see Kolisch and Sprecher (1997), Brucker et al. (1999) or Hartmann and Briskorn (2010)).

2.4.3 Example

To illustrate the VRPWVS, we consider a small example (see Figure 2.2) with two aircraft 001 and 007 calling at gates 1 and 2. For each aircraft, we need to conduct two jobs, namely loading off and loading on baggage (jobs 1 and 2 for aircraft 001 and 3 and 4 for 007). To model the need for first performing the loading off baggage job before the loading on baggage job, precedence relations exist for jobs 1 and 2, and 3 and 4, respectively (denoted as arrows in Figure 2.2). Three workers $\mathcal{W} = \{1, 2, 3\}$ and two vehicles $\mathcal{K} = \{1, 2\}$ are located at the depot. Each job i features a time window from ES_i to LS_i, and several modes with associated number of workers m and processing times $p_{i,m}$. Distances between the gates are given by a distance matrix not shown in the graph. Vehicles have a capacity of 2 workers. To keep the example simple, we exclude worker skills by assuming that each worker can work on each job and that each job has only one mode.

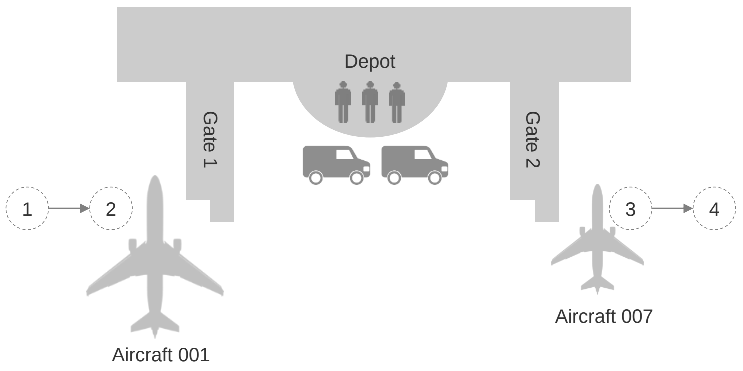

Figure 2.2: Example

In the following we describe a feasible solution for the example using the concept of transports. In time 1, worker 1 and 2 embark on vehicle 1 at the depot to travel to job 1. They arrive in time 4 (assuming a travel time of 3) and wait before the start of the job in time 5, which means that the transport from 0 to 1 is of category 0. Worker 3 travels with vehicle 2 to gate 2 to work on job 3. Having completed job 1, worker 1 uses vehicle 1 to move to gate 2, whereas worker 2 stays at gate 1 to work on job 2. Arriving in time 9 when job 3 is still ongoing (i.e. the transport from job 1 to 3 has category 8), worker 1 waits until time 10 to start work on job 4 jointly with worker 3.

The remainder of the description of the transports and job starts can be seen in Tables 2.2 and 2.3.

Assuming equal weights of $w_i = 1$ for all jobs, the resulting overall delay across the four jobs and the dummy end job is calculated to $2 + 3 = 5$.

2.5 Hybrid metaheuristic

The literature on related VRPMSs with movement and load synchronization offers a variety of solution procedures (see Drexl (2012) and Table 2.4).

Table 2.2: Example: transports of feasible solution

i	j	c	$a^d_{i,c}$	$a^a_{j,c}$	$\mathcal{W}$	$\mathcal{K}$
0	1	0	1	4	$1,2$	1
0	3	0	1	4	3	2
1	3	8	7	9	1	1
1	2	0	7	7	2	–
3	4	0	10	10	$1,3$	$1,2$
4	2	2	12	14	$1,3$	$1,2$
2	5	0	14	17	$1,2,3$	$1,2$

Table 2.3: Example: job data of feasible solution

Job	0	1	2	3	4	5
m	3	2	1	1	2	3
p_i	0	2	5	3	2	0
t_i	1	5	7	7	10	17
ES_i	1	5	7	7	8	14
LS_i	1	8	10	10	11	18
$t_i - ES_i$	–	0	0	0	2	3

Table 2.4: Overview of solution procedures for VRPMSs

Literature	**Specific solution procedures**
Drexl et al. (2013)	Two-step heuristic based on large neighborhood search
Drexl (2014)	None, but several branch and cut formulations
Hollis et al. (2006)	Heuristic column generation
Kim et al. (2010)	Greedy heuristic and particle swarm optimization

To solve the VRPWVS, we propose a hybrid metaheuristic consisting of a look-ahead construction heuristic and an improvement heuristic to find good solutions in reasonable time. The latter encompasses a tabu search (TS), guided local search (GLS) and large neighborhood search (LNS).

The selection of the hybrid metaheuristic is motivated by the large problem size of the VRPWVS and because we aim for a procedure that is suitable for a rolling planning horizon and therefore has a short runtime. In addition, according to Drexl (2012), there exists no powerful metaheuristic to solve an archetype of VRPMSs, such as the VRPWVS. A tabu search is implemented in our procedure since it has been applied successfully to other rich VRPs (see for example Nguyen (2014)). Allowing infeasible solutions is crucial for solving the VRPWVS because the problem can easily become infeasible. Therefore, GLS is employed as it allows for infeasible solutions. LNS is used as it performs well with VRPMSs and especially movement synchronization (see Drexl (2012), Meisel and Kopfer (2014)). We outline the construction heuristic in the next section and the improvement heuristic in Section 2.5.2.

2.5.1 Construction heuristic

The goal of the construction heuristic is to find an initial feasible solution with a good solution value. The construction heuristic employs two job sets: the supply job set encompasses jobs which have been performed and where workers or vehicles are still stationed at the job's parking position. The demand job set contains jobs that still need to be undertaken. The core idea is to iteratively create transports of category 0 from supply jobs to demand jobs. The procedure is structured as follows:

1. Set an initial value for the look-ahead depth k as well as a maximum look-ahead depth k_{max} (see Section 2.5.1.1 for an explanation of k).

2. Start with an empty solution in which all workers and vehicles are located at the start depot and add the dummy job at the start depot to the supply job set. All other jobs are in the demand job set, sorted in increasing order by their latest start time and incorporating precedence relationships. We use the job number i as a tie breaker.

3. While the demand job set is not empty, create new transports from jobs in the supply job set to jobs in the demand job set and update the three lists (for details see Section 2.5.1.1).

4. Stop if the solution is feasible. Otherwise, if $k < k_{max}$, increment k and go back to step 2, otherwise stop.

If the procedure stops with $k > k_{max}$ without a feasible solution, the best known infeasible solution (that is, the infeasible solution with the lowest solution value (2.32)) is selected as a start solution. In the objective function (2.32), infeasibility is taken into account by penalty terms. Section 2.5.2.1 outlines this concept in greater detail.

2.5.1.1 Look-ahead procedure

To find a feasible solution, we need to carefully select the sequence in which demand jobs are chosen (see step 3 above). To achieve this, we use a look-ahead procedure that is basically a limited enumeration of partial solutions.

First, the procedure builds a search tree consisting of the next k demand jobs and their respective modes using the supply job selected in the last iteration as the root node on level 0. A node is defined as a job with a specific mode. The procedure branches creating new nodes when two different jobs have the same latest start time or when one job has several modes. The search tree grows with k as does the computation time. On the other hand, the quality of the selection of the next demand job increases with k as well.

When the tree is built, each path is examined. During the examination of a path, the procedure iteratively aims to supply each node on the path with workers and vehicles. It starts with the node on level 1 and proceeds until the lowest level, which means we use a depth-first strategy. Examining a path can either yield feasibility (i.e. each node on the path can be supplied with workers within the time window) or infeasibility. In the first case, we compute the solution of the partial solution, while in the second case we mark the first node in the path that cannot be supplied in time and fathom all branches below it.

To determine which node to choose, we count the number of feasible paths that start from a level 1 node and select the node with the most feasible paths. As a tie breaker, we use the minimum weighted delay of all feasible paths.

2.5.1.2 Example

Applying the construction heuristic to the example given in Section 2.4.3 we obtain the supply job set $\{0\}$. Given $k = 3$, the demand job set equals jobs $\{1, 2, 3\}$ in the sequence $1, 2, 3$ due to the jobs' latest starts. Assuming that all three jobs have two modes and that job 2 and 3 have the same latest start, we can devise a search tree with levels 0 to $k = 3$ and 16 branches. Figure 2.3 illustrates one part of the tree. It shows that we can either choose node 1(1) (representing job 1 in mode 1) or node 1(2). Assuming that 1(1) has 3 feasible branches (see checkmarks in Figure 2.3) and 1(2) none, node 1(1) would be selected.

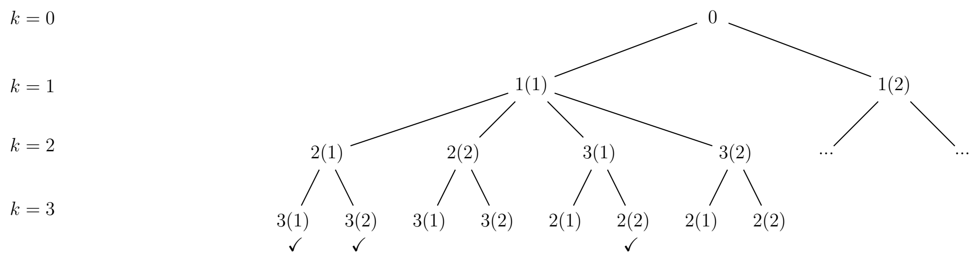

Figure 2.3: Construction heuristic search tree for example

2.5.2 Improvement heuristic

We propose an improvement heuristic with three major components: tabu search (TS) combined with guided local search (GLS) and large neighborhood search (LNS). The TS incorporates two local search (LS) moves specifically tailored to our problem.

The concept of the overall hybrid metaheuristic is given in Figure 2.4, including the construction heuristic with look-ahead (CH). A solution is either obtained by the construction heuristic or the last iteration of the improvement heuristic. If the solution is infeasible or if no improvement greater than Δ_{LNS} has been achieved during the last $iterations_{LNS}$ iterations, we employ LNS to destroy parts of the solution and repair it using the MILP. We use the LNS to make larger moves in the solution space for diversification and to resolve infeasibility. If no LNS is called, the procedure uses LS with transport swap moves to examine the neighborhood of the solution. We perform a number of n_S^{sol} transport swaps per solution because it is costly to update the routes and start times of a solution after each transport swap. If one of the emerging solutions has jobs that have too few workers, we perform a worker shift move of the LS to allocate more workers to these jobs and thus restore feasibility. For all generated solutions, whether resulting from the LNS or the LS, a decision by the tabu search mechanism is made regarding which solution is used next. To allow infeasible solutions, GLS is employed. The different parts of the improvement heuristic are described in more detail in the following.

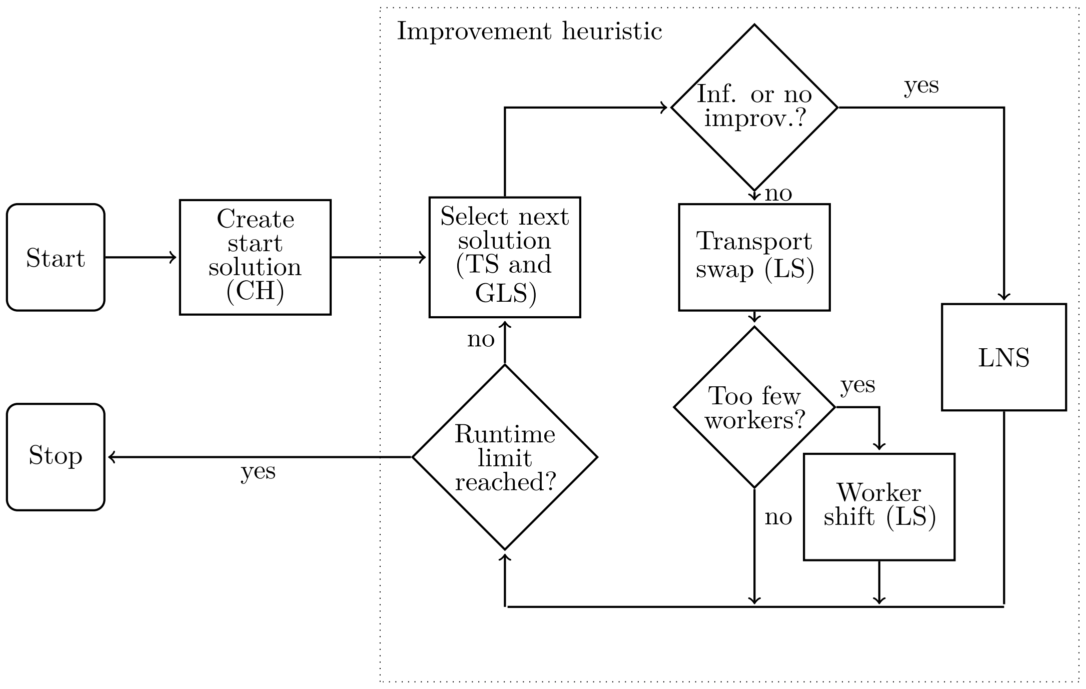

Figure 2.4: Hybrid metaheuristic illustration

2.5.2.1 Guided local search (GLS)

Solutions for the VRPWVS can easily become infeasible. Thus, our hybrid metaheuristic allows certain infeasible solutions by incorporating penalties into the objective function (see Faroe et al. (2003), Voudouris and Tsang (1999) and Tsang and Voudouris (1997)). The GLS forces local search schemes to move away from known solutions. In case an infeasible solution is found that violates a certain feature, a penalty term is added to the objective function $f(x)$ to obtain an augmented objective function $h(x)$

$$h(x) = f(x) + \lambda \cdot \sum_{\phi \in \Phi} c_\phi \cdot I_\phi(x) \tag{2.31}$$

where c_ϕ denotes the penalty cost term of a feature ϕ, and $I_\phi(x)$ is 1 if solution x has feature ϕ, 0 otherwise. λ is the penalty weight.

A feature is defined as a characteristic of a problem and we penalize two types of features. First, jobs with fewer workers than the minimum mode permits; second, vehicles with more workers than its capacity allows for. For infeasible job assignments, let indicator function $I_i^{job}(x)$ be 1 if solution x violates the minimum allowed number of workers for job i. For infeasible vehicle assignments, let $I_{k,i}^{cap}(x)$ be 1 if solution x exceeds the allowed capacity on vehicle k after job i. The penalty or cost terms are written as c_i^{job} and $c_{k,i}^{cap}$. The augmented objective function (2.31) can now be rewritten as

$$h(x) = f(x) + \lambda \cdot (\sum_{i \in J} c_i^{job} \cdot I_i^{job}(x) + \sum_{k \in K} \sum_{i \in J} c_{k,i}^{cap} \cdot I_{k,i}^{cap}(x)) \tag{2.32}$$

The two penalty terms are initially set to zero. During the course of the hybrid metaheuristic, each penalty term is increased by 1 if a solution violates the associated constraint.

2.5.2.2 Tabu search

Tabu search (TS), a metaheuristic introduced by Glover (1986), is employed to guide the procedure through the solution space and to steer the selection of solutions using two features. First, TS enables overcoming local optima by permitting non-improving local search moves. Second, it prevents cycling back to already known solutions by employing a list of forbidden moves: the tabu list.

Until a given time limit is reached, the TS initiates either LS or LNS to generate new solutions. The LS procedure makes use of two types of moves: transport swaps and worker shifts. In order to not repeat or reverse a LS move, a tabu list with length l_{TS} is maintained. The LS moves are described in Section 2.5.2.3. The TS then examines all newly found solutions for feasibility and objective function value. If there is a current feasible solution with a lower objective function value than the best known feasible solution, the current solution becomes the best solution and the best objective function value is updated. In the next iteration, the neighborhood of the current solution is examined. If the solution is feasible except for at least one of the two GLS features, we name it a penalty feasible solution. If a penalty feasible solution has a lower objective value than the best found feasible solution, we continue to explore the neighborhood of the penalty feasible solution. Recall that, according to GLS, when a GLS feature is violated, we increment the respective penalty term by 1.

2.5.2.3 LS moves

Local search is a common approach to examine the neighborhood of a solution using moves tailored to the problem. We employ two types of moves: transport swaps and worker shifts.

Transport swap A transport swap exchanges workers and vehicles from one transport with those of another transport and thus affects both worker and vehicle routes. Per iteration, we perform n_S^{it} swaps.

A transport swap selects two transports and exchanges workers and vehicles. After the exchange, workers' and vehicles' routes need to be updated. Before we explain the swap in detail, let us briefly introduce our notation. Let transport tr_1 be defined by the jobs i_1, j_1 and category c_1, and transport tr_2 by jobs i_2 and j_2 and category c_2. In addition, let h be the predecessor of i_1 and let k the successor of j_1, so that workers and vehicles of tr_1 have the route h, i_1, j_1, k

before the swap. Figure 2.5 illustrates this route as a solid line and the second transport tr_2 as a dotted line. There are two ways to update the routes and connect job i_2 with h (the same holds for connecting j_2 to k). First, we can redirect workers and vehicles so that their path goes directly from h to i_2 instead of to i_1 (see the dashed line in Figure 2.5). If this is not possible, e.g. due to workers still disembarking at i_1, we create an additional transport i_1 to i_2 of category 1 (i.e. after the end of i_1 and before the start of i_2) so that we create a detour h, i_1, i_2. The same logic applies to k, except that we would create an additional transport of category 2 (i.e. after the end of j_2 and j_1) for a detour.

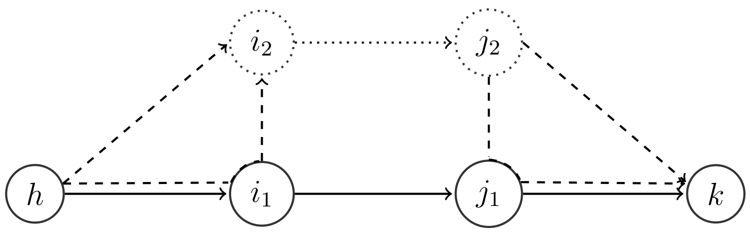

Figure 2.5: Transport swap illustration

Worker shift After a transport swap, some jobs may have fewer workers than their lowest mode permits, which leads to an infeasible solution. To mediate this problem, a worker shift is employed to redirect workers to jobs with too few workers.

For each job g_1 with too few workers, we find another job g_2 with a mode above the minimum mode and an overlapping time window. A worker shift selects a worker w from g_2 that meets the minimum skill required by g_1 and assigns w to g_1. As with the transport swap, we update the route of the worker after the shift. In contrast to the transport swap, we need to use existing vehicle routes (except for the rare case when the worker is permitted to walk) to get to g_1. Again, let h be the predecessor and k be the successor of job g_2. The route before the update is denoted as a solid line in Figure 2.6. A worker has three ways to reach g_1. First, by walking directly from g_2 to g_1, or second, by walking from the predecessor h to g_1. These two possibilities are simple, but happen rarely due to large distances at airports. The third option is to find an intersection of the current worker route with a vehicle route leading to g_1 (marked as a dashed line) before job g_2. Let us denote the intersection or crossing job as x. We then replace all jobs of the worker route from x to g_2 with all jobs of the route from x to g_1. However, the shift is only possible if the modes of jobs in between x and g_2 can be decreased (if the worker indeed works on the jobs) and if the new vehicle still has capacity for an additional worker.

Routing the worker from g_1 to g_2 works in a similar way. Figure 2.6 illustrates a worker shift and shows the third possibility for connecting job g_1 to the worker route.

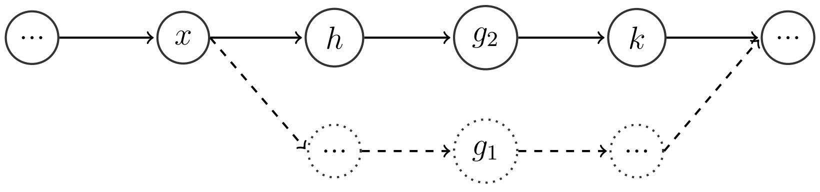

Figure 2.6: Worker shift illustration

2.5.2.4 Large neighborhood search

The concept of large neighborhood search has been proposed by Shaw (1998). LNS improves a given solution by iteratively destroying and repairing it, thus reaching a large neighbourhood of solutions. We employ an adjusted version of the MILP, named the LNSMILP, to repair destroyed solutions (see Franceschi et al. (2006)) and achieve diversification.

Destroy move We define a destroy move as randomly selecting a number of jobs from an existing solution and destroying them by removing them from the respective worker and vehicle routes. As a result, destroyed jobs are not supplied with any workers or vehicles and have no determined mode or start time, resulting in a partial solution. The destroy move is an important part of the LNS: if too small a part of the solution is destroyed, the LNS does not reach distant areas of the solution space. However, if too large a number of jobs is chosen, the repair move requires too long a processing time. We employ the approach introduced by Shaw (1998) to increase the percentage of jobs to be destroyed from a destroy start percentage ρ_{start} until ρ_{end}.

Repair move The repair move receives a partial solution from the destroy move and determines values for all variables associated with destroyed jobs. Given that finding a feasible solution for the VRPWVS is difficult, we employ the LNSMILP to find a feasible solution instead of a specific repair move. The LNSMILP does not allow solutions violating the number of workers on vehicles, but it uses the adjusted objective function (2.32) with a big M as c_i^{job} to allow for jobs with too few workers. We introduce the new binary decision variable h_i representing $I_i^{job}(x)$. It is equal to 1 if a job i has too few workers and 0 otherwise. This leads us to rewrite constraint (2.5) as

$$\sum_{m \in \mathcal{M}_i} y_{i,m} \cdot m \leq \sum_{w \in \mathcal{W}: q_i^i \leq q_w^w} b_{w,i} + M \cdot h_i \tag{2.33}$$

To ensure a short computation time for the LNSMILP, we fix start times t_i, modes $y_{i,m}$, transports $tr_{i,j,c}$, workers' and vehicles' routing variables $x_{w,i,j,c}$ and $v_{k,i,j,c}$ as well as workers' job assignment variables $b_{w,i}$ using in the following way. If there is no destroyed job on a route, we fix all six sets of mentioned variables. Otherwise, we do not set any variable, but know that for every job i on the route of a worker or a vehicle, the sum of all routing variables to any job j equals 1, so that we can set $\sum_{j \in Succ_i} \sum_{c \in \mathcal{C}} x_{w,i,j,c} = 1$.

Once a feasible solution is found, the destroy percentage is kept constant and we perform as many destroy and repair iterations as possible within a given time δ_{LNS} using the feasible solution from the last iteration as a starting point. This approach allows for reaching even more distant areas in the solution space and for diversification. After the LNS has been performed, we set the GLS penalty terms to zero and clear the tabu list to allow for an unconstrained LS in the new solution space.

2.6 Computational study

In this section, we report computational results for the MILP and the hybrid metaheuristic. We investigate the MILP with its various categories, examine the overall performance of the hybrid procedure as well as its components and compare the results of the hybrid metaheuristic with the current planning procedure of the ground handling company.

All experiments are conducted on a Windows 7 platform with a 3.4 GHz CPU and 12 GB RAM. To solve the MILP, we employ the off-the-shelf solver IBM CPLEX 12.6. The hybrid procedure and the MILP are implemented in Java.

In Section 2.6.1 we describe the instances generated from real-world data from Munich International Airport. Section 2.6.2 presents the parameters of the hybrid procedure. Both parameters and instances are used throughout the computational study, which consists of four parts: a MILP category study (Section 2.6.3); a comparison of the MILP with the hybrid procedure (Section 2.6.4); an assessment of the performance of the individual components of the hybrid metaheuristic (Section 2.6.5); and finally a comparison of the hybrid procedure with the current planning procedure of the ground handling company (Section 2.6.6).

2.6.1 Instances

For the computational study, we use real-world data from Munich International Airport from 2012 for a full day of operation encompassing 745 aircraft arriving and leaving during the course of the day. We cluster aircraft into four types small, medium, large and very large because aircraft types impact the number of workers required and the job duration. The associated weights per type and an example aircraft can be seen in Table 2.5.

In addition, the real-world data features more than 100 parking positions for aircraft, 80 vehicles with a capacity of five workers and a workforce of 80 workers including their skill levels that work at the same time in peak hours. The set of workers is tailored to unloading and loading of baggage, which is why we focus on the two jobs. Depending on the aircraft type, the number

Table 2.5: Considered aircraft types

Aircraft type	Example	Weight w
Small	A320	1.0
Medium	A330	1.1
Large	B777	1.2
Very large	A380	1.3

of workers per job ranges between two and five and the processing time of the job between 15 and 35 minutes. The time windows for jobs are derived from airport-internal guidelines. It should be noted that other jobs can easily be added to the model and a more detailed modelling of jobs can easily be achieved by dividing a job into several sub-jobs.

Test instances are generated from the real-world data set by randomly selecting a point in time (e.g., 9.a.m.) and randomly choosing a given number of aircraft that arrive at the airport during a predefined time interval (e.g., 45 minutes). The 10 instance sets in Table 2.6 contain between 2 and 40 aircraft and feature 10 instances each. The largest instance set is set 10, with 40 aircraft, and represents the maximum number of flights at peak times during a typical rolling planning horizon framework of 30 minutes at Munich International Airport.

Table 2.6: Overview of the 10 test instance sets

Instance set	**No. flights**	**No. jobs**	**No. workers**	**No. vehicles**
1	2	4	3	2
2	3	6	4	3
3	4	8	6	4
4	5	10	7	5
5	6	12	9	6
6	8	16	12	8
7	10	20	15	10
8	20	40	30	20
9	30	60	45	30
10	40	80	60	40

2.6.2 Hybrid procedure parameters

Based on results from pre-tests, we set the maximum look-ahead parameter of the construction heuristic to $k^{max} = 3$, the look-ahead parameter to $k = 1$, the tabu list size to $l_{TS} = 7$, the number of transport swaps per iteration to $n_S^{it} = 500$ and the number of subsequent transport swaps per solution to $n_S^{sol} = 2$, as updating the jobs' start time and mode is computationally expensive. The runtime limit of the LNS is set to $\delta_{LNS} = 15s$, the penalty weight of the GLS equals $\lambda = 0.4$. The hybrid procedure employs transport categories $0 - 3$ unless a comparison with the MILP requires fewer categories. The runtime of the hybrid metaheuristic is limited to $\delta_{hybrid} = 90$ seconds to be suitable for a real-world rolling horizon planning.

2.6.3 MILP category study

The goal of the MILP evaluation is to examine the maximum problem size that can be solved with different numbers of eligible categories, to investigate which categories are employed most often in optimal solutions and to observe how computational time and objective function value change as the number of possible categories changes.

In Table 2.7, we report the average runtime (t), the objective function value (obj) or the gap of the objective function value in comparison with that of the MILP with category 0 (obj vs 0) for the MILP run with categories 0, $0 - 1$, $0 - 3$, and all categories $0 - 8$. We indicate instance sets where we could not find an optimal or even a feasible solution with "$-$". In addition, Table 2.8 shows the average frequency of the categories when using all categories. We run each instance with a MILP runtime limit of $\delta_{MILP} = 30$ minutes.

As Table 2.7 shows, we can solve instances with up to 6 jobs with all categories, up to 8 jobs with 4 categories, instances with up to 12 jobs using two categories and up to 20 jobs with one category only. Reducing the number of categories reduces the runtime and permits us to solve larger instances. For example, we can solve instances with 8 jobs using categories $0 - 3$. Reducing the number of categories has only a minor effect on the objective function value. For example in instance set 2 with 6 jobs, allowing only transports of category

0, yields an average decrease of the objective function value of 0.53% in contrast to allowing transports of all categories $0-8$. We can observe that categories $0, 1, 2$ and 5 are most frequently chosen when all categories are available in instance sets 1 and 2 (see Table 2.8), which explains why solutions using only a subsets of categories tend to have a small gap compared to the solution with all categories. We conclude that the concept of transport categories can be an efficient way to model the movement synchronization aspect in the VRPWVS by providing high quality heuristic solutions when reducing the number of possible categories.

Table 2.7: Results for MILP category testing

Set (jobs)	**Category** 0		**Categories** $0,1$		**Categories** $0-3$		**Categories** $0-8$	
	t	obj	t	obj vs 0	t	obj vs 0	t	obj vs 0
	s	*abs*	*s*	%	*s*	%	*s*	%
1(4)	0.01	27.20	0.04	0.00	0.41	0.00	1.23	0.00
2(6)	0.04	32.26	0.22	0.00	8.54	0.48	361.98	0.53
3(8)	0.24	24.10	1.07	1.43	344.31	2.14	-	-
4(10)	0.40	39.38	48.67	0.47	-	-	-	-
5(12)	1.05	41.30	257.89	1.31	-	-	-	-
6(16)	10.68	31.20	-	-	-	-	-	-
7(20)	46.73	48.64	-	-	-	-	-	-

Table 2.8: Results for MILP category testing II

Category	0	1	2	3	4	5	6	7	8
Frequency	80.17%	4.31%	4.31%	0.00%	0.00%	6.90%	0%	0.86%	3.45%

2.6.4 Comparison of MILP with the hybrid procedure

In this chapter, we compare the hybrid procedure with the MILP to examine the hybrid procedure's performance.

In Table 2.9, we report the number of categories used in the MILP and the hybrid procedure (categories used), the absolute objective (Obj) of the MILP and the hybrid procedure (HY) and the gap of the objective function value of the MILP compared to that of the hybrid procedure (HY vs MILP).

On the whole, the hybrid procedure achieves an average gap of 4.43% compared to the MILP with the same number of categories. We can observe that the variance of the average gap across instance sets is low.

Table 2.9: Results for the hybrid metaheuristic vs. MILP

Set (jobs)	**Used categories**	**MILP**	**HY**	**HY vs MILP**
		Obj	Obj	Gap
		abs	*abs*	%
1(4)	$0,1$	27.20	27.99	2.91
2(6)	$0,1$	32.26	33.42	3.59
3(8)	$0,1$	23.90	24.76	3.61
4(10)	$0,1$	39.18	41.25	5.29
5(12)	$0,1$	40.32	42.60	5.63
6(16)	0	31.20	32.74	4.93
7(20)	0	48.64	51.10	5.07
Average	-	**34.27**	**34.92**	**4.43**

2.6.5 Performance study of the hybrid procedure components

We examine all instance sets to analyse the improvements achieved by each of the three components of the hybrid procedure. Table 2.10 shows the average improvement that the hybrid procedure achieves compared to the construction heuristic (HY vs CH). We report total improvements (TOTAL), and those that are caused by the transport swap (SWAP), worker shift (SHIFT) and the large neighbourhood search (LNS). In addition, we show the percentage of generated solutions (Generated solutions) for all three moves.

As reported in Section 2.5.2, the hybrid procedure first uses transport swaps (SWAP) to improve a solution, worker shifts (SHIFT) if the generated solution is infeasible with regards to its mode, and the large neighborhood search (LNS) if the two previous moves have not generated sufficient improvement. The overall improvements are smallest for instance sets 1 and 2 because for these instance sets the optimality gap of the hybrid procedure compared is smallest (see Table 2.7). In these instance sets, the percentage of solutions from the LNS is highest which demonstrates that the local search moves SWAP and SHIFT cannot find

Table 2.10: Results for the components of the hybrid metaheuristic in terms of improvements

Set (jobs)	**HY vs CH**				**Generated solutions**		
	Total %	SWAP %	SHIFT %	LNS %	SWAP %	SHIFT %	LNS %
1(4)	0.63	0.00	0.00	0.63	4.10	0.81	95.09
2(6)	1.48	0.00	1.48	0.00	2.83	0.26	96.61
3(8)	17.63	10.53	0.00	7.10	24.93	0.75	74.32
4(10)	18.62	11.15	2.26	5.21	44.76	1.82	53.42
5(12)	17.73	14.88	0.73	2.12	82.70	3.43	13.87
6(16)	24.64	23.85	0.79	0.00	90.04	1.37	8.60
7(20)	11.15	10.39	0.00	0.75	95.40	2.18	2.42
8(40)	12.56	11.12	1.44	0.00	98.50	1.39	0.11
9(60)	13.98	13.98	0.00	0.00	98.66	1.33	0.02
10(80)	11.60	10.55	1.05	0.00	97.53	2.47	0.00
Average	**13.00**	**10.64**	**0.77**	**1.58**	**63.94**	**1.58**	**34.48**

sufficient improvements. As instance sets grow, the percentage of solutions generated by the SWAP move increases and that of the LNS decreases. Instance sets $3-6$ have similar relative improvements that are split up between all three moves but where the greatest improvements are generated by the SWAP and the LNS. In instance sets $7-10$, most improvements can be attributed to the SWAP, because the other two moves are almost never called as the SWAP generates sufficient improvements and solutions do not lack workers for jobs. Moreover, the relative improvement decreases with problem size because the runtime limit of 90 seconds does not permit further improvements while the average absolute improvements are more comparable across instance sets.

2.6.6 Comparison of hybrid procedure with current planning procedure of the ground handling company

In this section, we compare the results of the hybrid metaheuristic with the current greedy-based planning procedure of the ground handling company. The current planning procedure employs a human decision maker who manually plans ground handling operations according to airport-specific best practices that cannot be disclosed in this paper. To obtain results of the current planning procedure for all instances, we implemented the ground handler's best practices that determine the dispatcher decisions.

In Table 2.11, we show the average gap between the current planning procedure (CP) and the construction heuristic (CH) in column "CP vs CH" as well as the gap between the current procedure and the hybrid procedure (HY) in column "CP vs HY". In addition, we report how many minutes the current procedure started each job after its latest start (CP lateness). In the MILP and in the hybrid procedure, a start time after the latest start would result in an infeasible solution. However, in the real-world such decisions take place which is why the current planning procedure may generate results with start times after the latest start.

Table 2.11: Results for the comparison of the hybrid procedure with a human decision maker

Set (jobs)	**CP vs CH**	**CP vs HY**	**CP lateness**
	%	%	min per flight
1(4)	11.99	12.62	0.30
2(6)	43.08	44.15	2.60
3(8)	46.86	57.53	2.23
4(10)	27.09	41.40	1.22
5(12)	37.06	49.81	1.35
6(16)	43.71	57.84	0.70
7(20)	48.81	56.00	1.12
8(40)	26.98	36.69	0.32
9(60)	40.43	48.45	0.08
10(80)	60.75	64.86	0.06
Average	**38.68**	**46.93**	**0.47**

Table 2.11 shows that the hybrid procedure outperforms the current procedure on average by 46.93%. Even the construction heuristic achieves results superior to the current procedure. The gap increases with the problem size. The average lateness per aircraft caused by the current greedy procedure is 0.47 minutes, whereas the hybrid procedure does not permit lateness. Thus, given a total of 745 planes per day, employing the hybrid procedure at the airport would mean that ground handling operations would delay aircraft by 350 minutes or 5.8 hours less per day in total. Assuming that each minute of

delay costs 72 Euro, using our procedure at the airport could save 9 million Euros per year.

2.7 Conclusion

Operational ground handling planning is vital for airports, because the increase in air traffic poses a major challenge for ground handling operations and labour is the main cost driver for ground handling operations. In this paper, we propose a new problem, the VRPWVS, to model ground handling planning at airports. The VRPWVS belongs to the class of VRPMSs and is best characterized by movement synchronization en route and active load synchronization. The concept of active load synchronization extends the well-established concept of load synchronization from Drexl (2012) because it distinguishes between active load (e.g., workers, machines) that works on the job and possibly influences its duration and passive load (e.g., milk, post). There is no other model that combines multi-mode jobs, skills, active load and movement synchronization en route. To model joint worker and vehicle movement, we introduce the concept of transports, a combination of workers and vehicles that move from one job to another within a specific time window. In addition, we present a new powerful hybrid metaheuristic that combines tabu search, guided local search and large neighborhood search to solve the VRPWVS.

Based on real-world data from Munich International Airport, we conduct computational experiments to examine the performance of the MILP and the hybrid procedure. We demonstrate that using few transport categories in the MILP allows for solving medium-sized instances with a small optimality gap. In addition, we show that the hybrid procedure performs well compared to the MILP and that improvements of 12.86% on average are mostly generated by the transport swap move given a runtime of 90 seconds. In addition, we compare the current planning procedure from the ground handling company with our hybrid procedure and find that using our procedure at the airport could reduce the lateness of all aircraft by 5.8 hours per day and save almost 9 million Euros per year.

Future work should focus on developing an exact solution procedure for the VRPWVS. In addition, the size and composition of a minimum ground handling workforce, which we currently take as an input for the VRPWVS, could be determined.

Chapter 3

Column Generation for Vehicle Routing Problems with Multiple Synchronization Constraints

3.1 Introduction

Many industries require workers, trucks, trailers or mini-vans to jointly complete tasks and jointly move between locations. Prominent examples for such industries include logistics and transportation, medical, maintenance or airport ground handling services. Despite its importance in the real world, the synchronization of various units is very rare in the vehicle routing problem (VRP) literature (see Lahyani et al. (2015) and Drexl (2012)) especially when it comes to movement and load synchronization.

In this paper, we propose a branch-and-price procedure for vehicle routing problems with synchronization constraints, namely we focus on column generation for the abstract vehicle routing problem with worker and vehicle synchronization (AVRPWVS). The AVRPWVS deals with workers and vehicles in an airport ground handling setting and aims at minimizing job delays as well as worker and vehicle routing costs. At an airport, ground handling operations encompass jobs for airplanes stationed at parking positions, e.g. unloading bags, cleaning or refueling the aircraft. Jobs need to be conducted in a certain sequence, e.g. unloading of inbound and transfer baggage needs to be finished before outbound baggage can be loaded. Furthermore, to ensure punctual flight operations, jobs have to be finished within time windows and can only be started once all required workers have arrived at the plane's parking position. Workers use vehicles to drive from one parking position to another due to large distances and security regulations at airports. Therefore, workers need to synchronize with vehicles in time and space. The AVRPWVS is a special case of the vehicle routing problem with worker and vehicle synchronization (VRPWVS) presented in Fink et al. (2016b) and is an archetype of the class of vehicle routing problems with multiple synchronization constraints (VRPMS) meaning that the AVRPWVS covers all five synchronization types introduced by Drexl (2012). Another example for an archetype of VRPMSs is the Vehicle Routing Problem with Trailers and Transshipments (VRPTT) introduced by Drexl (2007).

This paper represents the first column generation approach for a VRPMS archetype. We make two major contributions: First, we present two multi-commodity flow formulations based on non-trivial network representations with time discretization that efficiently model all five synchronization constraints. Second, we present a novel branch-and-price heuristic (BAPH) that combines branch-and-price with heuristically fixing sums of specially modified columns. Moreover, the BAPH can easily be extended to an exact branch-and-price procedure.

The paper is structured as follows. In Section 3.2, we outline the AVRPWVS. We provide an overview of related literature in Section 3.3, and propose two multi-commodity flow formulations of the AVRPWVS in Sections 3.4 and 3.5. We describe the BAPH in Section 3.6 and present computational results (Section 3.7). Finally, we summarize our work and propose areas for future research in Section 3.8.

3.2 Problem description

In this paper, we chose the AVRPWVS as an example for a VRPMS archetype, but it should be noted that our research can also be a starting point for other branch-and-price approaches for VRPMSs. In the following, we outline the AVRPWVS.

At airports, all aircraft are associated with a set of ground handling jobs such as refueling or baggage loading. Each plane can only depart when all ground handling jobs have been completed. The number of jobs and their processing times depend on the aircraft type, airline, destination and number of passengers.

Each job requires a certain minimum number of workers to be started. To shorten a job's processing time, additional workers can be assigned to the job. We assume that a job can only start when all assigned workers have arrived at the respective aircraft's parking position. Processing a job cannot be interrupted, nor can the number of assigned workers be altered during the job's processing. Each worker's availability is determined by the worker's shift. Additionally, each worker is located at a depot at the beginning and at the end of the shift.

Workers use vehicles to move from one location, such as a gate or depot, to another location. Hence, both workers and vehicles are passive units on their own. They can only move when they team up. This movement synchronization is not required if the distance between two job locations is zero, i.e. the jobs are located at the same parking position. The number of workers that can be seated on a vehicle is limited by the vehicle capacity and it should be noted that every worker can either drive or be a passenger on any vehicle. Once a vehicle arrives at a plane's parking position, workers either embark, disembark or remain seated on the vehicle. A vehicle can only move if at least one worker at the same position as the vehicle uses the vehicle. At the beginning of the day, all vehicles are located at the depot.

The goal of operational ground handling planning is to minimize the delay of jobs such that all aircraft can operate on time. While this is the major goal, minimizing the travelling of workers and vehicles as well as worker's waiting time on the airport apron also play a role.

3.3 Literature review

The AVRPWVS is a special case of the VRPWVS because it excludes worker qualifications and different vehicle capacities. Moreover, the AVRPWVS belongs to the class of NP-hard problems as it generalizes the NP-hard vehicle routing problem with time windows (VRPTW) (see Lenstra and Rinnooy Kan (1981)). The AVRPWVS shares the major properties with the VRPWVS: It belongs to the class of vehicle routing problems with multiple synchronization constraints (VRPMSs, see Drexl (2012)), is characterized best by its active load synchronization and movement synchronization en route and covers all five synchronization constraints which marks it as a VRPMS archetype. To the best of our knowledge, this is the first paper presenting a branch-and-price procedure for a VRPMS archetype. For a general introduction to branch-and-price, see Barnhart et al. (1998) or Desrosiers and Lübbecke (2005). In Section 3.3.1, we describe the five synchronization types and how the AVRPWVS covers them. We present related column generation approaches for VRPs with movement synchronization en route and at the depot in Section 3.3.2.

3.3.1 The five synchronization types in the AVRPWVS

Drexl (2012) introduced five synchronization types to characterize VRPMSs. In the following, we briefly describe each synchronization type and how they appear in the AVRPWVS. When we refer to the term *unit*, we mean a worker or vehicle. We define a *task* (used as synonym for the term job) as a service that has to be done (e.g., loading bags off an aircraft).

Resource synchronization entails that the supply of a given resource by all units is limited. In the AVRPWVS, vehicle capacity is limited for example when workers travel on vehicles from the start depot to a parking position. When multiple units coordinate their operation in time or space, *operation synchronization* is required. For example, prior to workers starting from the depot to a parking position, they need to embark on one out of many vehicles. *Movement synchronization* prevails when two different sorts of units require each other to travel from one location to another. For instance, workers and vehicles need to synchronize in time and space when they travel from the depot to a parking position. *Task synchronization* is required when more than one unit is used and means that each job has to be started at the same time by each unit. In the AVRPWVS, each job is completed exactly once by a given number of workers. *Load synchronization* ensures that, whenever any type of load is transferred between different units, no load gets lost and that the right amount of load is delivered to a customer. As described in Fink et al. (2016b), we can extend load synchronization to differentiate between active load (e.g. workers) and passive load (e.g. baggage). The AVRPWVS covers active load synchronization because the amount of workers that switch vehicles is synchronized. Our modelling approaches, see Sections 3.4 and 3.5, consider both operation and load synchronization implicitly: embarking and disembarking of workers is not explicitly modelled, neither is the changing of vehicles by workers. However, we explicitly model resource, movement and task synchronization.

3.3.2 Related column generation approaches for VRPs with movement synchronization

In the following, we briefly outline related column generation approaches for VRPs with movement synchronization.

Tilk et al. (2015) present an exact branch-and-price procedure for the active-passive vehicle-routing problem which was first introduced by Meisel and Kopfer (2014). This problem entails pickup-and-delivery requests that require a joint operation of active vehicles and passive vehicles, which in turn can be combined at each customer location. It is the first exact procedure for a VRP with movement synchronization en route. However, the active-passive vehicle routing problem does not entail load synchronization and is thus no VRPMS archetype.

Visser (2015) analyses a simultaneous vehicle routing and crew scheduling problem arising in the distribution of goods from a depot to customers using trailers, trucks and drivers. A truck and driver combination can switch trailers only at the depot. The author investigates the performance of construction heuristics and column generation methods with exact and heuristic pricing. To obtain an integral solution, the author solves the master problem as an integer linear program once column generation terminates.

Hollis et al. (2006) consider a combined vehicle and crew routing and scheduling problem to solve the urban mail distribution problem of Australia Post. They model the problem as a pick-up-and-delivery problem (PDP) with a time horizon of 24 hours, multiple depots and a heterogeneous set of drivers that needs to be synchronized with a heterogeneous set of vehicles at several locations. Therefore, movement synchronization en route prevails. The authors solve the problem in two steps. First, they determine feasible vehicle routes by solving a path-based MIP model with a heuristic column generation approach and second, they generate worker and vehicle schedules that fit to the routes generated by the heuristic column generation approach.

In their work, Xiang et al. (2006) describe a dial-a-ride problem with movement synchronization of workers with vehicles only at depots. The authors solve the problem using a heuristic and demonstrate that it finds solutions with a small gap to a lower bound obtained by an algorithm based on column generation.

3.4 Multi-commodity flow formulation I

In the following, we introduce the notation, the time-space network representation and a first multi-commodity flow formulation (MILP I) of the AVRPWVS.

3.4.1 Notation

3.4.1.1 Jobs and time

We define a set of ground handling jobs $\mathcal{J}$ that can only start when all workers assigned to the job have arrived. Neither can a job be interrupted nor can a worker leave during the processing of the job. Each job $h \in \mathcal{J} = \{1, ..., n\}$ has several possible start times $\mathcal{T}_h^S = \{ES_h, \ldots, LS_h\}$ with ES_h and LS_h representing the earliest and the latest start of a job, respectively. The earliest arrival and latest departure times for a worker and a vehicle at job h are given as $\mathcal{T}_h^A = \{EA_h, \ldots, LD_h\}$ where $EA_h = ES_h - \tau$ and $LD_h = LS_h + p_h^{max} + \tau$.

τ denotes a non-negative integer and p_h^{max} the longest processing time of job h. Each job h is executed exactly in one mode $m \in \mathcal{M}_h$ where m stands for the number of workers assigned to job h and is related to a processing time $p_{h,m}$. In addition, we define the start depot as a dummy job 0 and denote the set of ground handling jobs and the start depot as $\mathcal{J}^0$. Likewise, the end depot is modelled as a dummy job $n+1$ and the set of ground handling jobs and the end depot is defined as $\mathcal{J}^{n+1}$. Eventually, the set of all jobs is given as $\mathcal{J}^{0,n+1}$. Precedence relationships prevail because certain jobs (e.g. loading baggage) can only be started after other jobs (e.g. unloading baggage) have been completed. We define the set of all precedence relationships as $\mathcal{E} \subset \mathcal{J} \times \mathcal{J}$.

3.4.1.2 Vertices

We propose a set of vertices $\mathcal{V}$, where each vertex $i(h, t, type) \in \mathcal{V}$ is uniquely defined by job $h \in \mathcal{J}$, a discrete time $t \in \mathcal{T}_h^A$ and a type indicating if the vertex is an arrival or departure vertex. We define $\mathcal{V}_h^{arr}$ and $\mathcal{V}_h^{dep}$ as the sets of arrival or departure vertices of job h. The start depot job 0 features several departure vertices and the end depot several arrival vertices. Similar to jobs, we denote the set of all vertices including start and end depot vertices as $\mathcal{V}^{0,n+1}$. Throughout the paper, we use indexes i and j to denote vertices, and indexes g and h for jobs.

3.4.1.3 Arcs

An arc $(i, j) \in \mathcal{A}$ connects two vertices i and j, and we do not include any arc from a vertex to itself. We further cluster the set of all arcs $\mathcal{A}$ into four arc types: processing, transitioning, waiting and travel arcs denoted by the sets $\mathcal{A}^{proc}$, $\mathcal{A}^{trans}$, $\mathcal{A}^{wait}$ and $\mathcal{A}^{travel}$, respectively. Workers can either work on a job, which means they must be routed across a processing arc, or not work on a job, which means that they are routed across a transition arc to get from an arrival to a departure type vertex in the same time and location. Workers may also wait at a parking position using waiting arcs or travel to other job locations using travel arcs. We name waiting arcs between arrival vertices "arrival waiting arcs" and waiting arcs between departure vertices "departure waiting arcs". To reduce symmetry, vehicles do not use the set of departure waiting arcs but the set of arrival waiting arcs which we denote as $\mathcal{A}^{k,wait}$. In addition, vehicles do not process jobs and therefore cannot be routed across a processing arc. The resulting set of all vehicle arcs is denoted as $\mathcal{A}^k$ which excludes both processing and departure waiting arcs. The set of travel arcs with a positive travel duration is denoted as $\mathcal{A}^{travel,>}$ and that of travel arcs with a travel duration of zero as $\mathcal{A}^{travel,0}$. In addition, we define the set of all travel arcs pointing to vertex i as $\mathcal{A}_i^{travel,in}$ and all travel arcs pointing from i as $\mathcal{A}_i^{travel,out}$. Finally, we define the set of all travel arcs pointing from the start depot as $\mathcal{A}^{travel,start}$ and all travel arcs pointing to the end depot as $\mathcal{A}^{travel,end}$. Table 3.1 shows an overview of all used sets of arcs.

Table 3.1: Arcs

Arc set	**Description**
$\mathcal{A}$	Set of all arcs
$\mathcal{A}^{proc}, \mathcal{A}^{trans}, \mathcal{A}^{wait}, \mathcal{A}^{travel}$	Set of processing, transition, wait or travel arcs
$\mathcal{A}^{travel,>}$	Set of travel arcs with a travel duration greater than 0
$\mathcal{A}^{travel,0}$	Set of travel arcs with a travel duration equal to 0
$\mathcal{A}_i^{travel,in}$	Set of travel arcs pointing to vertex i
$\mathcal{A}_i^{travel,out}$	Set of travel arcs pointing from vertex i
$\mathcal{A}_i^{travel,start}$	Set of travel arcs pointing from the start depot 0
$\mathcal{A}_i^{travel,end}$	Set of travel arcs pointing to the end depot $n+1$
$\mathcal{A}^k$	Set of all arcs for vehicles (excluding processing and departure waiting arcs)
$\mathcal{A}^{k,wait}$	Set of all waiting arcs for vehicles (i.e. arrival waiting arcs)

Each arc has a weight representing time. The weight of a processing arc (i,j), $p_{i,j}^{arc}$, equals the processing time $p_{h,m}^{job}$ for a job h and a mode m. For a transition arc, the weight is 0 and for a waiting arc, the weight is 1 indicating the period of time between two points in time. Finally for a travel arc, the weight equals the travel duration $d_{i,j}$.

3.4.1.4 Workers and vehicles

We define the set of ground handling workers as $\mathcal{W} = \{1, ..., W\}$ and the set of vehicles as $\mathcal{K} = \{1, ..., K\}$. We assume that the vehicles' capacity, i.e. the number of workers that can be transported at a time, is equal for each vehicle and given as *cap*.

3.4.1.5 Decision variables

We use the following decision variables:

- Worker routing: $x^{\mathrm{w}}_{w,i,j}$ is 1, if worker w goes from vertex i to j
- Vehicle routing: $x^{\mathrm{k}}_{k,i,j}$ is 1, if vehicle k goes from vertex i to j
- Job start and mode: $y_{h,m,t}$ is 1, if job h starts with m workers at time t

3.4.2 Network representation

We define the worker network $\mathcal{G}^{\mathcal{W}} = (\mathcal{V}, \mathcal{A})$ based on the set of all vertices $\mathcal{V}$ and the set of all arcs $\mathcal{A}$ (see Figure 3.1). The vehicle network is similar to the worker network but features the set of arcs $\mathcal{A}^k$. The resulting vehicle network $\mathcal{G}^{\mathcal{K}} = (\mathcal{V}, \mathcal{A}^k)$ can be seen in Figure 3.2. An example for an excluded departure waiting arc in the vehicle network is the lack of a connection between the departure vertices (g, t_1) and (g, t_2), and an example for an excluded processing arc is the lack of a link between arrival vertex (g, t_1) and departure vertex (g, t_2).

3.4.3 Model

Before we present the model, let us define the parameters for the objective function. Remember that the main goal of operational ground handling planning is to minimize the delay of jobs, but we also want to minimize workers' and vehicles' route costs. Consequently, we define the objective function weight α for the delay of jobs, β for worker and γ for vehicle routes. We attain the parameter σ_h to job h. In our case, σ_h grows linearly with the job's aircraft size because the larger an aircraft, the more passengers are likely to suffer from job delays. The cost of a tour for worker w is defined as

$$C^{\mathrm{w}}_w = \sum_{(i,j)\in\mathcal{A}^{travel}} d_{i,j} \cdot x^{\mathrm{w}}_{w,i,j} + \sum_{(i,j)\in\mathcal{A}^{proc}} p^{arc}_{i,j} \cdot x^{\mathrm{w}}_{w,i,j} + \sum_{(i,j)\in\mathcal{A}^{wait}} 1 \cdot x^{\mathrm{w}}_{w,i,j} \tag{3.1}$$

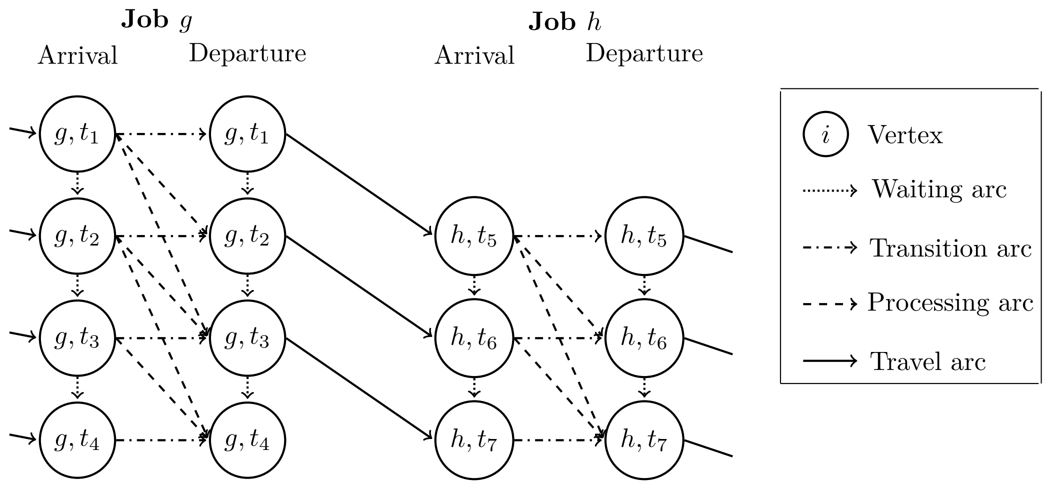

Figure 3.1: Part of a worker network

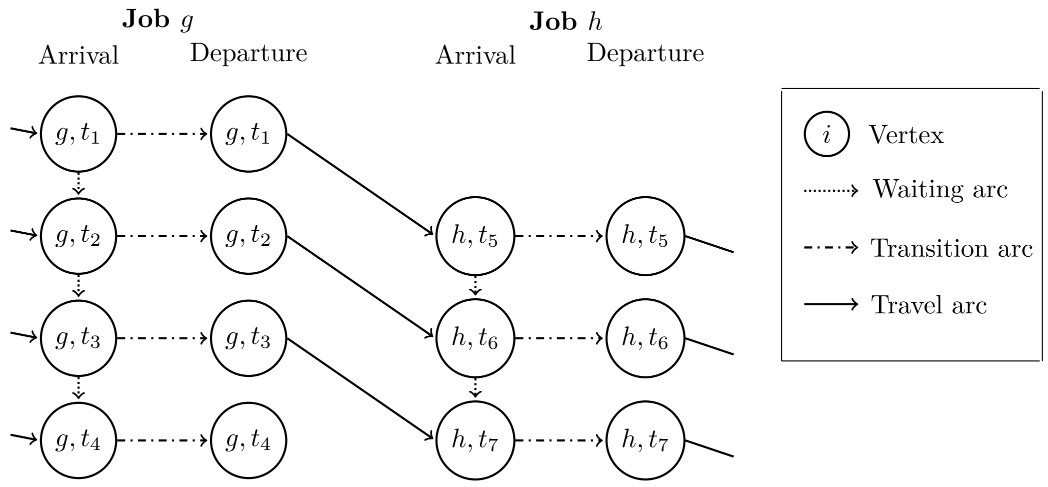

Figure 3.2: Part of a vehicle network

including the cost for traveling, processing and waiting. The cost of a tour for vehicle k is given as

$$C_k^{\mathrm{k}} = \sum_{(i,j)\in\mathcal{A}^{travel}} d_{i,j} \cdot x_{k,i,j}^{\mathrm{k}}. \tag{3.2}$$

including only the cost for travelling. For the vehicle route, we explicitly do not minimize the waiting time, as a high vehicle waiting time can result in less vehicle movements. In the following, we outline the MILP I:

$$\text{Minimize } \alpha \cdot \sum_{h \in \mathcal{J}} \sigma_h \cdot \Big(\sum_{m \in \mathcal{M}_h} \sum_{t \in \mathcal{T}_h^S} (t + p_{h,m}^{job}) \cdot y_{h,m,t} - ES_h \Big) +$$

$$\beta \cdot \sum_{w \in \mathcal{W}} C_w^{\mathrm{w}} + \gamma \cdot \sum_{k \in \mathcal{K}} C_k^{\mathrm{k}} \tag{3.3}$$

subject to

Job constraints:

$$\sum_{m \in \mathcal{M}_h} \sum_{t \in \mathcal{T}_h^S} y_{h,m,t} = 1 \qquad \forall\ h \in \mathcal{J} \tag{3.4}$$

$$\sum_{w \in \mathcal{W}} x^{\mathrm{w}}_{w,i(h,t,arr),j(h,t+p_{h,m}^{job},dep)} = m \cdot y_{h,m,t} \quad \forall\ h \in \mathcal{J},\ m \in \mathcal{M}_h, t \in \mathcal{T}_h^S \tag{3.5}$$

$$\sum_{m \in \mathcal{M}_g} \sum_{t \in \mathcal{T}_g^S} (t + p_{g,m}^{job}) \cdot y_{g,m,t} \leq$$

$$\sum_{m \in \mathcal{M}_h} \sum_{t \in \mathcal{T}_h^S} t \cdot y_{h,m,t} \qquad \forall\ (g,h) \in \mathcal{E} \tag{3.6}$$

Worker constraints:

$$\sum_{(i,j) \in \mathcal{A}^{travel,start}} x^{\mathrm{w}}_{w,i,j} = \sum_{(i,j) \in \mathcal{A}^{travel,end}} x^{\mathrm{w}}_{w,i,j} = 1 \qquad \forall\ w \in \mathcal{W} \tag{3.7}$$

$$\sum_{(j,i) \in \mathcal{A}_i^{in}} x^{\mathrm{w}}_{w,j,i} = \sum_{(i,j) \in \mathcal{A}_i^{out}} x^{\mathrm{w}}_{w,i,j} \qquad \forall\ w \in \mathcal{W}, i \in \mathcal{V} \tag{3.8}$$

Vehicle constraints:

$$\sum_{(i,j) \in \mathcal{A}^{travel,start}} x^{\mathrm{k}}_{k,i,j} = \sum_{(i,j) \in \mathcal{A}^{travel,end}} x^{\mathrm{k}}_{k,i,j} \leq 1 \qquad \forall\ k \in \mathcal{K} \tag{3.9}$$

$$\sum_{(j,i) \in \mathcal{A}_i^{in,k}} x^{\mathrm{k}}_{k,j,i} = \sum_{(i,j) \in A_i^{out,k}} x^{\mathrm{k}}_{k,i,j} \qquad \forall\ k \in \mathcal{K}, i \in \mathcal{V} \tag{3.10}$$

Movement synchronization constraints:

$$\sum_{k \in \mathcal{K}} x^{\mathrm{k}}_{k,i,j} \leq \sum_{w \in \mathcal{W}} x^{\mathrm{w}}_{w,i,j} \qquad \forall\ (i,j) \in \mathcal{A}^{travel,>} \tag{3.11}$$

$$\sum_{w \in \mathcal{W}} x^{\mathrm{w}}_{w,i,j} \leq \sum_{k \in \mathcal{K}} cap \cdot x^{\mathrm{k}}_{k,i,j} \qquad \forall\ (i,j) \in \mathcal{A}^{travel,>} \tag{3.12}$$

Variable declarations:

$$x^{\mathrm{w}}_{w,i,j} \in \{0,1\} \qquad \forall\ w \in \mathcal{W}, (i,j) \in \mathcal{A} \tag{3.13}$$

$$x^{\mathrm{k}}_{k,i,j} \in \{0,1\} \qquad \forall\ k \in \mathcal{K}, (i,j) \in \mathcal{A}^{k} \tag{3.14}$$

$$y_{h,m,t} \in \{0,1\} \qquad \forall\ h \in \mathcal{J}^{n+1}, m \in \mathcal{M}_h, t \in \mathcal{T}^{S}_{h} \tag{3.15}$$

The objective of the model (3.3) is to minimize the weighted delay caused by a deviation of a job's end time from its earliest start plus the costs of the workers and vehicles, each part weighted with its respective weight. Note that in the objective, the variables C^{w}_{w} and C^{k}_{k} are replaced by the Equations (3.1) and (3.2), respectively. Constraints can be clustered into job (3.4) - (3.6), worker (3.7) - (3.8), vehicle (3.9) - (3.10) and movement synchronization constraints (3.11) - (3.12) as well as variable declarations (3.13) - (3.15).

Job constraints (3.4) - (3.6): Each job is started once in one mode (3.4). Constraint (3.5) ensures that all workers use the same processing arc and connects the worker routing with the job variable (3.5). Constraint (3.6) represents the precedence relationship between pairs of jobs. To tighten the formulation, we henceforth replace constraint (3.6) with the precedence constraint proposed by Christofides et al. (1987):

$$y_{g,m,t} + \sum_{u \in \mathcal{T}^{S}_{h} : u < t + p^{job}_{g,m}} y_{h,m,u} \leq 1 \quad \forall\ (g,h) \in \mathcal{E}, t \in \mathcal{T}^{S}_{g}, m \in \mathcal{M}_g \tag{3.16}$$

Worker constraints (3.7) - (3.8): Constraint (3.7) enforces that each worker leaves and returns to the depot. Flow conservation of workers at every vertex other than the depot vertices is expressed by constraint (3.8).

Vehicle constraints (3.9) - (3.10): Constraint (3.9) enables vehicles to either leave or stay at the depot. If vehicles leave the depot, they also need to return. Constraint (3.10) represents the vehicle flow conservation constraint (see Ahuja et al. (1993)).

Movement synchronization constraints (3.11) - (3.12): For each travel arc that workers and vehicles share, the number of workers is at least as high as the number of vehicles to guarantee that there is always at least one driver per vehicle (3.11) and the number of workers is lower or equal to the capacity of all vehicles on the same arc (3.12). Movement synchronization is only needed when the travel time on a travel arc is greater than zero.

3.5 Multi-commodity flow formulation II

In the following, we present an alternative model for the AVRPWVS denoted as MILP II. It is inspired by Haase et al. (2001) and features worker routes but no explicit vehicle routes. In fact, the flow of vehicles is covered by both an extended worker network including vehicle travel (outlined below) and by additional vehicle flow variables capturing vehicles' waiting and travel of zero distance at the same parking position.

3.5.1 Notation

We now present additional notation necessary for the MILP II.

3.5.1.1 Additional arcs

To capture vehicle travel between locations with a travel duration greater than zero, we distinguish between worker travel arcs $\mathcal{A}^{travel,w}$ and combined worker and vehicle travel arcs $\mathcal{A}^{travel,w+v}$. Consequently, each travel between vertex i and j now features both a worker travel arc and a parallel worker and vehicle travel arc. We denote the set of all travel arcs as $\mathcal{A}^{travel} = \mathcal{A}^{travel,w+v} \cup \mathcal{A}^{travel,w}$ and the set of all travel arcs with costs greater than zero as $\mathcal{A}^{travel,>}$. Figure 3.3 illustrates the new set of worker and vehicle arcs in the so-called extended worker network. We denote the subset of worker and vehicle travel arcs that are parallel to the worker travel arc (i,j) as $\mathcal{A}^{travel,w+v}_{(i,j)}$.

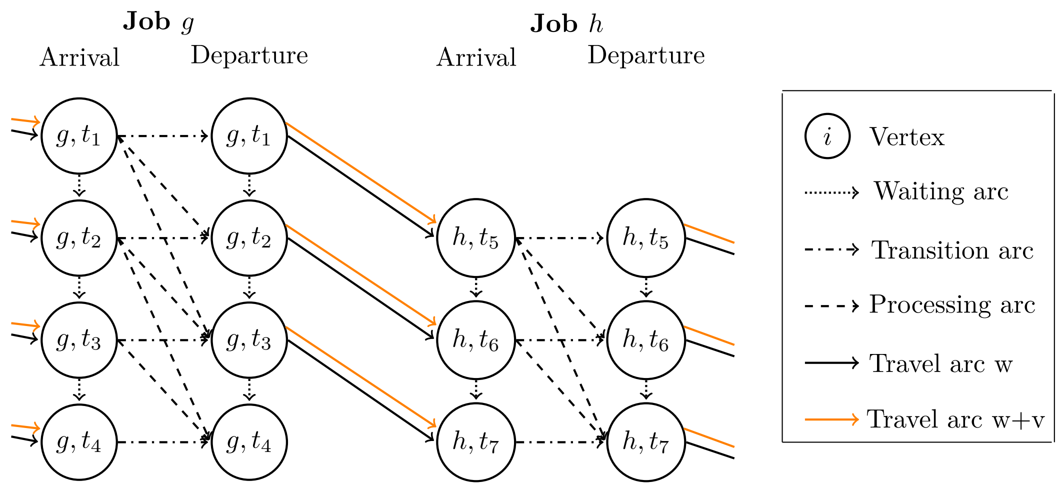

Figure 3.3: Part of an extended worker network

3.5.1.2 Additional decision variables

The travel of vehicles between locations with travel duration greater than zero is captured using the new set of travel arcs as introduced above. The vehicle flow “at the same location” is modelled by the decision variable $v_{i,j} \in [0, K]$. $v_{i,j}$ is defined for all vehicle arcs $\mathcal{A}^{\tilde{k}} = \mathcal{A}^{k,wait} \cup \mathcal{A}^{travel,0}$, i.e. for waiting and zero travel at a location where the vertices i and j belong to a new set of vehicle vertices $\mathcal{V}^k$. We define the set of vehicle vertices $\mathcal{V}^K$ for each job $h \in \mathcal{J}$, i.e. not for the start or end depot. Moreover, each vehicle vertex i is only given by its job h and its time t, but not by an arrival or departure type.

To sum it up, we map the vehicle network as in Figure 3.2 to the extended worker network and to the new flow variable $v_{i,j}$. Consequently, we can remove the vehicle routing decision variable $x^{\mathrm{k}}_{k,i,j}$ as it is no longer needed.

3.5.2 Model

To account for the additional worker and vehicle travel arcs $\mathcal{A}^{travel,w+v}$, the cost of a route of worker w can now be defined as

$$C_w^{\text{w+v}} = \sum_{(i,j)\in\mathcal{A}^{travel}} d_{i,j} \cdot x^{\text{w}}_{w,i,j} + \sum_{(i,j)\in\mathcal{A}^{proc}} p^{arc}_{i,j} \cdot x^{w}_{w,i,j} + \sum_{(i,j)\in\mathcal{A}^{wait}} 1 \cdot x^{\text{w}}_{w,i,j} \quad (3.17)$$

We now present the MILP II:

$$\text{Minimize } \alpha \cdot \sum_{h\in\mathcal{J}} \sigma_h \cdot \Big(\sum_{m\in\mathcal{M}_h} \sum_{t\in\mathcal{T}_h^S} (t + p^{job}_{h,m}) \cdot y_{h,m,t} - ES_h\Big) + \beta \cdot \sum_{w\in\mathcal{W}} C_w^{\text{w+v}} \quad (3.18)$$

subject to

Job constraints:

$$\sum_{m\in\mathcal{M}_h} \sum_{t\in\mathcal{T}_h^S} y_{h,m,t} = 1 \qquad \forall\, h \in \mathcal{J} \quad (3.19)$$

$$\sum_{w\in\mathcal{W}} x^{\text{w}}_{w,i(h,t,arr),j(h,t+p_{h,m},dep)} = m \cdot y_{h,m,t} \qquad \forall\, h \in \mathcal{J},\ m \in \mathcal{M}_h, t \in \mathcal{T}_h^S \quad (3.20)$$

$$y_{g,m,t} + \sum_{u\in\mathcal{T}_h^S : u < t + p^{job}_{g,m}} y_{h,m,u} \le 1 \qquad \forall\, (g,h) \in \mathcal{E}, t \in \mathcal{T}_g^S, m \in \mathcal{M}_g \quad (3.21)$$

Worker constraints:

$$\sum_{(i,j)\in\mathcal{A}^{travel,start}} x^{\text{w}}_{w,i,j} = \sum_{(i,j)\in\mathcal{A}^{travel,end}} x^{\text{w}}_{w,i,j} = 1 \qquad \forall\, w \in \mathcal{W} \quad (3.22)$$

$$\sum_{(j,i)\in\mathcal{A}_i^{in}} x^{\text{w}}_{w,j,i} = \sum_{(i,j)\in\mathcal{A}_i^{out}} x^{\text{w}}_{w,i,j} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{V} \quad (3.23)$$

Vehicle flow constraints:

$$\sum_{(i,j)\in\mathcal{A}^{travel,w+v,start}} x^{\text{w}}_{w,i,j} \le K \quad (3.24)$$

$$\sum_{w \in \mathcal{W}} \sum_{(j,i) \in \mathcal{A}_{i,in}^{travel,>,w+v}} x_{w,j,i}^{\mathrm{w}} + \sum_{(j,i) \in \mathcal{A}_{i,in}^{\tilde{k}}} v_{j,i}$$

$$= \sum_{w \in \mathcal{W}} \sum_{(i,j) \in \mathcal{A}_{i,out}^{travel,>,w+v}} x_{w,i,j}^{\mathrm{w}} + \sum_{(i,j) \in \mathcal{A}_{i,out}^{\tilde{k}}} v_{i,j} \qquad \forall\ i \in \mathcal{V}^k \tag{3.25}$$

Movement synchronization constraints:

$$\sum_{w \in \mathcal{W}} x_{w,i,j}^{\mathrm{w}} \leq \sum_{w \in \mathcal{W}} \sum_{(i,j) \in A_{i,j}^{travel,>,w+v}} x_{w,i,j}^{\mathrm{w}} \cdot (cap - 1) \qquad \forall\ (i,j) \in \mathcal{A}^{travel,>,w} \tag{3.26}$$

Variable declarations:

$$x_{w,i,j}^{\mathrm{w}} \in \{0,1\} \qquad \forall\ w \in \mathcal{W}, (i,j) \in \mathcal{A} \tag{3.27}$$

$$v_{i,j} \in [0,K] \qquad \forall\ (i,j) \in \mathcal{A}^{\tilde{k}} \tag{3.28}$$

$$y_{h,m,t} \in \{0,1\} \qquad \forall\ h \in \mathcal{J}^{n+1}, m \in \mathcal{M}_h, t \in \mathcal{T}_h^S \tag{3.29}$$

The objective of the model (3.18) is similar to the objective of the MILP I (3.3) except that we remove the vehicle routes and use extended worker routes. Note that in the objective, the variable $C_w^{\mathrm{w+v}}$ is replaced by the term in Equation (3.17). Note that the objective function value of an optimal solution of the MILP II is equal to that of the MILP I because a worker route features vehicle travels in its cost.

Job and worker constraints (3.19) - (3.23): Job constraints remain unchanged and worker constraints as well except that the set of all arcs and the set of travel arcs now encompass also worker and vehicle arcs.

Vehicle flow constraints (3.24) - (3.25): The number of vehicles that leave the depot must be less or equal to K as defined in constraint (3.24). Constraint (3.25) ensures that the vehicle flow coming into each vehicle vertex equals the flow leaving the vehicle vertex.

Movement synchronization constraint (3.26): To synchronize workers and vehicles, we use constraint (3.26) to ensure there is enough vehicle capacity. Note that we can only offer a capacity of $cap - 1$ on a worker and vehicle travel arc as one capacity unit is already reserved for the worker driving the vehicle. As a consequence, we need no constraint enforcing at least one worker per vehicle, as this is given by the worker and vehicle travel arc.

Constraints (3.27) - (3.29) define the decision variables. In comparison to the MILP I, the MILP II encompasses approximately half the number of constraints, slightly fewer binary decision variables but considerably more continuous variables.

3.6 Branch-and-price

We propose a branch-and-price heuristic (BAPH) for the AVRPWVS as column generation is renown as an efficient approach for solving VRPs (see Desrosiers and Lübbecke (2005), Grønhaug et al. (2010) or Desaulniers (2010)). To perform column generation, we reformulate the MILP II as a set-partitioning master problem (MP) in Section 3.6.1 and define the pricing subproblem in Section 3.6.2. Column generation (CG) solves a linear relaxation and is an iterative approach that in each iteration considers only a subset of all feasible columns or routes in the MP, which is why it is called restricted master problem (RMP). While the RMP guides the generation of new columns, the creation of new columns is achieved by the subproblem (SP), also called the pricing problem. In each iteration, the RMP is solved with a given set of columns as a linear program (LP). Then, the dual values of the RMP are used in the SP to generate new routes. The SP either finds new columns with negative reduced costs that are thereafter added to the RMP or the procedure terminates and a valid lower bound of the minimization problem is found. To solve the SP, we use a dynamic program labelling algorithm given in Section 3.6.3. Section 3.6.4 describes several acceleration strategies we implemented to speed up the procedure. Finally, Section 3.6.5 outlines how we obtain integral solutions.

3.6.1 Set-partitioning master problem formulation

We reformulate the MILP II as a set-partitioning model using Dantzig-Wolfe decomposition (see e.g. Lübbecke and Desrosiers (2005)). Let r denote a worker route or column and $\mathcal{R}$ the set of all columns. For each route r we define the following notation:

- f_r: binary variable equal to 1, if the route r is selected
- $\delta_{i,j,r}$: binary parameter equal to 1, if the arc from i to j is used in route r
- $\delta^{w}_{i,j,r}$: binary parameter equal to 1, if the worker travel arc from i to j is used in route r
- $\delta^{w+v}_{i,j,r}$: binary parameter equal to 1, if the worker and vehicle travel arc from i to j is used in route r
- C^{r}_r: cost of route r as given in (3.17)

We can now define the MP as follows:

$$\text{Minimize } \alpha \cdot \sum_{h\in\mathcal{J}(n+1)} \sigma_h \cdot \Big(\sum_{m\in\mathcal{M}_h}\sum_{t\in\mathcal{T}^S_h}(t+p^{job}_{h,m})\cdot y_{h,m,t} - ES_h\Big) + \beta\cdot\sum_{r\in R} C^{\mathrm{r}}_r \cdot f_r \tag{3.30}$$

subject to

$$\sum_{m\in\mathcal{M}_h}\sum_{t\in\mathcal{T}^S_h} y_{h,m,t} = 1 \qquad \forall\ h\in\mathcal{J} \tag{3.31}$$

$$\sum_{r\in\mathcal{R}} \delta_{i(h,t,arr),j(h,t+p_{h,m},dep),r}\cdot f_r = m\cdot y_{h,m,t} \qquad \forall\ h\in\mathcal{J},\ m\in\mathcal{M}_h, t\in\mathcal{T}^S_h \tag{3.32}$$

$$y_{g,m,t} + \sum_{u\in\mathcal{T}^S_h : u<t+p^{job}_{g,m}} y_{h,m,u} \le 1 \qquad \forall\ (g,h)\in\mathcal{E}, t\in\mathcal{T}^S_g, m\in\mathcal{M}_g \tag{3.33}$$

$$\sum_{r\in\mathcal{R}} f_r = W \tag{3.34}$$

$$\sum_{r \in \mathcal{R}} \sum_{(i,j) \in \mathcal{A}^{travel,>,w+v,start}} \delta_{i,j,r}^{w+v} \cdot f_r \leq K \tag{3.35}$$

$$\sum_{r \in \mathcal{R}} \sum_{(j,i) \in \mathcal{A}_{i,in}^{travel,>,w+v}} \delta_{j,i,r}^{w+v} \cdot f_r + \sum_{\mathcal{A}_{i,in}^{\tilde{k}}} v_{j,i}$$

$$= \sum_{r \in \mathcal{R}} \sum_{(i,j) \in \mathcal{A}_{i,out}^{travel,>,w+v}} \delta_{i,j,r}^{w+v} \cdot f_r + \sum_{\mathcal{A}_{i,out}^{\tilde{k}}} v_{i,j} \quad \forall\ i \in \mathcal{V}^k \tag{3.36}$$

$$\sum_{r \in \mathcal{R}} \delta_{i,j,r}^{w} \cdot f_r \leq \sum_{r \in \mathcal{R}} \delta_{i,j,r}^{w+v} \cdot (cap - 1) \cdot f_r \quad \forall\ (i,j) \in \mathcal{A}^{travel,>,w} \tag{3.37}$$

$$f_r \in \{0,1\} \quad \forall\ r \in \mathcal{R} \tag{3.38}$$

$$v_{i,j} \in [0,K] \quad \forall\ (i,j) \in \mathcal{A}^{\tilde{k}} \tag{3.39}$$

$$y_{h,m,t} \in \{0,1\} \quad \forall\ h \in \mathcal{J}^{n+1}, m \in \mathcal{M}_h, t \in \mathcal{T}_h^S \tag{3.40}$$

The objective function (3.30) corresponds to that of MILP II (3.18) as it includes only extended worker routes.

Constraints (3.31) - (3.33) correspond to constraints (3.19) - (3.21) in the MILP II. Constraints (3.34) and (3.35) limit the number of workers and vehicles in the problem. The vehicle flow constraint (3.25) from MILP II is mapped to constraint (3.36). Constraint (3.37) corresponds to constraint (3.26) in MILP II and constraints (3.38) to (3.40) define the decision variables.

3.6.2 Pricing problem

To generate new columns, we solve a worker subproblem or pricing problem which corresponds to a shortest path problem with resource constraints (SPPRC) on the extended worker network. For a general introduction to SPPRCs, see Irnich and Desaulniers (2005).

The SPPRC aims at finding the route r with the lowest reduced cost, where the reduced cost of a route is defined as the sum of the reduced costs of its arcs. The dual multiplier for constraint (3.32) is given as $\pi_{h,m,t}^{(3.32)}$. Likewise, we set $\pi^{(3.34)}$, $\pi^{(3.35)}$, $\pi_i^{(3.36)}$ and $\pi_{(i,j)}^{(3.37)}$. Table 3.2 summarizes the reduced cost per arc type.

Table 3.2: Reduced cost per arc type

Arc	**Reduced cost**
$(i(h,t,arr), j(h,t+p_{h,m},dep)) \in \mathcal{A}^{proc}$	$\beta \cdot p_{i,j}$ - $\pi^{(3.32)}_{h,m,t}$
$(i,j) \in \mathcal{A}^{travel,>,W+V}$	$\beta \cdot 2 \cdot d_{i,j} + \pi^{(3.36)}_{i} - \pi^{(3.36)}_{j} + \pi^{(3.37)}_{i}$
$(i,j) \in \mathcal{A}^{travel,>,W}$	$\beta \cdot d_{i,j}$ - $\pi^{(3.37)}_{i,j}$

3.6.3 Subproblem labelling algorithm

We solve the pricing problem with a dynamic programming labelling algorithm. Such an algorithm iteratively builds new paths starting from a source to a sink. All relevant information of a path is stored in a label (see Section 3.6.3.1) for each of the path's vertices. In order not to enumerate all paths, the labelling algorithm discards paths that are not useful by using a dominance rule (see Section 3.6.3.2). The whole algorithm is described in Section 3.6.3.3.

3.6.3.1 Label definition

A labelling algorithm generates labels for each partial or complete path. In each label l, we store the current vertex, the parent label id, the cumulative reduced cost of the partial or full path and the last time a transition arc was used on the path (called "last idle time“) . The reason for storing the last idle time in a label is to avoid cycling. The extended worker network in Figure 3.3 does not permit starting a job twice for one route due to tight time windows. However, it is possible that a cycle of zero length occurs because workers may use a transition arc in combination with a zero travel arc enabling a return to the same vertex at zero cost. To avoid this, we save the last idle time and include it as a dominance criterion in our dominance rule.

3.6.3.2 Dominance rule

We store all non-dominated labels of one vertex i in a container of labels called bucket F_i. To examine the dominance of a new label at vertex i, we compare it to each label in F_i. A label l dominates a label l' if the reduced cost of label l

is equal or lower than that of l' and if the last idle time of l is equal or lower than that of l', with at least one of the two inequalities being strict.

3.6.3.3 Labelling algorithm

Having outlined both the label definition and the dominance rule, we now explain the labelling algorithm (see Algorithm 1). Before starting, the set of all vertices $\mathcal{V}$ is sorted by ascending time. As a first tie breaker, we give arrival vertices precedence over departure vertices with the same time. A second tie breaker is the job id. Moreover, the procedure initializes a start label l_0, the bucket for start vertex 0, F_0, and the set of same time vertices $\mathcal{V}^{sameTime}$.

It then iterates through each vertex $i \in \mathcal{V}$ and distinguishes between two cases: first, if the time associated with vertex i is equal to that of the last examined vertex, it scans vertex i and moves i into the set of undominated vertices with the same time $\mathcal{V}^{sameTime}$ if i is not dominated by any label in F_i and if i is not already in $\mathcal{V}^{sameTime}$. Second, if the time of the currently analysed vertex i is higher than that of the last analysed vertex, the labelling algorithm examines each vertex in set $\mathcal{V}^{sameTime}$. To scan or examine a vertex, we use the method $extendVertex()$ that is illustrated in Algorithm 2. When we use $extendVertex()$ to extend vertex i, the method generates for each arc (i, j) pointing from i a new label l_j. The method then determines if l_j is allowed to enter F_j using the dominance rule. Once all vertices i are examined, we recursively build shortest paths.

3.6.4 Acceleration Strategies

We apply several acceleration strategies to reduce the runtime of the CG both in the SP and the RMP.

3.6.4.1 Stabilization of dual vector

We use dual variable stabilization as introduced by Wentges (1997) to reduce the well-known oscillation of the dual variables passed from the master to the subproblem. It combines the latest dual vector π^{last} with the so far best dual vector π^{best} so that the duals employed in the current iteration are

Algorithm 1 Labeling algorithm

```
1: $l_0 \leftarrow (0, \emptyset, 0, 0)$ //initialize label $l_0$
2: $F_0 \leftarrow l_0$ //initialize $F_0$ with $l_0$
3: $\mathcal{V}^{sameTime} \leftarrow \emptyset$ //initialize set of same time vertices as empty
4: for $i \in \mathcal{V}$ do
5:    if sameTimeAsLastVertex(i) then
6:       $extendVertex(i)$ // Extend label i
7:    else
8:       while $\mathcal{V}^{sameTime} \neq \emptyset$ do
9:          $removeVertex(i)$ //Remove i from the set of vertices with the same
            time
10:         $extendVertex(i)$ // Extend label i
11:      end while
12:   end if
13: end for
14: $findShortestPaths(\mathcal{F})$ //use label buckets to recursively find shortest paths
```

Algorithm 2 extendVertex() method

```
1: for $l_i \in F_i$ do
2:    for $j \in SUCCESSORS_i$ do
3:       $l_j \leftarrow (j, l_i, reducedCost, travelCost, lastTimeIdle)$
4:       $checkDominance(l_j, F_j)$ //Check dominance based on reduced cost and
         last time idle and potentially add new label to bucket
5:       if $l_j \in F_j$ & $sameTimeAsLastVertex(i)$ then
6:          $\mathcal{V}^{sameTime} \leftarrow j$
7:       end if
8:    end for
9: end for
```

$\pi^{now} = \theta \cdot \pi^{best} + (1 - \theta) \cdot \pi^{last}$ where θ denotes the weight assigned to the best dual vector.

3.6.4.2 Dynamic constraints

In the RMP, the capacity constraint (3.26) is constructed for each worker and vehicle travel arc with a positive distance resulting in a very large number of constraints. However, only few travel arcs are likely to be used and therefore only few capacity constraints are binding. Thus, we use dynamic constraints as proposed by Desaulniers et al. (2002). We start with an RMP excluding constraints (3.26) and add only those constraints to the RMP that are needed when appending new columns to the RMP.

3.6.4.3 Adding multiple columns

To speed up column generation, adding multiple columns per iteration can help. However, one needs to choose the type and number of columns wisely, as many non-useful columns in the RMP may increase the runtime of the procedure by leading the duals astray. Therefore, we add several task-disjoint columns to the RMP in each iteration (see Desaulniers et al. (2002)). Two columns are task-disjoint, if they cover different jobs. For example, let column r_1 process the jobs $1, 2, 4$, column r_2 process the jobs $3, 5, 6$ and column r_3 process the jobs $1, 3, 5$. Then columns r_1 and r_2 are disjoint, but r_3 is not disjoint from r_1 and r_2. We add to the RMP at most m sets of task-disjoint columns.

3.6.5 Integer solutions

To obtain integer solutions, we propose a branch-and-price heuristic (BAPH) that combines branching on the number of vehicles and on jobs with fixing of routes. The novel aspect of the BAPH is that we do not fix single columns, which has often lead to successful heuristic procedures (see for instance Joncour et al. (2010), Frey et al. (2016)), but rather a sum of specially modified routes.

Figure 3.4 shows an overview of the BAPH. Until there is no unexplored node any more or until a given time limit is reached, the BAPH procedure iteratively runs column generation (CG) that is adjusted based on the currently

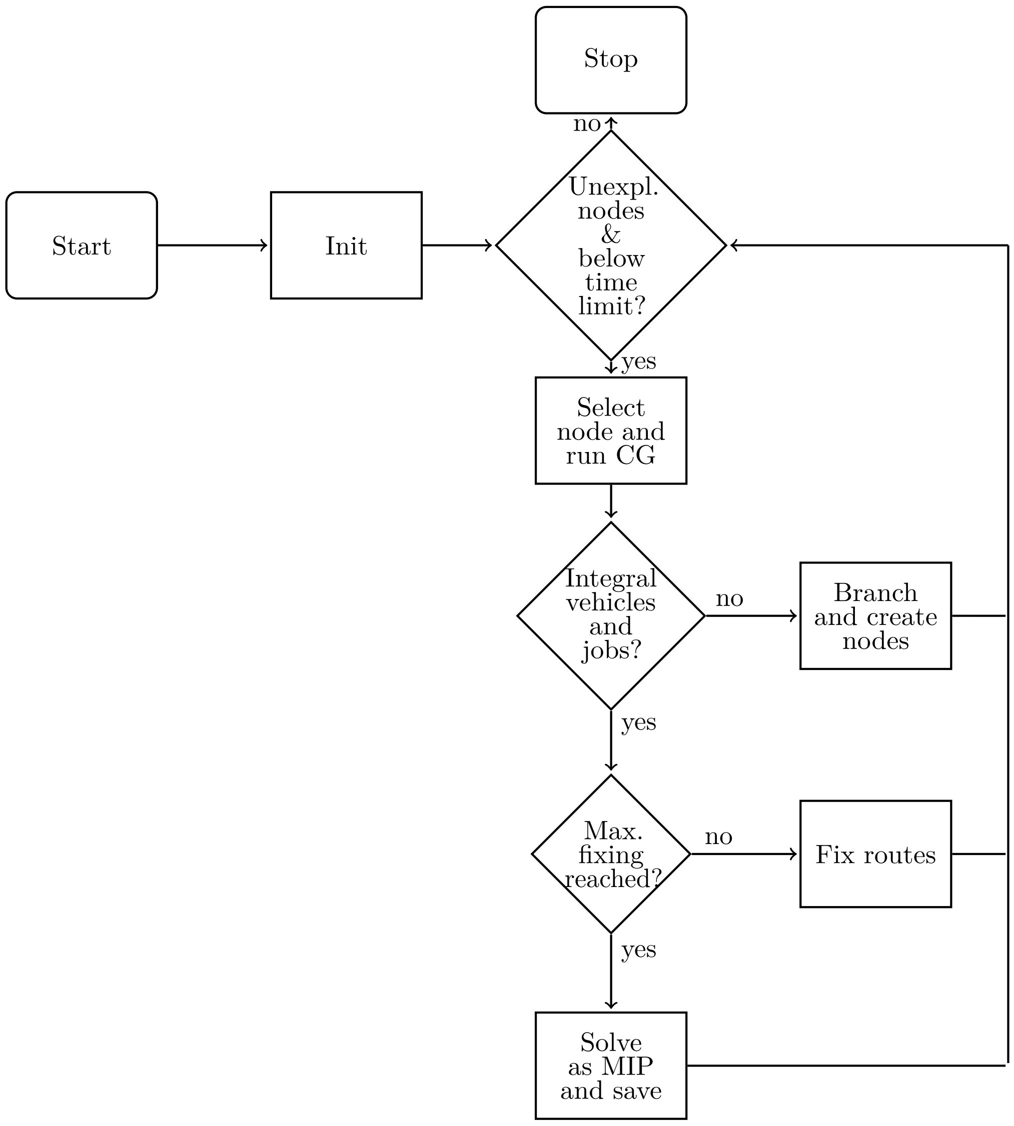

Figure 3.4: BAPH illustration

selected node. In a node, we store information of a (partial) solution such as the lower and upper bounds on the number of vehicles, forbidden processing arcs or fixed sums of routes. We employ a depth-first search. During the course of the BAPH, we first aim at an integral number of vehicles and integral job variables, which we achieve through branching (see Section 3.6.5.1) and which corresponds rather to a traditional branch-and-price approach. When the variables for the number of vehicles and for jobs are integral, we have the option to either branch on travel arcs which would result in a fully-fledged branch-and-price procedure or to fix columns. We found that fixing sums of specially modified routes (see Section 3.6.5.2) and then handing the solution to a MIP solver (see Section 3.6.5.3) and thus solving the problem heuristically yields good results at a comparability low runtime. Naturally, extending the BAPH to an exact branch-and-price procedure is straightforward as only branching on travel arcs is required. In the following, we describe the three major components of the BAPH in more detail.

3.6.5.1 Branching

We employ a branching hierarchy that first ensures an integral number of vehicles and second integral job variables. The number of employed vehicles k is an auxiliary variable not explicitly shown in the RMP, it is defined as $k = \sum\limits_{r\in\mathcal{R}} \sum\limits_{(i,j)\in\mathcal{A}^{travel,w+v,start}} \delta^{w+v}_{i,j,r} \cdot f_r$. Let $\tilde{k}$ equal the number of vehicles and $\tilde{y}_{h,m,t}$ the value of the variable for job h at mode m starting in time t. When $\tilde{k}$ is fractional, we branch on this variable by imposing $\sum\limits_{r\in\mathcal{R}} \sum\limits_{(i,j)\in\mathcal{A}^{travel,w+v,start}} \delta^{w+v}_{i,j,r} \cdot f_r \leq \lfloor \tilde{k} \rfloor$ on one branch and $\sum\limits_{r\in\mathcal{R}} \sum\limits_{(i,j)\in\mathcal{A}^{travel,w+v,start}} \delta^{w+v}_{i,j,r} \cdot f_r \geq \lceil \tilde{k} \rceil$ on the other. We enforce the lower or the upper bound by adding an additional constraint to the RMP. When $\tilde{y}_{h,m,t}$ is fractional, we impose $\sum\limits_{n\in\mathcal{M}_h:n\leq m} \sum\limits_{u\in\mathcal{T}^S_h:u\leq t} y_{h,m,t} = 0$ in one branch and $\sum\limits_{n\in\mathcal{M}_h:n>m} \sum\limits_{u\in\mathcal{T}^S_h:u>t} y_{h,m,t} = 0$ in the other. We enforce the branching by removing all columns violating the latter equations from the RMP. In addition, we eliminate the respective processing arcs from the subproblem.

3.6.5.2 Fix routes

When both the number of vehicles and jobs are integral, we start to iteratively fix routes using the following three-step process: First, we select a route using some selection strategy. Second, we create all modifications of the selected route. Third, we set the sum of all modified routes plus the selected route equal or greater than one in the RMP. We now illustrate each step in more detail: i) We select the route with a variable value closest to 0.5. ii) Given a selected route $\bar{r}$, we create its set of modified routes $\mathcal{R}^{\bar{r}}$. $\mathcal{R}^{\bar{r}}$ not only includes $\bar{r}$ itself but also all possible route modifications of $\bar{r}$. A route modification is defined as creating a copy of $\bar{r}$ denoted $\bar{r}^{'}$, selecting a travel arc of $\bar{r}^{'}$ and replacing the travel arc with its parallel travel arc. I.e. if the selected travel arc (i,j) is a worker travel arc $(i,j) \in \mathcal{A}^{travel,w}$, then we replace (i,j) with its parallel worker and vehicle travel arc $(i,j) \in \mathcal{A}^{travel,w+v}$. Creating modifications of the selected tour $\bar{r}$ has the advantage that for each worker travel arc, there exists at least one route with a parallel worker and vehicle travel arc. In other words, creating modified routes ensures movement synchronization. iii) Finally, we add a new constraint to the RMP such that $\sum\limits_{r \in \mathcal{R}^{\bar{r}}} f^r \geq 1$. We stop fixing when all routes in the RMP that are part of the basis belong to a set of fixed routes.

3.6.5.3 Solve as MIP

Given an integral number of vehicles and jobs as well as a maximum number of fixed sums of routes, we solve the RMP with all its columns as a MIP using CPLEX and save the solution in the pool of integral solutions. When the procedure terminates, the best feasible integral solution of all generated integral solutions is returned.

3.7 Computational study

In this section, we report results of our computational experiments. We investigate the performance of the two MILPs in terms of runtime, integrality gap and number of branch-and-bound nodes. Then, we compare the BAPH with the better performing MILP II solving it both for 1% and a zero optimality gap. Finally, when testing the BAPH, we examine the various implemented acceleration strategies.

All experiments are run on a Windows 7 platform with a 3.4 GHz CPU and 12 GB RAM and on a single thread. The MILP as well as the LP of the BAPH are solved with the off-the-shelf solver IBM CPLEX 12.6. The BAPH and the MILPs are implemented in the programming language Java.

3.7.1 Data instances

During the whole computational study, we employ data from Munich International Airport from 2012 for a regular day of operation entailing 745 aircraft that arrive and leave during the course of the day. It is the same dataset as used in Fink et al. (2016b). The data comprises more than 100 parking positions for aircraft, 80 vehicles with a capacity of 5 workers and a workforce of 100 workers working in parallel during peak hours. Depending on the aircraft type, the number of workers per job ranges between $2-5$ and the processing time of the job between $15-35$ minutes.

Instances are generated by randomly selecting a point in time and then choosing a given number of aircraft in a given time interval, e.g. selecting 10 aircraft in a time interval between 8 and 9 a.m. We associate each airplane with two jobs, namely unloading and loading baggage because the given workforce data is tailored to these two jobs. Other jobs could easily be added in the model. Each instance set contains 8 individual instances. Table 3.3 shows the instance sets with the number of workers and vehicles. It should be noted that we solve instances up to 32 jobs due to runtime limitations and while not removing any arcs from the network. Larger problem instances can be solved with the BAPH when removing arcs at the cost of potentially achieving worse solutions. However, this is not part of this paper but can be a subject of future research.

Table 3.3: Overview of the instance sets

Instance set	**No. jobs**	**No. workers**	**No. vehicles**
1	4	3	2
2	6	4	3
3	8	6	4
4	10	8	5
5	12	10	6
6	16	12	8
7	20	14	10
8	24	16	12
9	28	18	14
10	32	20	16

3.7.2 Algorithm settings

We set the objective function weights $\alpha = 0.8$, $\beta = 0.1$, $\gamma = 0.1$ as we foremost want to minimize the job delay. The delay weight per job is set according to the associated aircraft type. For small aircraft such as an Airbus A320 we set $\sigma_h = 1.0$, for medium aircraft such as an Airbus A330 we set $\sigma_h = 1.1$, for a large aircraft such as a Boeing $B777$ we set $\sigma_h = 1.2$ and for very large aircraft such as an Airbus A380 we set $\sigma_h = 1.3$. The runtime limit of the MILP models and of the BAPH is $3,600$ seconds. Moreover, we aim for a maximum of three integral solutions in the BAPH because during pretests we found that after three feasible integral solutions, the BAPH did not achieve significantly better solutions. Therefore, when we report the BAPH overall runtime, it is either determined by the time when the BAPH finds the third feasible integral solution, by the runtime limit or by the procedure stopping because there exists no unexplored node any more. We set the weight for the dual stabilization as $\theta = 0.5$.

3.7.3 MILP testing

In the following MILP testing we compare both flow formulations. Table 3.4 shows for both flow formulations MILP I and MILP II the average runtime in seconds to solve the MIP (Time) as well as the average number of branch-and-bound nodes used by CPLEX (No. nodes). Moreover, we present the share of instances for which the MILP II was faster (II faster) and the average integrality gap (INT).

Table 3.4: Results for MILP comparison

Jobs	**MILP I**		**MILP II**		**Comparison**	**INT**
	Time(s)	No. nodes	Time(s)	No. nodes	II faster(%)	GAP(%)
4	1.07	6.63	1.20	5.25	37.50	2.83
6	20.19	135.25	11.20	156.00	100.00	2.56
8	231.16	527.25	88.65	385.63	75.00	1.78
10	2,720.05	1,126.75	1,499.74	1,950.63	75.00	2.40
Average	743.12	448.97	400.20	624.38	71.88	2.30

The MILP II shows lower average runtimes than the MILP I for all instance sets except 4 jobs and is overall faster for 71.88% of all instances. The integrality gap is on average 2.30% which means that in particular for the larger 10 jobs instances, closing the gap via branch-and-bound requires a growing number of branch-and-bound nodes. Note that the MILP I reaches the runtime limit for 5 of all 8 instances with 10 jobs, the MILP II only for 2.

3.7.4 MILP and BAPH comparison

In this section, we compare the MILP II with the BAPH. Table 3.5 shows runtime (Time) and solution value (Obj) of the MILP II given a zero optimality gap (0% GAP), a 1% optimality gap (1% GAP), and the BAPH. In addition, we show the solution value difference of the BAPH compared to the MILP II 0% (vs 0%) and the 1% (vs 1%). We indicate instance sets for which we could not find at least one optimal solution with a "-".

Table 3.5: Results for BAPH vs MILP II testing

Jobs	0% **GAP**		1% **GAP**		**BAPH**		**Difference**	
	Time(s)	Obj	Time(s)	Obj	Time(s)	Obj	vs0%(%)	vs1%(%)
4	1.20	71.81	1.16	71.87	2.07	71.96	0.21	0.12
6	11.20	115.51	9.10	115.79	3.94	115.65	0.12	-0.12
8	88.65	165.56	60.45	165.88	9.66	165.81	0.15	-0.04
10	1,499.74	201.15	1,149.58	201.69	83.05	203.59	1.20	0.93
12	–	–	2,968.89	242.87	623.66	241.75	–	-0.47

We can observe that the solution value of the MILP II 1% is slightly worse than that of the MILP II 0%, but that the runtime of the MILP II 1% is much lower. Contrasting the BAPH and the MILP II, we can remark that the BAPH runs only for a fraction of the time of the MILP II (both the optimal and 1% version) except for the smallest instances with 4 jobs while achieving an average solution value that is very close to the optimal value: the average difference between the MILP II at 0% and the BAPH ranges between 0.12% and 1.20%. Comparing the MILP at 1% and the BAPH we can see that the BAPH finds better solutions for the instance sets with $6, 8$ and 12 jobs on average, and for 58% of all tested instances. Note that the MILP II 0% again reaches the runtime limit for 2 10-job instances and the MILP II 1% does likewise for 6 12-job instances and for 2 10-job instances.

3.7.5 BAPH testing

The goal of the BAPH testing is to demonstrate that the BAPH finds good solutions at a comparably short runtime also for larger problem instances and that the implemented acceleration strategies are efficient.

3.7.5.1 BAPH testing: runtime and solution quality

In Table 3.6, we present the LP objective function value (LP obj) obtained by running the CG without early termination. For the BAPH, we report the first found integral solution's solution value (Int obj f.) and runtime (Time f.), as well as the best integral solution's solution value (Int obj) and runtime (Time). Moreover, we report the number of feasible integral solutions found by the BAPH (Sol) and the integrality gap (GAP) between the LP and the integral solution value.

Table 3.6: Results for BAPH testing: runtime and solution quality

Jobs	LP obj	Time f.(s)	Int obj f.	Time(s)	Int obj	Sol	GAP (%)
4	69.78	0.64	72.02	2.07	71.96	2.38	3.03
6	112.55	3.05	115.76	3.94	115.65	3.00	2.67
8	162.61	5.83	165.88	9.66	165.81	3.00	1.92
10	196.33	52.76	203.70	83.05	203.59	3.00	3.57
12	234.35	52.82	241.85	623.66	241.75	3.00	3.06
16	299.92	101.20	310.56	280.97	310.07	3.00	3.27
20	372.28	436.89	387.58	806.69	387.19	3.00	3.85
24	439.61	928.11	460.14	2,037.29	460.14	2.75	4.46
28	506.69	1,664.94	529.24	3,268.64	528.77	2.13	4.17
32	580.48	2,139.50	605.09	3,600.00	605.09	1.25	4.07

We can see that the runtime of the BAPH increases with the problem size until it reaches the runtime limit of 3,600 seconds for each 32-job instance. Moreover, for the largest instance sets with 24, 28 and 32 jobs, the average number of found solutions decreases due to the runtime limit. For all other instance sets except for 4 jobs, the procedure stops at three found integral solutions. Also note that when comparing Int obj with Int obj f., we can observe that the average solution value is better than the value of the first found solution, which indicates that continuing the search for an integral solution after having found a first solution makes sense. However, the improvement in solution value comes at the cost of a large runtime increase: the runtime until three solutions are found is often more than double that of the time to find the first feasible solution. Therefore, if one wants to find good solutions quickly, running the BAPH until the first solution is a viable option. The gap between the lowest found LP solution and the best MIP solution stays in an average range of 1.92% to 4.46%. Given that the average integrality gap in Table 3.4 equals 2.30%, we can conclude that the BAPH finds solutions with values close to the optimum also for larger instances.

3.7.5.2 BAPH testing: acceleration strategies

Table 3.7 shows runtime (Time) and solution value (Obj) of the best found integral solution by the BAPH and distinguishes between several configurations: the original configuration (Original), the original configuration without stabilization (W/O stab, see Section 3.6.4.1), the original configuration without dynamic constraints (W/O dynamic, see Section 3.6.4.2) and the original configuration without adding several disjoint columns (W/O disjCols, see Section 3.6.4.3). In the latter case, we instead add in each iteration the 10 columns with the lowest reduced cost. Note that for the largest three instance sets with $24, 28$ and 32 jobs, the comparison of configurations is futile because not all instances could be solved by each configuration (indicated by a "-"). Instead, Table 3.8 presents the share of solved instances by the various configurations for the largest three instance sets.

Overall, Table 3.7 demonstrates that the original configuration performs best in terms of both average runtime and solution value for instances up to 20 jobs where all four configurations can solve all instances. Furthermore, the configuration without stabilization finds only slightly worse results in slightly greater runtime and the configuration without adding several disjoint columns performs worst. Considering the solution value for single instance sets, the

Table 3.7: Results for BAPH testing: acceleration strategies

Jobs	**Original**		**W/O stab**		**W/O dynamic**		**W/O disjCol**	
	Time(s)	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)	Obj
4	2.07	71.96	0.93	71.84	1.09	71.84	1.94	72.37
6	3.94	115.65	3.02	115.80	2.59	116.02	2.67	116.32
8	9.66	165.81	17.04	165.63	8.73	167.43	17.04	169.01
10	83.05	203.59	66.63	205.81	62.76	208.61	235.83	204.85
12	623.66	241.75	574.79	247.96	509.85	249.01	978.24	252.02
16	280.97	310.07	182.79	312.88	536.01	312.51	464.32	313.90
20	806.69	387.19	1,469.86	389.62	1,784.99	389.40	2,523.01	397.66
24	2,037.29	460.14	2,563.15	459.75	–	–	–	–
28	3,268.64	528.77	–	–	–	–	–	–
32	3,600.00	605.09	–	–	–	–	–	–
Avg.[1]	258.58	213.72	330.72	215.65	415.14	216.40	603.29	218.02

[1] For instances with 4-20 jobs

original configuration performs best for 6, 10, 12, 16 and 20-job instances and finds the solution with the best value for 54% of all tested instances up to 20 jobs. In the remaining instance sets, it is outperformed only by a small margin by the configuration without stabilization. Moreover, the two configurations without dynamic constraints and without adding several disjoint columns perform worst. Looking at the runtime for single instance sets, the original configuration is fastest for the instance sets with 20 and 24 jobs. For small and medium-sized instances except for 8-job instances, the configuration without stabilization and partly the configuration without dynamic constraints find solutions faster.

Table 3.8 shows that for 24 jobs, the original and the configuration without stabilization can find integral solutions for all instances, and that the two configurations can still solve 50% of all 32-job instances. The other two configurations drop to 62.50% for 24 jobs and cannot find a single feasible integral solution for any 32-job instance.

3.8 Conclusion

In this paper, we investigate the AVRPWVS as an example and archetype of the class of VRPMSs. The AVRPWVS deals with routing workers to jobs at different locations, determining jobs' start times and modes while synchronizing

Table 3.8: Results for BAPH testing: share of largest instances solved

Jobs	**Share of instances with feasible solution in runtime limit** (%)			
	Original	W/O stab	W/O dynamic	W/O addCols
24	100.00	100.00	62.50	62.50
28	100.00	62.50	0.00	37.50
32	50.00	50.00	0.00	0.00

the worker with vehicle routes. This work focuses on the application airport ground handling, but the AVRPWVS can also be applied to other industries such as logistics or healthcare.

We present two multi-commodity flow formulations based on a time-space network, the MILP I and MILP II. Both MILPs achieve the same integrality gap of 2.30% on average, but the MILP II performs better in terms of runtime. In addition, we develop a branch-and-price heuristic (BAPH) that, besides conventional variable branching, features a novel concept of fixing sums of modified routes. We demonstrate that the BAPH finds solutions close to the optimal solution at a comparably short runtime.

There are several possibilities for future research. First, one could investigate an extension of the BAPH to an optimal branch-and-price procedure by branching on travel arcs. Furthermore, the novel fixing approach could be applied to other VRPMSs with movement synchronization and one could apply the BAPH to the original VRPWVS by incorporating several subproblems for workers and vehicles. Moreover, an investigation of runtime and solution value of the BAPH when removing arcs could be a possible path for future research.

Chapter 4

Conclusion

This chapter summarizes the dissertation findings and provides directions for future research.

4.1 Summary

This work considers the VRPWVS as well as its variant, the AVRPWVS. The two problems belong to the class of VRPMSs and constitute a VRPMS archetype because they cover all synchronization types. Chapter 2 examines the VRPWVS from an application perspective inspired by the problem setting of a ground handling company at Munich International Airport. Chapter 3 focuses on the AVRPWVS from a theoretical perspective as one example of a VRPMS.

The application perspective in Chapter 2 extends the well-known concept of load synchronization by distinguishing between active and passive load synchronization. A MILP covering all synchronization types including active load synchronization is presented. It is based on transports, which are defined as a combination of workers and vehicles that jointly move between locations in specific relative time windows and basically model movement synchronization. To solve the VRPWVS in very short time, a novel powerful metaheuristic consisting of a tabu search, guided local search and large neighborhood search is developed. During the computational study in Section 2.6 using real-world data from Munich International Airport, the MILP model is shown to solve only small problem instances. However, limiting the number of transport categories in the MILP transforms it to an efficient heuristic and enables it to solve medium-sized instances with a low optimality gap. Moreover, the metaheuristic finds solutions with an average optimality gap of 4.43% in a very short runtime of 90 seconds and, if employed at the airport, could reduce the lateness of aircraft by 5.8 hours per day and save almost 9 million Euro per year.

In Chapter 3, two multi-commodity flow formulations for the AVRPWVS are developed: the MILP I features the routing of workers and vehicles explicitly, whereas the MILP II routes only workers explicitly and implicitly captures the movement of vehicles as a vehicle flow. To solve the AVRPWVS, a novel branch-and-price heuristic is developed. For this purpose, the MILP II is decomposed into a set-partitioning master problem and a subproblem using Dantzig-Wolfe decomposition. A dynamic programming labelling algorithm is then employed to solve the subproblem. To speed up the column generation, three acceleration strategies are employed: stabilization of the dual vector, addition of several disjoint columns in each iteration and dynamic constraints in the master problem. To obtain integral solutions, the BAPH first branches on the number of vehicles and job variables and only then starts fixing sums of modified routes in a novel way. During computational experiments, it is shown that both flow formulations feature an average integrality gap of 2.30% which explains the considerable number of branch-and-bound nodes needed by CPLEX especially for larger instances. Furthermore, the MILP II outperforms the MILP I in terms of runtime. Compared to the MILP II, the BAPH achieves near-optimal results at a much lower runtime, which does not change when solving larger instances with up to 32 jobs indicated by a largely stable gap between the LP solution value and the best BAPH integral solution value. Finally, the experiments show that the BAPH acceleration techniques are efficient as they result in the lowest average solution value and lowest average runtime when compared to other configurations.

4.2 Future research

There are several possible areas for future research on both the VRPWVS and the AVRPWVS. First, an exact branch-and-price-and-cut procedure for the AVRPWVS could be created. A first step in this endeavor could be to extend the BAPH presented in this dissertation by branching on travel arcs. In a second step, different skill levels and vehicle capacities could be introduced as additional types of subproblems to also find exact solutions to the VRPWVS. In addition, one could investigate the strategic workforce composition currently taken as an input for the VRPWVS. Such a study could help ground handling companies to better plan and size their staff, which could ultimately result in considerable cost savings.

Finally, this dissertation should encourage other researchers to further explore the class of VRPMSs as it constitutes a branch of rich VRPs with a high relevance to numerous industries and functions. Hopefully, the approaches and findings presented in this dissertation will help to achieve this goal.

Appendices

Appendix A

Abbreviations

AVRPWVS	Abstract VRPWVS
BAPH	Branch-and-price heuristic
CG	Column generation
GLS	Guided local search
LNS	Large neighborhood search
LP	Linear program
LS	Local search
MILP	Mixed integer linear program
MIP	Mixed integer program
PDP	Pickup and delivery problem
RMP	Restricted master problem
SP	Subproblem
TS	Tabu search
VRP	Vehicle routing problem
VRPMS	VRP with multiple synchronization constraints
VRPTT	VRP with trailers and transshipments
VRPTW	VRP with time windows
VRPWVS	VRP with worker and vehicle synchronization

Appendix B

Notation

B.1 Notation Chapter 2

Indices

c	Category c
i, j	Jobs i, j
k	Vehicle k
m	Mode m
w	Worker w

Sets

$\mathcal{C}$	Transport categories $\{0, \dots, 8\}$
$\mathcal{C}^a_<$, $\mathcal{C}^a_-$, $\mathcal{C}^a_>$,	Categories arriving before, during or after a job's processing
$\mathcal{C}^d_<$, $\mathcal{C}^d_-$, $\mathcal{C}^d_>$,	Categories departing before, during or after a job's processing
$\mathcal{E}$	Precedences between jobs
$\mathcal{J}$	Real jobs $\{1, \dots, n\}$
$\mathcal{J}^0$	Real jobs and start depot $\{0, \dots, n\}$
$\mathcal{J}^{n+1}$	Real jobs and end depot $\{1 \dots, n+1\}$
$\mathcal{J}^{0,n+1}$	Real jobs, start and end depot $\{0, \dots, n+1\}$

$\mathcal{K}$	Vehicles $\{1, \dots, K\}$
$\mathcal{M}_i$	Modes $\{m_i^{min}, \dots, m_i^{max}\}$ for job i
$Pred_i$	Predecessors of job i
$\mathcal{Q}$	Skill levels $\{1, \dots, Q\}$
$Succ_i$	Successors of job i
$Succ_i^+$	Successors of job i with a distance above 0
$\mathcal{W}$	Workers $\{1, \dots, W\}$

Parameters

cap_k	Capacity of vehicle k
c_i^{job}	Penalty cost for job i with too few workers
$c_{k,i}^{cap}$	Penalty cost for vehicle k after job i with too many workers
$d_{i,j}$	Travel duration between job i and j
d^{max}	Maximum travel duration for which workers are allowed to walk
ES_i, LS_i	Earliest and latest start time for job i
EA_i, LD_i	earliest and latest arrival/departure time for job i
$< i, j >$	Precedence between job i and j
$iterations_{LNS}$	Number of iterations with improvement $< \Delta_{LNS}$ to call LNS
k	Construction heuristic look-ahead depth
k_{max}	Maximum construction heuristic look-ahead depth
l_{TS}	Tabu search list length
m_i^{min}, m_i^{max}	Minimum and maximum number of workers for job i
M	Big integer constant
n_S^{it}	Number of transport swaps per iteration
n_S^{sol}	Number of transport swaps per solution
$p_{i,m}$	Processing time of job i with m workers
q_{w}^{w}	Skill level of worker w
q_{i}^{i}	Minimum skill level required for job i
w_i	Delay weight of job i

τ	Maximum worker waiting time
δ_{hybrid}	Runtime limit of hybrid metaheuristic
δ_{MILP}	Runtime limit of MILP-model
δ_{LNS}	LNS repair time limit
λ	GLS penalty weight in objective function
ρ_{start}	LNS start destroy percentage
ρ_{end}	LNS end destroy percentage
Δ_{LNS}	Improvement threshold to call LNS

Binary variables

$b_{w,i}$	1, if worker w is assigned to job i
h_i	1, if job i has too few workers
$tr_{i,j,c}$	1, if transport goes from job i to job j with category c
$v_{k,i,j,c}$	1, if vehicle k goes from job i to job j on transport of category c
$x_{w,i,j,c}$	1, if worker w travels from job i to job j on transport of category c
$y_{i,m}$	1, if job i is started in mode m

Continuous variables

$a^a_{i,c}$	Arrival time at job i with transport c
$a^d_{i,c}$	Departure time from job i with transport c
e_i	End time of job i
t_i	Start time of job i

Integer variables

$u^{\mathrm{k}}_{k,i}$	Number of visit of job i for vehicle k
$u^{\mathrm{w}}_{w,i}$	Number of visit of job i for worker w

B.2 Notation Chapter 3

Notation in addition to or with different usage than in Section B.1.

Indices

i, j	Vertices i, j
g, h	Jobs g, h
r	Route r
t	Time t
v	Vehicle v

Sets

$\mathcal{A}$	Set of all arcs
$\mathcal{A}^k$	Set of all arcs for vehicles (excl. processing and departure waiting arcs)
$\mathcal{A}^{k,wait}$	Set of all vehicle waiting arcs (i.e. arrival waiting arcs)
$\mathcal{A}^{\tilde{k}}$	Set of all arcs for vehicles in MILP II (incl. zero travel arcs and arrival waiting arcs)
$\mathcal{A}^{proc}$	Set of processing arcs
$\mathcal{A}^{trans}$	Set of transitioning arcs
$\mathcal{A}^{travel}$	Set of travel arcs
$\mathcal{A}^{travel,end}$	Set of travel arcs from the end depot $n+1$
$\mathcal{A}_i^{travel,in}$	Set of travel arcs pointing to vertex i
$\mathcal{A}^{travel,start}$	Set of travel arcs from the start depot 0
$\mathcal{A}_i^{travel,out}$	Set of travel arcs pointing from vertex i
$\mathcal{A}^{travel,w+v}$	Set of all worker and vehicle travel arcs
$\mathcal{A}_{(i,j)}^{travel,w+v}$	Set of all worker and vehicle travel arcs that are parallel to the worker travel arc (i, j)
$\mathcal{A}^{travel,>}$	Set of travel arcs with a duration > 0

$\mathcal{A}^{travel,0}$	Set of travel arcs with a duration of 0
$\mathcal{A}^{wait}$	Set of waiting arcs
F_i	Bucket (set) of non-dominated labels of vertex i
$\mathcal{R}$	Set of routes
$\mathcal{R}^{\bar{r}}$	Set of modified routes for route $\bar{r}$
$\mathcal{T}$	Discrete time horizon $\{1, \ldots, T\}$
$\mathcal{T}_h^A$	Earliest arrival and latest departure times at job h
$\mathcal{T}_h^S$	Possible start times for job h
$\mathcal{V}$	Set of vertices
$\mathcal{V}_h^{arr}$, $\mathcal{V}_h^{dep}$	Set of arrival or departure vertices per job h
$\mathcal{V}^k$	Set of vehicle vertices in MILP II
$\mathcal{V}^{sameTime}$	Set of vertices with the same time
$\mathcal{V}^{0,n+1}$	Set of vertices including dummy start and end vertices

Binary variables

f_r	1, if route r is selected
$x^{\mathrm{k}}_{w,i,j}$	1, if vehicle k goes from vertex i to j
$x^{\mathrm{w}}_{w,i,j}$	1, if worker w goes from vertex i to j
$y_{h,m,t}$	1, if job h is started with m workers in time t

Continuous variables

C^{k}_k	Auxiliary variable for costs of route of vehicle k
C^{r}_r	Auxiliary variable for costs of route r
C^{w}_w	Auxiliary variable for costs of route of worker w
$v_{i,j}$	Flow of vehicles between vertex i and j

Parameters

cap	Capacity of a vehicle
$d_{i,j}$	Travel duration between vertex i and j
(i,j)	Arc between vertex i and j
$< g,h >$	Precedence between job g and h
l_i	Label of vertex i
$p^{job}_{h,m}$	Processing time of job h with m workers
p^{max}_h	Longest processing time of job h
$p^{arc}_{i,j}$	Weight of processing arc between vertices i and j
α	Objective function weight of delays
β	Objective function weight of worker route
$\delta_{i,j,r}$	1, if the arc (i,j) is used in route r
$\delta^{w}_{i,j,r}$	1, if the worker travel arc (i,j) is used in route r
$\delta^{w+v}_{i,j,r}$	1, if the worker and vehicle travel arc (i,j) is used in route r
γ	Objective function weight of vehicle route
σ_h	Delay weight of job h
$\pi^{(3.32)}_{h,m,t}$, $\pi^{(3.34)}$, $\pi^{(3.35)}$ $\pi^{(3.36)}_i$, $\pi^{(3.37)}_{(i,j)}$	Dual values associated with RMP constraints
θ	Dual stabilization weight
π^{best}	Best dual vector
π^{last}	Latest dual vector
π^{now}	Dual vector employ in the current iteration

Appendix C

VRPWVS MILP vehicle constraints and variable definitions

$$\sum_{j \in Pred_i} \sum_{c \in \mathcal{C}} v_{k,j,i,c} = \sum_{j \in Succ_i} \sum_{c \in \mathcal{C}} v_{k,i,j,c} \qquad \forall\, k \in \mathcal{K}, i \in \mathcal{J} \tag{C.1}$$

$$\sum_{j \in Pred_i} \sum_{c \in \mathcal{C}^a_-} v_{k,j,i,c} \leq \sum_{j \in Succ_i} \sum_{c \in \mathcal{C}^d_{-,>}} v_{k,i,j,c} \qquad \forall\, k \in \mathcal{K}, i \in \mathcal{J} \tag{C.2}$$

$$\sum_{j \in Pred_i} \sum_{c \in \mathcal{C}^a_>} v_{k,j,i,c} \leq \sum_{j \in Succ_i} \sum_{c \in \mathcal{C}^d_>} v_{k,i,j,c} \qquad \forall\, k \in \mathcal{K}, i \in \mathcal{J} \tag{C.3}$$

$$x_{w,i,j,c} \in \{0,1\} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J}^0, j \in Succ_i, c \in \mathcal{C} \tag{C.4}$$

$$b_{w,i} \in \{0,1\} \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J}^{0,n+1} \tag{C.5}$$

$$t_i \in [ES_i, LS_i] \qquad \forall\, i \in \mathcal{J}^{0,n+1} \tag{C.6}$$

$$y_{i,m} \in \{0,1\} \qquad \forall\, i \in \mathcal{J}, m \in \mathcal{M}_i \tag{C.7}$$

$$u^v_{k,i} \in [0,n] \qquad \forall\, k \in \mathcal{K}, i \in \mathcal{J}^0 \tag{C.8}$$

$$u^w_{w,i} \in [0,n] \qquad \forall\, w \in \mathcal{W}, i \in \mathcal{J}^0 \tag{C.9}$$

$$v_{k,i,j,c} \in \{0,1\} \qquad \forall\, k \in \mathcal{K}, i \in \mathcal{J}^0, j \in Succ_i, c \in \mathcal{C} \tag{C.10}$$

$$tr_{i,j,c} \in \{0,1\} \qquad \forall\, i \in \mathcal{J}^0, j \in Succ_i, c \in \mathcal{C} \tag{C.11}$$

$$a^d_{i,c} \in [EA_i, LD_i] \qquad \forall\ i \in \mathcal{J}^0, c \in \mathcal{C} \tag{C.12}$$

$$a^a_{i,c} \in [EA_i, LD_i] \qquad \forall\ i \in \mathcal{J}^{n+1}, c \in \mathcal{C} \tag{C.13}$$

$$e_i \geq 0 \qquad \forall\ i \in \mathcal{J}^{n+1} \tag{C.14}$$

Appendix D

Example

We briefly outline a graph for the VRPWVS illustrating our example as shown in Section 2.4.3. Note that in the graph below, nodes represent jobs and arcs represent transports. For each transport arc, we denote the category, the set of workers and the set of vehicles as $\{c, \mathcal{W}, \mathcal{K}\}$.

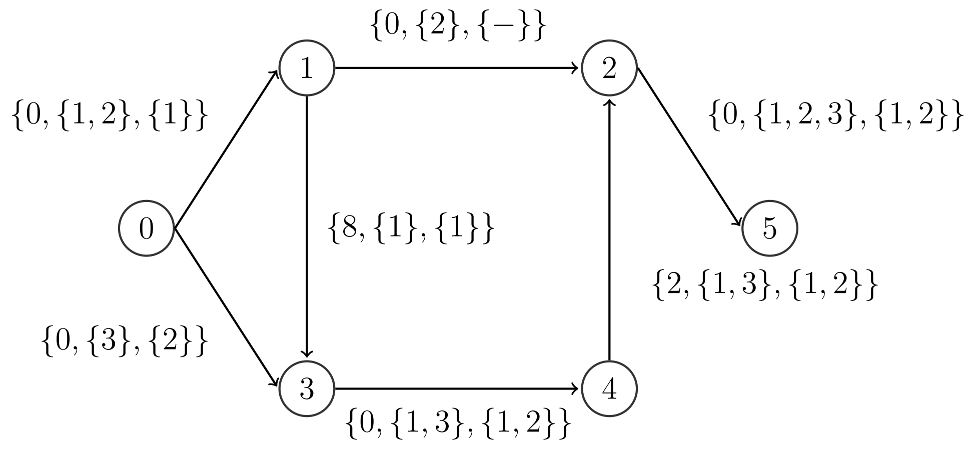

Figure D.1: Illustration of a solution for the example

Bibliography

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1st edition.

Andreatta, G., Giovanni, L. D., and Monaci, M. (2014). A Fast Heuristic for Airport Ground-Service Equipment and Staff Allocation. *Procedia - Social and Behavioral Sciences*, 108(0):26 – 36.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329.

Boeing Commercial Airplanes (2013). Current Market Outlook 2013 to 2032. Internet.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2015). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300 – 313.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41.

Christofides, N., Alvarez-Valdés, R., and Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, 29(3):262–273.

Clausen, T. (2011). Airport ground staff scheduling. Doctoral dissertation, Technical University of Denmark.

Cook, A., Tanner, G., and Anderson, S. (2004). Evaluating the true cost to airlines of one minute of airborne or ground delay. Technical report, Eurocontrol.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, 58(1):179–192.

Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2002). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In Ribeiro, C. C. and Hansen, P., editors, *Essays and Surveys in Metaheuristics*, volume 15, pages 309–324. Springer, Boston, MA.

Desrosiers, J. and Lübbecke, M. E. (2005). A primer in column generation. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, volume 1, pages 1–32. Springer, New York.

Desrosiers, J., Soumis, F., and Desrochers, M. (1984). Routing with time windows by column generation. *Networks*, 14(4):545–565.

Dohn, A., Kolind, E., and Clausen, J. (2009). The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers & Operations Research*, 36(4):1145–1157.

Drexl, M. (2007). On some generalized routing problems. Doctoral dissertation, RWTH Aachen University.

Drexl, M. (2012). Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3):297–316.

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

Drexl, M., Rieck, J., Sigl, T., and Press, B. (2013). Simultaneous Vehicle and Crew Routing and Scheduling for Partial- and Full-Load Long-Distance Road Transport. *BuR - Business Research*, 6(2):242–264.

Eurocontrol (2012). Performance Review Report. An Assessment of Air Traffic Management in Europe during the Calendar Year 2011. Technical report, EUROCONTROL.

Faroe, O., Pisinger, D., and Zachariasen, M. (2003). Guided local search for the three-dimensional bin-packing problem. *INFORMS Journal on Computing*, 15(3):267–283.

Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., and Soumis, F. (2016a). Column Generation for Vehicle Routing Problems with Multiple Synchronization Constraints. Working paper, TUM School of Management, Technische Universität München.

Fink, M., Frey, M., Kiermaier, F., and Kolisch, R. (2016b). Synchronized worker and vehicle routing for ground handling at airports. Working paper, TUM School of Management, Technische Universität München.

Franceschi, R. D., Fischetti, M., and Toth, P. (2006). A new ILP-based refinement heuristic for Vehicle Routing Problems. *Mathematical Programming*, 105(2-3):471–499.

Frey, M., Kolisch, R., and Artigues, C. (2016). Column generation for outbound baggage handling at airports. Working paper, TUM School of Management, Technische Universität München.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Golden, B. L., Raghavan, S., and Wasil, E. A. (2008). *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media.

Grønhaug, R., Christiansen, M., Desaulniers, G., and Desrosiers, J. (2010). A branch-and-price method for a liquefied natural gas inventory routing problem. *Transportation Science*, 44(3):400–415.

Haase, K., Desaulniers, G., and Desrosiers, J. (2001). Simultaneous vehicle and crew scheduling in urban mass transit systems. *Transportation Science*, 35(3):286–303.

Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14.

Herbers, J. (2005). Models and Algorithms for Ground Staff Scheduling on Airports. Doctoral dissertation, RWTH Aachen University.

Herbers, J. (2006). Representing labor demands in airport ground staff scheduling. In *Operations Research Proceedings 2005*, pages 15–20. Springer.

Hollis, B., Forbes, M., and Douglas, B. (2006). Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post. *European Journal of Operational Research*, 173(1):133–150.

Hooker, J. N. and Natraj, N. R. (1995). Solving a General Routing and Scheduling Problem by Chain Decomposition and Tabu Search. *Transportation Science*, 29(1):30–44.

Ip, W. H., Wang, D., and Cho, V. (2013). Aircraft Ground Service Scheduling Problems and Their Genetic Algorithm With Hybrid Assignment and Sequence Encoding Scheme. *IEEE Systems Journal*, 7(4):649–657.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, volume 1, pages 33–65. Springer, New York.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., and Vanderbeck, F. (2010). Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36:695–702.

Kara, I., Laporte, G., and Bektas, T. (2004). A Note on the Lifted Miller-Tucker-Zemlin Subtour Elimination Constraints for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*, 158(3):793 – 795.

Kiermaier, F., Frey, M., and Bard, J. F. (2016a). The flexible break assignment problem for large tour scheduling problems with an application to airport ground handlers. Working paper, TUM School of Management, Technische Universität München.

Kiermaier, F., Frey, M., and Bard, J. F. (2016b). Flexible cyclic rostering in the service industry. Working paper, TUM School of Management, Technische Universität München.

Kim, B.-I., Koo, J., and Park, J. (2010). The combined manpower-vehicle routing problem for multi-staged services. *Expert Systems with Applications*, 37(12):8424–8431.

Kolisch, R. and Sprecher, A. (1997). PSPLIB - A project scheduling problem library. *European Journal of Operational Research*, 96(1):205–216.

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.

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics*, 54(8):811–819.

Lenstra, J. K. and Rinnooy Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected Topics in Column Generation. *Operations Research*, 53(6):1007–1023.

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

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329.

Nguyen, P. (2014). Meta-Heuristic Solution Methods for Rich Vehicle Routing Problem. Doctoral dissertation, CIRRELT.

Nowak, M., Ergun, O., and White, C. C. (2008). Pickup and Delivery with Split Loads. *Transportation Science*, 42(1):32–43.

Padrón, S., Guimarans, D., Ramos, J. J., and Salma, F.-T. (2016). A bi-objective approach for scheduling ground-handling vehicles in airports. *Computers & Operations Research*, 71:34–53.

Rapajic, J. (2009). *Beyond Airline Disruptions.* Ashgate Publishing Limited, Surrey, UK.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98*, pages 417–431. Springer.

Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., and Meisel, F. (2015). Branch-and-price for the active-passive vehicle-routing problem. Working paper, Johannes Gutenberg-Universität Mainz.

Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*, volume 18. Siam.

Tsang, E. and Voudouris, C. (1997). Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119–127.

Visser, T. R. (2015). Synchronization in simultaneous vehicle and crew routing and scheduling problems with breaks. Master's thesis, Universiteit Utrecht.

Voudouris, C. and Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499.

Wentges, P. (1997). Weighted Dantzig–Wolfe Decomposition for Linear Mixed-integer Programming. *International Transactions in Operational Research*, 4(2):151–162.

Xiang, Z., Chu, C., and Chen, H. (2006). A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, 174(2):1117–1139.