A Multi-Depot Pickup and Delivery Problem with a Single Hub and Heterogeneous Vehicles

Stefan Irnich
Lehrstuhl für Unternehmensforschung (Operations Research),
Rheinisch-Westfälische Technische Hochschule Aachen, Germany
e-mail: sirnich@or.rwth-aachen.de

Abstract

This paper introduces a special kind of multi-depot pickup and delivery problem. In contrast to the general pickup and delivery problem (GPDP, see e.g. [19, 31]) all requests have to be picked up at or delivered to one central location which has the function of a hub or consolidation point. In hub transportation networks routes between customers and the hub are often short, i.e. involve only one or very few customers. The reason for this can been seen in narrow time windows as well as in high quantities which make it possible to fully load a vehicle at one customer. Thus, the focus here is on problems where all possible routes can easily be enumerated, i.e. the problem primarily considers the assignment of transportation requests to routes. We assume that many problems in transportation logistics can be modeled and solved similarly whenever routes can be enumerated and the temporal aspects of transportation requests are important.

Keywords: Transportation, Modeling, Pickup and Delivery, Hub

1 Introduction

This paper introduces a special kind of multi-depot pickup and delivery problem. It has applications in transportation networks where a division into global and local services is possible. Here we assume global services between main locations to be fixed. Local services are transportation requests feeding each of the locations on the main network, the so-called hubs. The design of a feeding network for a single hub means to cover a set of transportation requests, pickups as well as deliveries. In contrast to the general pickup and delivery problem

(GPDP, see e.g. [19, 31]) all requests have to be either picked up at or delivered to the hub.

The problem we examine here considers multiple depots and heterogeneous vehicles. It is, therefore, called 'multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles' (MDPDPSH). In the following pickup always means to load something at a location and deliver it to the hub. Delivery is always defined as loading some good at the hub and deliver it to a location. One characteristic of hub transportation networks is that many different requests have to be transported between one location and the hub. Thus, in general we assume that there are different requests for pickup or delivery at a specific location, i.e. requests with a different quantity and with different time windows occur. For the rest of the paper we further assume that all pickup or delivery locations are also depots of a given heterogeneous fleet of vehicles.

The main difference to most other vehicle routing and scheduling problems is that planning primarily decides on the assignment of requests to a vehicle. The routing information is less important than the decision which requests are transported by the same vehicle and what type of vehicle is used.

Many vehicle routing and scheduling problems can be formulated as set partitioning problems (SPP) or set covering problems (SCP). Very successful solution methods have been developed around these formulations. They all have in common that requests R have to be fulfilled by a cost minimal subset T^* of the set T of all feasible trips. The costs of T^* are always defined as the sum of the costs c_t of all selected trips $t \in T^*$. In the SPP or SCP columns correspond to feasible trips. They can be described by the incidence vector $z^t = (z^t_1, \ldots, z^t_{|R|})$ which is a vector of $|R|$ entries either one or zero indicating if the trip t covers the corresponding requests or not. The set covering or set partitioning models can be stated as

$$\min \sum_{t \in T} c_t \lambda_t \tag{1}$$

subject to

$$\sum_{t \in T} z^t_r \lambda_t \left\{ \begin{matrix} \geq \\ = \end{matrix} \right\} 1 \quad \text{for all } r \in R \tag{2}$$

$$\lambda_t \in \{0, 1\} \quad \text{for all } t \in T. \tag{3}$$

Classes of problems differ in their particular definition of a request and possibly some additional constraints which have to be added to the model (1) to (3). For the vehicle routing problem (VRP) requests usually mean visiting customers to meet their demands using a vehicle of restricted capacity. The VRP with time windows (VRPTW, see [16]) additionally takes into account a time window for every customer. For pickup and delivery problems (see e.g. [19, 31, 32]) a request corresponds to a transportation of a quantity from a pickup location to a delivery location. In a vehicle scheduling environment (see e.g. [29]) a request normally is a trip between two locations with a fixed starting time and duration. Desaulniers et al. [17] present an excellent overview and a unified framework for deterministic time constrained vehicle routing and crew scheduling problems.

We will show that a two-stage procedure, which first enumerates all (relevant) columns and then solves the SCP model using a set covering heuristic, is an easy to implement algorithm. Two-stage algorithms of this kind are widely used in the area of airline scheduling (see e.g. [18]). In the crew pairing problem, for example, a set of pairings (a sequence of duties of a crew forming a legal workday, starting and ending at a crew base) has to be constructed to cover a set of flight legs at minimal costs.

The aim of this paper is to present some new algorithmic and methodological results. We describe a heuristic algorithm which solves large scale instances of the MDPDPSH. But we focus on instances where the time windows of the transportation requests are relatively narrow. Consequently most of the routes between the depots and the hub are short routes. This specific property of the problem enables us to give a new formulation of the concepts of feasible and efficient trips.

For other routing and scheduling problems like the VRP, VRPTW, GPDP, and MDVSP the most or only successful solution methods are based on column generation and branch-and-price (an overview is given in [5], special purpose algorithms are presented e.g. in [1, 16, 30, 32]). We will also show how column generation may be used for solving the MDPDPSH. The decomposition we use in the enumeration procedure leads to a pricing problem with a new structure of a combined clique and knapsack problem. A subsequent paper will present results on this approach.

Our work was inspired by a practical problem at the Deutsche Post

AG, Germany´s post service. Reorganization of the Deutsche Post AG has forced massive structural and organizational changes. A new transportation network has been designed to reduce operation costs while keeping the service standards high. The Elite Foundation in collaboration with the department of Operations Research of RWTH Aachen have developed a decision support system which assists planners at Deutsche Post AG to improve their plans (see [22, 23]). The MDPDPSH is one subproblem, the ground feeding problem, in the design process of the global area transportation network (GATN): 84 letter mail centers (LMCs) are located in different regions of Germany. Two out of them have to exchange the letter mail addressed to customers in the other region. Because of time restrictions letter mail transportation is partly done by air-services (see [13]). So LMCs situated near one airport have to deliver letter mail to flights starting at the airport. On the other hand, letter mail from flights landing at the airport has to be picked up and delivered to the LMCs. The airport corresponds to a hub and the LMCs correspond to pickup and delivery points. The letters which have to be transported between LMCs and flights represent the transportation requests.

This paper is organized as follows. In section 2 the MDPDPSH is defined and a network model is presented. Starting from this model we develop a new description of feasible and efficient trips. These descriptions stem from the decomposition of the entire problem into assignment subproblems for each route/vehicle pair. In section 3 we present the algorithm we have developed for the MDPDPSH. The algorithm consists of two phases. In the first phase efficient feasible trips are enumerated. An algorithm to enumerate relevant route/vehicle combinations as well as an algorithm to enumerate assignments of requests to each route/vehicle combination are presented. The second phase uses a set covering heuristic for which we give some brief remarks. In section 4 we present the computational results for 30 real-world test instances of the MDPDPSH. In section 5 we discuss the application of the branch- and-price methodology to the MDPDPSH. The structure of the pricing problem as well as some remarks on the branching scheme are included. Finally, in section 6 we make some concluding remarks.

2 The multi-depots pickup and delivery problem with a single hub (MDPDPSH)

We give a formulation of the MDPDPSH as a network model. Although the model is mathematically well-defined, the problem of representing feasible trips can be done in a (what we believe) more adequate manner. We show that a decomposition of trips into route/vehicle combinations, on the one hand, and transported requests, on the other hand, opens up new possibilities in this direction.

2.1 Notation

Let $R = \{r_1, \ldots, r_{|R|}\}$ be the set of transportation requests between the hub or consolidation point 0 and some locations $N = \{1, \ldots, n\}$. Every request involves the hub 0, so the set R can be partitioned into two subsets R_p and R_d, i.e. $R = R_p \cup R_d$. On the one hand, requests $r \in R_p$ require a pickup at location $v(r)$ and a delivery to the hub 0. On the other hand, requests $r \in R_d$ have to be picked up at the hub 0 and delivered to location $v(r)$. R_p are called pickup requests and R_d are called delivery requests. Every request $r \in R$ has an associated time window $[a_r, b_r]$. The entire transportation process has to take place within this time window. Consequently, a_r is the earliest pickup time and b_r is the latest delivery time. A quantity q_r has to be transported.

All locations $N = \{1, \ldots, n\}$ serve as depots of vehicles. That means that all vehicles starting at depot $v \in N$ have to return to the same depot v at the end of the planning period. The vehicle fleet is assumed to be heterogeneous and F is the set of all different types of vehicles. Vehicle types differ in cost and time for driving and loading, and capacity. At depot $v \in N$ the subset F_v of vehicle types is located. The number of vehicles of a certain type available at a depot $v \in N$ is unbounded. We have to point out that we always (except for section 2.3) consider only *types* of vehicles $f \in F$ but for short we speak of a vehicle f where no confusion may occur.

In order to describe different costs and times of a vehicle f on its way to the hub or back from the hub we double all locations. Let $N^- = \{1^-, \ldots, n^-\}$ be the set of pickup locations and $N^+ = \{1^+, \ldots, n^+\}$ be the set of delivery locations. 0^- corresponds to the entrance and 0^+ to the exit of the hub. We set $N_0^- = N^- \cup \{0^-\}$ and $N_0^+ = N^+ \cup \{0^+\}$. The transportation process done by a vehicle of type f can be modeled

by a weighted digraph $(V, A; (c_{ij}^f), (t_{ij}^f))$ where $V = N_0^- \cup N_0^+$ is the set of nodes and

$$A = (N_0^- \times N_0^-) \cup (N_0^+ \times N_0^+) \cup \{(0^-, 0^+)\} \cup \{(v^+, v^-) | v \in N\}$$

is the set of arcs. Let t_{ij}^f be the time and c_{ij}^f be the costs of driving and possibly loading when vehicle f drives from i to j. For $i, j \in N^-$ or $i, j \in N^+$ loading at i or unloading at j respectively is included. Arcs of the form $(i0^-)$ mean loading at $i \in N^-$, driving from i to the hub, and unloading at the hub. For arcs $(0^+ j)$ with $j \in N^+$ we have an analogue meaning. The arc $(0^- 0^+)$ is used when transportation to and from the hub is considered. Finally, the arcs of the form (v^+, v^-) are needed to model routes as cycles in (V, A). Thus, we make a clear distinction between nodes of this network V and locations N. An example of a small network is presented in figure 1 and will be discussed in greater detail in section 2.3.

For all depot locations $v \in N^- \cup N^+$ the set

$$R(v) = \{r \in R | v(r) = v\}$$

includes all requests which have to be picked up from v for $v \in N^-$ or delivered to v for $v \in N^+$ respectively.

Finally, waiting is penalized by a cost factor c_{wait}^f according to the time the vehicle f waits. Obviously only waiting at the hub location makes sense. But waiting at the hub is required whenever some pickup request $r \in R_p$ and some delivery request $r' \in R_d$ having $b_r < a_{r'}$ are transported by the same vehicle.

2.2 Trips

The problem is to find a cost minimal set of trips which realize all transportation requests. Thus, we have to concentrate on the definition of a trip: a trip $t = (p, f, R_t)$ is a combination of

1. a route p;
   All routes have to be cycles starting and ending at the same depot v_1. The cycle $p = (v_1, \ldots, v_{n_p}, v_{n_p+1} = v_1)$ in (V, A) has to visit 0^- exactly once, or 0^+ exactly once, or both. Consequently, three types of routes can occur:

(a) A route can visit only pickup locations and the hub and, thus, is of the form $(v_1, \ldots, v_{n_p-1}, 0^-, v_1)$ with $\{v_1, \ldots, v_{n_p-1}\} \subset N^-$.

(b) On the contrary it can visit the hub and only delivery nodes, i.e. $p = (v_1, 0^+, v_3, \ldots, v_{n_p}, v_1)$ with $\{v_3, \ldots, v_{n_p}, \} \subset N^+$.

(c) A route can visit both pickup and delivery locations and, therefore, p is of the form $(v_1, \ldots, v_k, 0^-, 0^+, v_{k+3}, \ldots, v_{n_p}, v_1)$ with $\{v_1, \ldots, v_k\} \subset N^-$ and $\{v_{k+1}, \ldots, v_{n_p}\} \subset N^+$.

For short we call a route of type (a) pickup route, a route of type (b) delivery route, and a route of type (c) pickup and delivery route.

2. a vehicle of type f;

3. a subset of requests $R_t \subset R$;
Only requests belonging to visited locations can be transported. On the route $p = (v_1, \ldots, v_{n_p}, v_1)$ the requests

$$R(p) = \bigcup_{i \in \{1,\ldots,n_p\}: v_i \neq 0^-, 0^+} R(v_i)$$

can possibly be transported.

In order to describe the feasibility of trips three conditions have to be satisfied:

- (Availability) The vehicle of type f must be available at the depot v_1 from where the trip starts (and where it ends), i.e. $f \in F_{v_1}$.

- (Capacity) The capacity of the vehicle f must be large enough on the sub-route to the hub as well as on the sub-route from the hub to the depot, i.e.

$$\sum_{r \in R_p \cap R_t} q_r \leq Q^f \quad \text{and} \quad \sum_{r \in R_d \cap R_t} q_r \leq Q^f. \tag{4}$$

- (Time Compatibility) All requests R_t transported by trip t must be compatible concerning their time windows. We will focus on this aspect in section 2.4.

2.3 Network Model

The model we present now is a network model which deals with *vehicles* and not *types of vehicles*. Therefore, let K be a set of vehicles and $f(k) \in F$ the type of vehicle $k \in K$. As before a route can be defined as a cycle in the network but to deal with temporal aspect we need a start and an end of a cycle. Therefore, we define backward arcs B as the connection of the end of the route with its start:

$$B = \{(0^-, j)|j \in N^-\} \cup \{(j, 0^+)|j \in N^+\} \cup \{(j^- j^+)|j \in N\}$$

In order to ensure that a vehicle of a certain type f is available at the depot where the trip starts (availability) we define a set of forbidden (backward) arcs

$$\begin{aligned} X^f &= \{(0^-, j)|j \in N^-, f \notin F_j\} \cup \{(j, 0^+)|j \in N^+, f \notin F_j\} \\ &\quad \cup \{(j^- j^+)|j \in N, f \notin F_j\}. \end{aligned}$$

All backward arcs to and from locations j where vehicle f is not available are not allowed. Let A^f be the arc set A from which the forbidden arcs X^f are removed. Every cycle in (V, A^f) corresponds to a feasible route p (for some vehicle of type f) if and only if the hub node 0^-, the hub node 0^+ or both are visited exactly once. This is equivalent to the condition that one of the arcs belonging to

$$H_0 = \{(0^- i)|i \in N^-\} \cup \{(0^- 0^+)\} \cup \{(i 0^+)|i \in N^+\}$$

is used. A dummy arc $(0^- 0^-)$ is added to the arc set A^f (and also to the sets B and H_0) in order to allow not using a vehicle k. Figure 1 shows the network (V, A^f) for the case of three depots $N = \{1, 2, 3\}$ and one type of vehicle f only available at depot 1 and 3, the forbidden arcs $X^f = \{(0^-, 2^-), (2^+, 0^+), (2^+, 2^-)\}$ are removed from the arc set A^f. The network model can now be stated as follows: the decision variables x_{ij}^k describe the route of vehicle k, the variables z_r^k describe whether request r is assigned to vehicle k or not, and T_j^k is the time of arriving/leaving location j with vehicle k.

Figure 1 about here

$$\min \sum_{k \in K} \left(\sum_{(ij) \in A^{f(k)}} c_{ij}^{f(k)} x_{ij}^k \right) + c_{wait}^{f(k)} \left(T_{0^+}^k - T_{0^-}^k \right) \tag{5}$$

subject to

$$\sum_{k \in K} z_r^k = 1 \qquad \text{for all } r \in R \tag{6}$$

$$\sum_{j:(lj) \in A^{f(k)}} x_{lj}^k = \sum_{j:(jl) \in A^{f(k)}} x_{jl}^k \qquad \text{for all } k \in K, l \in V \tag{7}$$

$$\sum_{(ij) \in H_0} x_{ij}^k = 1 \tag{8}$$

$$z_r^k = 1 \Longrightarrow \sum_{j \in N_0^- : (lj) \in A^{f(k)}} x_{lj}^k = 1$$
$$\text{for all } k \in K, r \in R_p, l = v(r) \in N^- \tag{9}$$

$$z_r^k = 1 \Longrightarrow \sum_{j \in N_0^+ : (jl) \in A^{f(k)}} x_{jl}^k = 1$$
$$\text{for all } k \in K, r \in R_d, l = v(r) \in N^+ \tag{10}$$

$$z_r^k = 1 \Longrightarrow a_r \leq T_{v(r)}^k \qquad \text{for all } r \in R_p \tag{11}$$

$$z_r^k = 1 \Longrightarrow T_{0^-}^k \leq b_r \qquad \text{for all } r \in R_p \tag{12}$$

$$z_r^k = 1 \Longrightarrow a_r \leq T_{0^+}^k \qquad \text{for all } r \in R_d \tag{13}$$

$$z_r^k = 1 \Longrightarrow T_{v(r)}^k \leq b_r \qquad \text{for all } r \in R_d \tag{14}$$

$$x_{ij}^k = 1 \Longrightarrow T_i^k + t_{ij}^{f(k)} \leq T_j^k \quad \text{for all } k \in K, (ij) \in A^{f(k)} \setminus B \tag{15}$$

$$\sum_{r \in R_p} z_r^k q_r \leq Q^{f(k)} \qquad \text{for all } k \in K \tag{16}$$

$$\sum_{r \in R_d} z_r^k q_r \leq Q^{f(k)} \qquad \text{for all } k \in K \tag{17}$$

$$T_{0^+}^k - T_{0^-}^k \geq 0 \quad \text{for all } k \in K \tag{18}$$

$$x_{ij}^k \in \{0, 1\} \qquad \text{for all } k \in K, (ij) \in A^{f(k)} \tag{19}$$

$$z_r^k \in \{0, 1\} \qquad \text{for all } r \in R, k \in K \tag{20}$$

$$T_j^k \geq 0 \qquad \text{for all } j \in V, k \in K \tag{21}$$

The costs (5) consist of two components, one is route dependent and the other is time-dependent (waiting at the hub). The constraints (6) ensure that every request is transported. (7) are the classical network flow constraints of the routes. Every route visits the hub 0 exactly once because of constraint (8). Constraints (9) and (10) ensure that a vehicle k visits all pickup and delivery locations for requests transported by k. (11) to (15) force time compatibility. (16) and (17) are the capacity

constraints of the route from the depot to the hub and from the hub to the depot respectively. Finally, constraint (18) ensures that the waiting time is non-negative.

In contrast to formulations of the GPDP (see e.g. [19, 31]) this formulation does not have a different node for every pickup location and every delivery location in the network (V, A^f). Instead of this we have only one node for geographically identical locations but we distinguish between pickup locations and delivery locations. The advantage of reducing the number of request locations is not apparent if we try to use a standard method for solving (5) to (21). This would mean to decompose the entire problem into a master problem and one subproblem for each depot and each vehicle. The master problem is a set partitioning problem of the form (1) to (3). The subproblems are shortest path problems with side constraints concerning time, precedence relations, and capacity (see e.g. [32]).

2.4 Decomposition into route/vehicle combinations

We decided to do a different decomposition because of the special characteristic of the presented problem. The question in the MDPDPSH is not really to determine routes for vehicles. Because of the relatively narrow time windows of the requests and the large quantities (compared to the vehicle capacities) which have to be transported most routes are short, i.e. direct trips between depot and hub are favoured most of the time. Only a fraction of routes visits some additional pickup or delivery locations which are different from the start-depot.

In section 3.1 we give a possible definition of what 'relevant combinations' could mean. We now take a closer look at the advantage of having such a combination (p, f) of a route p and a type of vehicle f. Remember that columns of the SPP/SCP (1) to (3) correspond to trips and that a trip $t = (p, f, R_t)$ is a combination of a route p, a type of vehicle f, and some requests R_t. If a feasible combination of a route p and a vehicle f is given, then the feasibility of the trip t only depends on the choice of the requests R_t. For the rest of this subsection we focus on this aspect.

First of all for given p and f we know the set of all requests $R(p)$, which can be served by the route p. The question is whether a subset $R_t \subseteq R(p)$ of requests can be transported by the same vehicle f on the route p. The choice of the requests R_t has to take into account three

aspects:

- Time compatibility: We can give an exact definition of what time compatibility means when we look at the constraints (11) to (15) for a given vehicle k (of type $f(k)$) and a given route p (which has a correspondence in the decision variables x_{ij}^k for $(ij) \in A^{f(k)}$). All feasible values of the variables z_r^k and T_i^k corresponding to the constraints (9) to (15) imply a time compatible set $R_t = \{r \in R(p) | z_r^k = 1\}$. Although time compatibility is now clearly defined, we give a more adequate formulation as a clique at the end of this section.

- Capacity: As we have seen before the capacity constraints are given by (4) or similarly in the network model by (16) and (17).

- Route compatibility: It only makes sense to visit a location $v \in N^- \cup N^+$ if at least one request is picked up (for $v \in N^-$) or delivered (for $v \in N^+$). That means that the set R_t must contain at least one element of every set $R(v_i)$ (with $v_i \neq 0^-, 0^+$) if the route p is of the form $(v_1, v_2, \ldots, v_{n_p}, v_1)$.

Concerning time compatibility the advantage of knowing route p and vehicle f is that time compatibility becomes a property depending on pairs of requests $r_1, r_2 \in R(p)$. For a request $r \in R$ it is now possible to compute a corresponding 'hub time window' $\mathrm{htw}(r, p, f)$. For pickup requests $r \in R_p$ this time window indicates the time interval possible for arriving at the hub when transporting r on route p with vehicle f. Let $r \in R(p) \cap R_p$ be a pickup request of route $p = (v_1, \ldots, v_k, 0^-, \ldots)$. The request r is transported from its pickup location $v(r) = v_j$ (for some j with $1 \leq j \leq k$) to the hub 0^-. The time for this transportation process is

$$d(r, p, f) := \sum_{i=j}^{k} t_{v_i v_{i+1}}^f \qquad (\text{for } r \in R(p) \cap R_p).$$

The time window for arriving at the hub is, therefore, given by

$$\mathrm{htw}(r, p, f) = [a_r + d(r, p, f), b_r] \qquad (\text{for } r \in R(p) \cap R_p).$$

Analogously for a delivery request $r \in R(p) \cap R_p$ on the route p the time window $\mathrm{htw}(r, p, f)$ describes the time interval for leaving the hub

under the condition that r is transported. If the route p is given by $p = (v_1, \ldots, 0^+, v_k, \ldots, v_{n_p}, v_1)$ and request r has to be delivered to location $v(r) = v_j$ (with $k \leq j \leq n_p$), then the time of the transportation process is

$$d(r, p, f) := \sum_{i=k}^{j} t^f_{v_{i-1}v_i} \qquad (\text{for } r \in R(p) \cap R_d).$$

The hub time window for leaving the hub is

$$\text{htw}(r, p, f) = [a_r, b_r - d(r, p, f)] \qquad (\text{for } r \in R(p) \cap R_d).$$

The question whether two requests $r, r' \in R(p)$ can be transported by the same vehicle f on route p can be decided by looking at the hub time windows $\text{htw}(r, p, f)$ and $\text{htw}(r', p, f)$. Two pickup requests or two delivery requests r, r' are compatible if their hub time windows have a non-empty intersection. On the other hand, a pickup request r and a delivery request r' are compatible if delivery of r to the hub can be done before pickup of r' at the hub. Using this information we define a binary symmetric relation $\overset{(p,f)}{\sim}$ by

$$r \overset{(p,f)}{\sim} r'$$
$$\iff \begin{cases} \text{htw}(r, p, f) \cap \text{htw}(r', p, f) \neq \emptyset & \text{if } r, r' \in R_p \\ & \text{or } r, r' \in R_d, \\ a_r + d(r, p, f) \leq b_{r'} - d(r', p, f) & \text{if } (r \in R_p \text{ and } r' \in R_d). \end{cases}$$

Especially the fact $\text{htw}(r, p, f) = \emptyset$ means that request r cannot be transported by vehicle f on the route p because the transportation process needs more time than the time window $[a_r, b_r]$ allows.

We are now able to describe the region of all feasible assignments of requests $R_t \subset R(p)$ to a route/vehicle combination (p, f). Let $p = (v_1, \ldots, v_{n_p}, v_1)$ be the route and $f \in F_{v_1}$ a type of vehicle available at the depot v_1. The indicator variables $z_r \in \{0, 1\}$ for every $r \in R(p)$ are equal to one if and only if $r \in R_t$.

$$z_r + z_{r'} \leq 1 \quad \text{for all } r, r' \in R(p) \text{ with } r \overset{(p,f)}{\not\sim} r' \tag{22}$$

$$\sum_{r \in R_p} q_r z_r \leq Q^f \tag{23}$$

$$\sum_{r \in R_d} q_r z_r \leq Q^f \tag{24}$$

$$\sum_{r \in R(v_i)} z_r \geq 1 \quad \text{for all } 1 \leq i \leq n_p \text{ with } v_i \neq 0^-, 0^+ \tag{25}$$

$$z_r = 0 \quad \text{for all } r \in R(p) \text{ with } \mathrm{htw}(r, p, f) = \emptyset \tag{26}$$

$$z_r \in \{0, 1\} \quad \text{for all } r \in R(p) \tag{27}$$

The structure of constraints (22) to (27) is now clearly visible. The requests in R_t have to satisfy the clique constraints (22). To elaborate this we define a graph $G^{(p,f)} = (R(p), E^{(p,f)})$ with the set of possible requests as nodes and a set of edges corresponding to the compatibility relation $\overset{(p,f)}{\sim}$, i.e. $(r, r') \in E^{(p,f)}$ if and only if $r \overset{(p,f)}{\sim} r'$. Any time compatible set $R_t \subset R(p)$ is a clique in $G^{(p,f)}$, i.e. a subset of pairwise adjacent nodes. The constraints (23) and (24) form knapsack constraints on the pickup requests and the delivery requests in R_t respectively. Finally we add the route compatibility constraints (25) and the constraints (26) and call them 'additional constraints'.

Example 1 In this example $p = (1^-, 2^-, 0^-, 0^+, 1^+, 1^-)$ is a pickup and delivery route. The times for vehicle f are $t^f_{1^-2^-} = 50$, $t^f_{2^-,0^-} = 100$, $t^f_{0^-0^+} = 0$, and $t^f_{0^+1^+} = 120$ minutes, the capacity of vehicle f is $Q^f = 10$. The set $R(p)$ consists of eight requests, two and three pickup requests at depot 1^- and 2^- respectively, and three delivery requests to the start-depot 1^+. Table 1 describes them in detail.

Table 1 about here

The table also contains the hub time windows $\mathrm{htw}(r, p, f)$, which can easily computed from the data above. Figure 2 shows the graph $G^{(p,f)}$ with some additional information. A feasible subset of requests $R_t \subseteq R(p)$ corresponds to a subset of nodes which is a clique (time compatibility), has weights (the number above a node is its weight q_r) which satisfy the two knapsack constraints for pickup and delivery (capacity), and contains at least one node from every dotted box corresponding to the different depot locations (route compatibility).

Figure 2 about here

3 The Algorithm

Using the set partitioning/covering model (1) to (3) explicitly implies a two-phase algorithm. First we have to construct the model by enumerating all relevant trips. The decomposition of this task into the enumeration of all relevant route/vehicle combinations and after that

assigning all relevant subsets of requests to all route/vehicle combinations is the approach we follow in this paper. The point is that checking the efficiency of a trip $t = (p, f, R_t)$ is much easier when we look at the requests R_t of a fixed relevant combination (p, f). We switched to a set covering (instead of set partitioning) model for two reasons: on the one hand, solving a SCP is in general much easier than solving the equivalent SPP. On the other hand, only *covering* all requests can be guaranteed if only columns corresponding to *efficient* trips are included in the model (1) to (3).

Secondly, the solver for set covering problems has to compute at least a 'good', feasible solution of the SCP. We decided to implement a heuristic algorithm to speed up computation times and to handle even large-scale instances of the MDPDPSH.

3.1 Enumeration of route/vehicle combinations

We first derive a simple criterion for testing whether a combination of a route p and a type of vehicle f is relevant or not. The idea behind this criterion is to check if time compatibility and route compatibility can be fulfilled simultaneously. We will show that this can be done by a polynomial time procedure.

For a given route $p = (v_1, v_2, \ldots, v_{n_p}, v_1)$ we define

$$D_p^{(p)} = \{v_i | v_i \in N^-, 1 \leq i \leq n_p\}$$

and

$$D_d^{(p)} = \{v_i | v_i \in N^+, 1 \leq i \leq n_p\}$$

as the set of all visited depots for pickup and delivery. For $r \in R(p)$ $\mathrm{htw}(r, p, f)$ is the hub time window, which can be written as $[a_r^*, b_r^*]$. We define two sets of times, one for the pickup requests and one for the delivery requests in $R(p)$:

$$T_p^{(p,f)} = \{a_r^* | r \in R(p) \cap R_p\} \quad \text{and} \quad T_d^{(p,f)} = \{b_r^* | r \in R(p) \cap R_d\}$$

In order to ensure that the product set $T_p^{(p,f)} \times T_d^{(p,f)}$ is always non-empty we define $T_p^{(p,f)} = \{-\infty\}$ for delivery routes and $T_d^{(p,f)} = \{\infty\}$ for pickup routes. We can now state the following lemma:

Lemma 1 *Let $c = (p, f)$ be a combination of a route p and a vehicle type f. Combination c is relevant (i.e. time compatibility and route*

compatibility can be fulfilled simultaneously) if and only if the following condition is satisfied

$$\begin{aligned} \exists(t_p, t_d) \quad \in \quad & T_p^{(p,f)} \times T_d^{(p,f)} : \\ & (t_p \leq t_d \\ & \wedge \forall v_i \in D_p^{(p)} : \exists r_i \in R(v_i) : t_p \in \mathrm{htw}(r_i, p, f) \\ & \wedge \forall v_j \in D_d^{(p)} : \exists r_j \in R(v_j) : t_d \in \mathrm{htw}(r_j, p, f)). \end{aligned} \tag{28}$$

The complexity of checking this condition is polynomially bounded by the size of the set of possible requests $|R(p)|$.

Proof: We first proof the validity of the criterion (28): assume that $R_t \subset R(p)$ is time and route compatible. It is now easy to see that $t_p^* := \min_{r \in R_t \cap R_p} a_r^*$ and $t_d^* := \max_{r \in R_t \cap R_d} b_r^*$ fulfill condition (28).
On the other hand, if the pair (t_p, t_d) and requests $r_i \in R(v_i)$ (for each $v_i \in D_p^{(p)}$ or $v_i \in D_d^{(p)}$ respectively) satisfy condition (28), then the set $R_t := \{r_i | v_i \in D_p^{(p)} \cup D_d^{(p)}\}$ is time and route compatible. Therefore, the route p is relevant.
Now we proof the statement on the complexity: let $n = |R(p)|$ be the size of set $R(p)$. Obviously the computation of the hub time windows $\mathrm{htw}(r, p, f)$ for $r \in R(p)$ can be done in $\mathcal{O}(n^2)$ time (only the case with a route of length n_p, $n_p \leq |R(p)|$ makes sense, otherwise route compatibility is violated). There are at the most n^2 pairs (t_p, t_d) with $t_p \leq t_d$ to examine. For each such a pair there are at the most $|R(p)|$ depots $v_i \in D_p^{(p)} \cup D_d^{(p)}$. This is true because the number of depots can be estimated by $|D_p^{(p)} \cup D_d^{(p)}| \leq n_p \leq |R(p)|$. Finally we have to test at most $|R(v_i)| \leq |R(p)|$ requests $r_i \in R(v_i)$. Thus, the total complexity is of order $\mathcal{O}(n^4)$. □

It is also easy to consider the capacity conditions: we only have to take the request $r_i \in R(v_i)$ with $t_p \in \mathrm{htw}(r_i, p, f)$ or $t_d \in \mathrm{htw}(r_i, p, f)$ respectively, which has a minimal weight q_{r_i}. We can sum up all q_{r_i} for pickup requests and delivery requests separately and check these sums against the capacity Q^f. We left this aspect out of lemma 1 for clarity of the presentation.

Algorithm 1: Enumeration of relevant route/vehicle combinations:

1. Start with an empty set $\mathcal{L}$ (labeled) and a set U (unlabeled) of short routes between depot and hub for all available vehicles:

$$\begin{aligned} U &= \{\big((v^-, 0^-, v^-), f\big) \,|\text{for all } v \in N, f \in F_v\} && (29)\\ &\cup\{\big((v^+, 0^-, v^+), f\big) \,|\text{for all } v \in N, f \in F_v\} && (30)\\ &\cup\{\big((v^-, 0^-, 0^+, v^+, v^-), f\big) \,|\text{for all } v \in N, f \in F_v\} && (31) \end{aligned}$$

   The set (29) corresponds to pickup routes, the set (30) to delivery routes, and the set (31) to pickup and delivery routes.

2. Choose any combination $c = (p, f)$ of U and label it, i.e. remove it from U and insert it into the set $\mathcal{L}$ (labeled). If it is possible to extend the route p by one additional new location v (insert v just before hub 0^- if it is different from the locations visited so far or insert v just behind hub 0^+ if it is different from the locations which are visited later).
   All new combinations $\tilde{c} = (\tilde{p}, f)$ which satisfy criterion (28) are inserted into the set U.

3. Repeat step 2 until the set U is empty. Then $\mathcal{L}$ is the set of all relevant route/vehicle combinations.

In general the algorithm always stops after a finite number of iterations because the number of routes is finite. In our specific case where time windows for transportation are relatively narrow this algorithm constructs only short routes.

3.2 Enumeration of (efficient) trips

The idea is to enumerate all relevant assignments of requests $R_t \subset R(p)$ of a given pair (p, f). In subsection 2.4 we discussed in detail what feasibility of a trip $t = (p, f, R_t)$ means, i.e. which properties the set R_t must have. Thus, from a theoretical point of view the model (1) to (3) is well defined. The problem with this model is that it is not possible to enumerate all feasible trips due to their large number in practically relevant instances of the MDPDPSH. But even if it is possible to enumerate them, solving a large SCP remains a difficult task. The concept of efficiency may help to overcome these problems.

First of all we point out that the cost of a trip $t = (p, f, R_t)$ mainly depends on the route p and the vehicle f. The requests R_t only influence the time for waiting at the hub. Because waiting is required if R_t contains pickup requests as well as delivery requests and delivery to the hub has to be performed before pickup.

In general the cost for waiting should be dominated by the other costs. As a consequence every two feasible sets of requests $R_1, R_2 \subseteq R(p)$ lead to trips (p, f, R_1) and (p, f, R_2) with nearly identical costs. We call a trip $t = (p, f, R_t)$ *efficient* if there is no proper superset R^*, $R^* \supset R_t$ such that $t^* = (p, f, R^*)$ is a feasible trip. Efficiency requires putting as many requests $R(p)$ into the vehicle f as possible on its route p. This is completely equivalent to the general concept of efficiency if the costs for waiting at the hub c^f_{wait} are zero. Even if the costs are positive but relatively small, both concepts coincide with one another.

The point is that enumeration of all efficient trips for a given combination (p, f) is a practicable task. For the MDPDPSH only a fraction of all feasible trips is also efficient. Thus, to reduce the size of the model (1) to (3) we only include columns corresponding to efficient trips. This can be seen as a heuristic whenever the costs for waiting at the hub c^f_{wait} are positive.

We present an algorithm to enumerate efficient trips. The procedures are given in a syntax similar to the C or C++ programming language. We assume that two simple functions 'RouteCompatible(p, f, R_t)' and 'Efficient(p, f, R_t)' are available. Function 'RouteCompatible(p, f, R)' returns the value 'TRUE' if the trip (p, f, R_t) satisfies the route compatibility constraint (25) and 'FALSE' otherwise. The function 'Efficient(p, f, R_t)' returns 'TRUE' for efficient trips (p, f, R_t) and 'FALSE' for inefficient trips.

Another function 'ChooseNextCompatible(R_t, C)' returns some request $c \in C$. The idea behind this function is the following. Let $R_t, C \subset R(p)$ be subsets of requests. The set R_t contains the already assigned requests (i.e. R_t is a clique in the time compatibility graph $G^{(p,f)} = (R(p), E^{(p,f)})$), and the set C contains all requests compatible with every element of R_t. Therefore, each set $R_t \cup \{c\}$ for any $c \in C$ is also a clique in $G^{(p,f)}$. The function decides which element of the set C has to be assigned to R_t next. To speed up the computation we implemented the following rule. If $p = (v_1, \ldots, v_{n_p}, v_1)$ is the route, then we distinguish two cases:

1. If for every $1 \leq i \leq n_p$ the set R_t contains at least one element of $R(v_i)$ (i.e. $R(v_i) \cap R_t \neq \emptyset$), then return the request $r \in C$ with minimum index.

2. Otherwise let $1 \leq i^* \leq n_p$ be the minimum index i with $R(v_i) \cap R_t = \emptyset$. Then return the element $c \in C \cap R(v_{i^*})$ with minimum index. (Remark: The set $C \cap R(v_i)$ is not empty if the set $R_t \cup C$ is route compatible.)

Thus, the idea is to return one element belonging to each of the sets $R(v_i)$ at the beginning in order to construct route compatible subsets as fast as possible.

Algorithm 2: Enumeration of requests for a given route/vehicle pair (p, f)

Enumerate(p, f, R_t, C)
(
// Tests for Termination
if not RouteCompatible$(p, f, R_t \cup C)$
  then return
if $C = \emptyset$ *and* Efficient(p, f, R_t)
  then insert R_t *into* $\mathcal{R}^{(p,f)}$
if $C = \emptyset$ *then return*
r^*:=ChooseNextCompatible(R_t, C)
// Two possible decisions:
// a) insert r^* into R_t
if $((r^* \in R_p$ *and* $q(R_t \cap R_p) + q_{r^*} \leq Q^f)$
  or $(r^* \in R_d$ *and* $q(R_t \cap R_d) + q_{r^*} \leq Q^f))$
  then Enumerate$(p, f, R_t \cup \{r^*\}, \{c \in C | c \neq r^* \wedge \forall r \in R_t : c \overset{(p,f)}{\sim} r\})$
// b) do not insert r^* into R_t
Enumerate$(R_t, C \setminus \{r^*\})$
)

The procedure 'Enumerate' starts with R_t as the empty set and C including all possible requests which can be transported, i.e. have a true hub time window:

Enumerate$(\emptyset, \{r \in R(p) | \text{htw}(r, p, f) \neq \emptyset\})$

The result of the procedure 'Enumerate' for a given pair (p, f) is the set $\mathcal{R}^{(p,f)}$ containing the sets of requests R_t which lead to efficient trips (p, f, R_t).

Example 2 For the data of example 1 only three different efficient trips $t = (p, f, R_t)$ exist, namely $R_{t_1} = \{2, 3, 4, 6\}$, $R_{t_2} = \{2, 4, 6, 7\}$ or $R_{t_3} = \{2, 4, 5, 8\}$. The algorithm presented here needs 26 recursive calls to enumerate these three trips. We want to point out that the number of efficient trips is small in comparison to the number of 22 feasible trips. Computational studies indicate that this is also true on average.

In addition to the ideas elaborated above we decided to include some dominance checks. If two different types of vehicles $f_1, f_2 \in F_v$ are available at depot v, vehicle f_1 dominates f_2 if f_1 has higher costs than f_2 while capacity and driving times of f_1 are at the most as large as those of f_2. In these cases we do not construct any route/vehicle combinations using vehicle f_2 and starting at v. In most cases it is not possible to exclude vehicle type f_2 from all trips because the dominating vehicle type f_1 is not available at all depots.

Another helpful criterion takes advantage of the fact that a vehicle f_1 with higher capacity than f_2 should normally be more expensive and slower. If a set of requests R_t is computed for the combination (p, f_1), but R_t fits into the smaller vehicle f_2, we do not construct the trip (p, f_1, R_t).

Finally we do not include any pickup and delivery trip $t = (p, f, R_t)$ (i.e. a trip with $R_t \cap R_p \neq \emptyset$ and $R_t \cap R_d \neq \emptyset$) which can be replaced by a cheaper combination of a pickup trip and a delivery trip. To elaborate this idea we construct from the route $p = (v_1, \ldots, v_k, 0^-, 0^+, v_{k+3}, \ldots v_{n_p}, v_1)$ the corresponding pickup route $p_1 = (v_1, \ldots, v_k, 0^-, v_1)$ and the corresponding delivery route $p_2 = (v_{n_p}, 0^+, v_{k+3}, \ldots, v_{np})$. If the combined costs of the trips $t_1 = (p_1, f, R_t \cap R_p)$ and $t_2 = (p_2, f, R_t \cap R_d)$ are less than the costs of t, then including t into the model (1) to (3) makes no sense.

3.3 The set covering solver

The set covering problem (SCP) belongs to the class of $\mathcal{NP}$-complete problems [21]. Many exact and heuristic algorithms have been pub-

lished the last twenty years. Most exact algorithms (see e.g. [4, 8, 11]) use Lagrangian heuristics and subgradient optimization embedded into a branch and bound environment. Heuristic algorithms mainly split into two classes: some use principles similar to the exact methods but incorporate heuristic ideas for computing good, feasible solutions and to fix variables [9, 14, 15, 24]. Other methods [10, 20] are based on local search controlled by genetic algorithms as a meta-heuristic.

As far as we know, the most successful heuristic for the SCP up to now was published in 1996 by A. Caprara, M. Fischetti and P. Toth [14, 15]. It outperforms other heuristic methods in computation speed and solution quality. We decided to implement this method with some small modifications [25].

We were able to solve 48 of 65 of the test problems from the OR-library [7] to the optimum (or to the best known solution up to now) on average two to three times faster than reported in [14, 15].

3.4 Overview of the algorithm

We now present a brief overview of the algorithm:

1. Model generation:
   (a) Enumeration of relevant route/vehicle combinations: Algorithm 1 determines the set $\mathcal{L}$ of all relevant route/vehicle combinations $c = (p, f)$.
   (b) Enumeration of relevant trips: For each route/vehicle combination $c = (p, f) \in \mathcal{L}$ call Algorithm 2. Algorithm 2 determines the set $\mathcal{R}^{(p,f)}$ of efficient request subsets. Construct the set of trips
   $$T = \{t = (p, f, R_t) | (p, f) \in \mathcal{L}, R_t \in \mathcal{R}^{(p,f)}\}.$$
   (c) Perform dominance checks on the set T as explained in section 3.2. Remove all dominated trips from the set T.
2. Set covering solver:
   Solve the SCP (1) to (3) with the corresponding set of trips T.
3. Postprocessing:
   Assign multiple covered requests to exactly one trip: In some cases requests are covered by more than one trip $t \in T^*$ of a

solution $T^* \subset T$ of the SCP (1) to (3) (that does not mean that any of the trips T^* is redundant).

4 Computational Results

The algorithm described above has been implemented as one subsystem in the decision support system ISLT (see [22, 23] for details). ISLT is successfully used by planners at Deutsche Post AG´s headquarters in Bonn.

The main objective of the computational tests was to evaluate the influence of the parameters number of requests $|R|$, number of depots $|N|$, and number of vehicle types $|F|$ over the size of the constructed SCP model. In order to judge the quality of (at least some) solved instances we first focus on the determination of a lower bound for the MDPDPSH.

4.1 Computation of Lower Bounds

The idea is to relax the problem by allowing to use fractional numbers of vehicles. Obviously in this case it makes no sense to use other routes than the direct ones between depots and hub. Consequently, the problem splits into $|N|$ single problems for transportation between each of the depots and the hub (we assume that the triangle inequality is valid for costs and times, i.e. $c^f_{ij} + c^f_{jk} \geq c^f_{ik}$ and $t^f_{ij} + t^f_{jk} \geq t^f_{ik}$ for all arcs $(ij), (jk), (ik) \in A^f$).

The only decision we have to make is which type of vehicle we use for a specific request and which pickup requests are transported by the same vehicle with which delivery requests. The second question is important because we always have the alternative to transport pickup and delivery requests by the same vehicle, or to transport them in two vehicles which are empty on one way between depot and hub. The costs for waiting at the hub compete with the costs for using empty vehicles on the way to the hub or back from the hub. This aspect becomes more obvious when we consider the costs. For the moment we assume that we use some fixed vehicle $f \in F$. The costs for transporting one unit of a pickup request $r \in R_p$ by a trip which does only pickups is

$$c^f_{r,X} = \frac{c_{v(r),0^-} + c_{0^-,v(r)}}{Q^f}.$$

Similarly the costs for transporting one unit of a delivery request $r \in R_d$ by a trip which does only performs deliveries is

$$c^f_{Y,r} = \frac{c_{v(r),0^+} + c_{0^+,v(r)}}{Q^f}.$$

If a pickup request $r \in R_p$ and a delivery request $r' \in R_d$ of the same depot (i.e. $v(r) = v^-$, $v(r') = v^+$ for some $v \in N$) are transported together, additional costs for waiting at the hub may occur. Consequently, the costs of transporting one unit of r and one unit of r' by the same vehicle of type f are

$$c^f_{r,r'} = \frac{c_{v(r),0^-} + c_{0^+,v(r')} + c^f_{wait}(\min\{0, a_{r'} - b_r\})}{Q^f}.$$

In cases where no common transport of r and r' is possible (i.e. $a_r > b_{r'}$) we set $c^f_{r,r'} = \infty$.

The question of choosing a type of vehicle is now easy to decide. We have to choose from the set of available vehicles $F_{v(r)}$ the one with the lowest costs. In addition, this vehicle must be able to transport the request (i.e. must have a non-empty hub time window). The corresponding minimal costs are $c^*_{r,X}$ for $r \in R_p$, $c^*_{Y,r}$ for $r \in R_d$, and $c^*_{r,r'}$ for $r \in R_p, r' \in R_d$.

It is easy to see that a lower bound LB_v for the costs of transportation between depot $v \in N$ and the hub is given by the value of the following classical transportation problem (TP, see e.g. [27]). The TP is defined by supply nodes $S_v = R(v^-) \cup \{Y\}$ and demand nodes $D_v = R(v^+) \cup \{X\}$. With the definitions $q_X = \sum_{r \in R(v-)} q_r$, $q_Y = \sum_{r \in R(v^+)} q_r$, and $c_{Y,X} = 0$, all quantities q_s, q_d and costs $c^*_{s,d}$ for $s \in S, d \in D$ are well-defined.

Lemma 2 *A lower bound of the MDPDPSH is given by*

$$LB = \sum_{v \in N} LB_v.$$

The proof of this lemma is left to the reader.

4.2 Test Instances

We used data of six different hubs labeled from A to F. Each hub determines a set of requests R and a set of depots N. Table 2 describes these

hub locations. The quantity of letters to be transported is measured by the corresponding number of letter transport boxes (LTB, one LTB is about 4.5 kg).

Table 2 about here

Five different vehicle fleet scenarios are used to analyze the influence of increasing the number of vehicle types. Scenario 1 and scenario 2 consider only small but quick transporters. Scenarios 3 to 5 also allow more and more larger but slower lorries. Details are given in table 3.

Table 3 about here

Test instances have been generated by combining each hub with each of the vehicle fleet scenarios. Therefore, table 4 contains for each hub A to F and each vehicle fleet scenario 1 to 5 some characteristic information:

route/vehicle comb.	This row contains the number of different relevant route/vehicle combinations $\|\mathcal{L}\|$.
relevant trips	The number of different relevant trips found by the enumeration procedure is given here. This number is also the number of columns $\|T\|$ in the set covering problem (1) to (3).
trips in solution	The number of trips in the heuristic solution of the MDPDPSH is displayed here.
overall cost	This row contains the cost of the heuristic solution, i.e. the sum of the cost of all trips used.
lower bound	This is the lower bound of the specific instance computed according to lemma 2.
time model	The computation time for the construction of the model is given here. This contains the time for the enumeration of relevant route/vehicle pair and the enumeration of efficient trips. All computational tests were performed on a 300 MHz AMD K6 Personal Computer under Windows NT 4.0 with 128 MB RAM. The algorithm was programmed in C++ and compiled using the Microsoft Visual C++ compiler, version 5.0. The compiler target option was set to 'release'.
time SCP solver	This is the time for solving the corresponding set covering problem.

time SCP best sol. The set covering heuristic finds good heuristic solutions in a fraction of the overall computation time. This row shows the time needed to compute the best solution output at the end.

Table 4 about here

We were able to solve all of the 30 test instances of the MDPDPSH. In order to keep the number of route/vehicle combinations small we implemented some additional rule for the 10 large-scale instances of hub E and hub F. All routes were only allowed to have at most two pickup locations and at most two delivery locations. In addition to this we decided to include at most 1000 relevant trips in the SCP for each route/vehicle combination. Whenever more than 1000 trips are found we randomly choose 1000 different ones. SCP ranging from 34 rows and 242 columns to 293 rows and 357797 columns.

Computation times for the instances of hub A and B are below five seconds, for instances of hub C and D below six minutes, and for instances of hub E and F below 150 minutes.

The number of feasible, efficient trips primarily grows with the number of requests. The quotient 'number of trips' to 'number of requests' rises from about 7 for the smallest instance of hub A to about 1200 for the large-scale instance of hub F. Nevertheless, the absolute number of feasible, efficient trips remains in an algorithmically tractable area. Capara, Fischetti, and Toth report on large-scale SCP instances with more than one million columns solved by their implementation of the algorithm [14, 15]. But due to the complexity of the SCP solution times of the algorithm strongly increase with the number of columns in model (1) to (3).

Unfortunately, the lower bounds computed by the relaxation described in section 4.1 are in general weak. The ratio of 'overall costs' to the computed 'lower bound' is between 1.36 and 2.79. Only for some large-scale instances (e.g. F1, F2, F3) these bounds are tight, thus, computed solutions can be judged as 'good' solutions.

Another important aspect becomes visible by comparing the number of different vehicle types $|F|$ with the number of feasible trips $|T|$ in the model. Due to the dominance criteria elaborated in section 3.2 the number of relevant route/vehicle combinations grows sub-linearly in the number of different vehicle types $|F|$ and the maximal capacity Q^f (i.e. $|T| < \mathcal{O}\left(|F| \cdot \max_{f \in F} Q^f\right)$). Thus, analyzing scenarios with a more diverse vehicle fleet may be possible.

Concerning computational times a further result is that the time for model construction is always less than the time for solving the SCP. Therefore, the bottleneck of the entire algorithm seems to be solving the constructed SCP model. But one has to keep in mind that 'good' feasible solutions are already available after a small fraction of time before the SCP solver terminates. Nevertheless, a solution time below 150 minutes even for the largest instances seems acceptable.

From the practical point of view (i.e. of the planners at the Deutsche Post AG) integration of the MDPDPSH into the decision support system ISLT has resulted in a powerful tool to support planners in redesigning their ground feeding networks. Compared to former manually planned solutions they were able to reduce costs on average by about 15%. In some special cases improvements of up to 30% could be achieved.

5 Future Research Activities

The main problem in judging the results presented in the previous section is that lower bounds based on the relaxation of section 4.1 are weak. As outlined before, a column generation approach is supposed to produce stronger lower bounds as many publications in the vehicle routing and crew scheduling area show. The strength of column generation lower bounds is not only supported by computational studies but also by theoretical results (see e.g. [12]). In this section we follow [5] in respect to terms and concepts.

We propose column generation for the MDPSPSH according to the decomposition approach on route/vehicle pairs. More precisely, we start from the restricted master program (RMP) of the set partitioning formulation (1) to (3) (i.e. the LP-relaxation with only some feasible columns $T' \subset T$). The pricing problem for route/vehicle combination (p, f) has to assign requests $R_t \subset R(p)$ to this combination (p, f). Let $\pi_r, r \in R$ be the dual prices of the constraints (2) of the RMP. The pricing problem of combination $c = (p, f) \in \mathcal{L}$ has the feasibility region (22) to (27). Its objective function is given by

$$\min \tilde{c}_{(p,f)} = c_{(p,f)} + c^f_{wait} \cdot t_{wait} - \sum_{r \in R(p)} \pi_r z_r \tag{32}$$

with $c_{(p,f)} = \sum_{i=1}^{n_p} c^f_{v_i v_{i+1}}$ the costs of using vehicle f on the route p,

and t_{wait} the time for waiting at the hub. The waiting time t_{wait} can be determined by

$$z_r = 1 \wedge z_{r'} = 1 \Longrightarrow t_{wait} \geq a_{r'} - b_r$$
$$\text{for all } r \in R(p) \cap R_p \text{ and all } r' \in R(p) \cap R_d \qquad (33)$$
$$t_{wait} \geq 0 \qquad (34)$$

Consequently the pricing problem is given by (32), (22) to (27), (33) and (34). For pure pickup or pure delivery trips the waiting time t_{wait} is always zero, thus, restrictions (33) and (34) can be removed. The costs $-\pi_r$ of elements $r \in R(p)$ are the weights of the clique nodes as well as the profits of the items in the knapsack.

Pricing problems which determine shortest paths under side constraints are usually solved by dynamic programming. For the GPDP, for example, recent dynamic programming algorithms are only able to handle corresponding shortest path problems of moderate size (about 50 transportation requests and small loads q_r and capacities Q^f, see [32]). In contrast to this we believe that branch and bound algorithms are more appropriate in our case of a combined clique/knapsack problem since most exact solution approaches for maximum (weighted) cliques as well as knapsack problems use branch and bound. Recent algorithms for these problems [2, 3, 26] are able to solve instances with a few hundred nodes or thousands of items respectively.

The question is how to handle a combined clique/knapsack problem algorithmically. On the one hand, it is possible to use algorithms for the maximum weighted clique problem and to view the knapsack constraints as side constraints, which have to be considered in addition. On the other hand, there exist many successful algorithms for the knapsack problem and it may be possible to incorporate the clique constraints into one of them in an easy manner.

Finally, we give some remarks on a branching strategy for a branch-and-price algorithm for the MDPDPSH. Branching is required when the RMP is solved to the optimum, no new columns price out, and the optimal solution of the RMP is not integral. According to the branching rule due to Ryan and Foster [28, 5], two requests $r, s \in R$ have either to be assigned to the same trip or to different trips. More precisely, if $\lambda = (\lambda_t)_{t \in T'}$ is a fractional solution of the RMP (i.e. at least one component is fractional), then there exist two different requests

$s, r \in R$ with

$$0 < \sum_{t \in T' : z_r^t = 1, z_s^t = 1} \lambda_t < 1.$$

Consequently branching on the two subsets

$$T_1 = \{t \in T | z_r^t = z_s^t = 1 \vee z_r^t = z_s^t = 0\}$$

and

$$T_2 = \{t \in T | z_s^t + z_r^t \leq 1\}$$

can be done. The branch of set T_1 requires that requests r and s are assigned to the same trip. In the pricing problem this can be easily achieved by replacing the corresponding nodes of r and s by a new common node (only route/vehicle combinations with $r \overset{(p,f)}{\sim} s$ have to be considered). In the branch of set T_2 a trip is only allowed to transport one of the requests s and r, or none of them. This is also compatible with the pricing problem when we define s and r as non-compatible nodes $r \overset{(p,f)}{\not\sim} s$. This branching strategy is, therefore, compatible with the structure of the pricing problem.

6 Conclusions

This paper has introduced a special type of pickup and delivery problems, which has important applications in several areas of transportation. Many transportation systems are configured so that a number of scheduled regular services provide the backbone of the system. This implies that feeding operations are necessary in order to transport commodities to and from the entry points (hubs) of the backbone system. For example, in the LTL motor carrier industry with regular scheduled services these hubs are the end-of-line terminals. In public transport these may be intercity bus or train stations and in the airline industry they are central air hubs. This shows that the postal system we have considered is one of many possible applications of this problem.

A common feature of pickup-and-delivery routes in such systems is the small number of stops on the route to and from the hub. This often makes it possible to enumerate all possible routes of the feeding vehicles. On the other hand, there are usually many commodities with possibly different time windows, which have to be transported between

the hub and the local service points. Instead of making copies of the pickup or delivery point for each request as in the usual approaches we have chosen to exploit the special structure of the problem. Mathematically, this leads to subproblems which combine features of clique, knapsack, and some additional constraints. Using this approach, which is integrated into a decision support system for planning of transportation in mail delivery, we have been able to solve fairly large real-world instances. A comparison with existing solutions showed the potential for substantial gains.

The results we presented here suggest several promising paths for future research. First, the lower bounds we derived from the transportation problems are rather poor due to their simplicity. Secondly, we have shown that the core model we have used can be conveniently integrated into a column generation/branch-and-price algorithm. We believe that such an approach will be able to solve at least medium-size instances to optimality and provide good lower bounds. This will allow us to judge the quality of the heuristic solutions more precisely.

Acknowledgements

The author is grateful to the Deutsche Post AG in particular to the director of the transportation department J. Weith for the possibility to engage in this interesting project and to the project-leader M. Katz for constructive collaboration. Special thanks to Tore Grünert for many fruitful discussions and some remarks and corrections on early versions of this paper.

References

[1] Appelgren, L.H. (1969), "A column generation algorithm for the ship scheduling algorithm", *Transportation Science* **3**, 53-68.

[2] Babel, L. (1991), "Finding Maximum Cliques in Arbitrary and in Special Graphs", *Computing* **46**, 321-341.

[3] Babel, L., and Tinhofer, G. (1990), "A Branch and Bound Algorithm for the Maximum Clique Problem", *ZOR - Methods and Models of Operations Research* **34**, 207-217.

[4] Balas, E., and Ho, A. (1980), "Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study", *Mathematical Programming Study* **12**, 37-60.

[5] Barnhart, C., and Johnson, E.L., and Nemhauser, G.L., and Savelsbergh, M.W.P., and Vance, P.H. (1998), "Brach-and-price: column generation for solving huge integer programs", *Operations Research* **46**, 316-329.

[6] Barnhart, C., and Scheur, R.R. (1996), "Air network design for express shipment service", *Operations Research* **44**, 852-863.

[7] Beasley, J. (1990), "OR-Library: distributing test problems by electronic mail", *Journal of the Operational Research Society* **40**, 1069-1072.

[8] Beasley, J. (1987), "An algorithm for the set covering algorithm", *European Journal of Operational Research* **31**, 85-93.

[9] Beasley, J. (1990), "A Lagrangian heuristic for the set-covering problem", *Naval Research Logistics* **37**, 151-164.

[10] Beasley, J., and Chu, P.C. (1996), "A genetic algorithm for the set covering problem", *European Journal of Operational Research* **94**, 392-404.

[11] Beasley, J., and Jornsten, K. (1992), "Enhancing an algorithm for the set covering problem", *Journal of Operational Research* **58**, 293-300.

[12] Bramel, J., and Simchi-Levi, D. (1997), "On the effectiveness of set covering formulations for the vehicle routing problem with time windows", *Operations Research* **45**, 295-301.

[13] Büdenbender, K., and Grünert, T., and Sebastian, H.-J. (1998), A tabu search algorithm for the direct flight network design problem, Working Paper 98/14, Department of Operations Research, Rheinisch-Westfälische Technische Hochschule Aachen, 1998.

[14] Caprara, A., and Fischetti, M., and Toth, P., "A heuristic algorithm for the set covering problem", in: W.H. Cunningham, and S.T. McCormick, and M. Queyranne (eds.), *Proc. 5th IPCO*, Springer LNCS 1084, 1996, 72-81.

[15] Caprara, A., and Fischetti, M., and Toth, P. (1996), "A heuristic algorithm for the set covering problem", Working Paper, DEIS, University of Bologna, Italy.

[16] Desrosiers, J., and Soumis, F., and Desrochers, M. (1984), "Routing with time windows by column generation", *Networks* **14**, 545-565.

[17] Desaulniers, G., and Desrosiers, J., and Ioachim, I., and Solomon, M.M., and Soumis F., and Villeneuve, D., "A unified framework for deterministic time constrained vehicle routing and crew scheduling problems", in: G. Crainic and G. Laporte (eds.), *Fleet Management and Logistics*, Kluwer, Boston, 1998, 57-93.

[18] Desaulniers, G., and Desrosiers, J., and Gamache, M., and Soumis, F., "Crew scheduling in air transportation", in: G. Crainic and G. Laporte (eds.), *Fleet Management and Logistics*, Kluwer, Boston, 1998, 169-185.

[19] Dumas, Y., and Desrosiers, J., and Soumis, F. (1991), "The pickup and delivery problem with time windows", *European Journal of Operational Research* **54**, 7-22.

[20] Eremeev, A. (1998), "A genetic algorithm with a non-binary representation for the set covering problem", working paper, Omsk Branch of Sobolev Institute of Mathematics, RAS.

[21] Garey, M., and Johnson, D., *Computers and Intractibility: A Guide to NP-Completeness*, W. Freemand and Co., New York, 1979.

[22] Grünert, T., and Sebastian, H.-J., and Thärigen, M. (1999), "The design of a letter-mail transportation network by intelligent techniques", *Proceedings of the Hawai'i International Conference On System Sciences*, to appear.

[23] Grünert, T., and Sebastian, H.-J., "Planning models for long-haul operations of postal and express shipment companies", Working Paper, Department of Operations Research, Rheinisch-Westfälische Technische Hochschule Aachen, 1998.

[24] Haddadi, S. (1997), "A simple Lagragean heuristic for the set covering problem", *European Journal of Operational Research* **97**, 200-204.

[25] Irnich, S., "Modellierungs eines Tourenplanungsprobems als Set Covering Problem und Implementierung eines heuristischen Lösungsverfahrens", Internal Paper, Elite Foundation, 1998.

[26] Martello, S., and Toth, P., *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, Chichester, England, 1990.

[27] Nemhauser, G.L., and Wolsey, L.A., *Integer and Combinatorial Optimization*, Wiley, New York, 1988.

[28] Ryan, D.M., and Foster, B.A., "An integer programming approach to scheduling", in: *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, North-Holland, Amsterdam, 1981, 269-280.

[29] Ribeiro, C., and Soumis, F. (1994), "A column generation approach to the multiple-depot vehicle scheduling problem", *Operation Research* **42**, 41-52.

[30] Savelsbergh, M.W.P. (1997), "A branch-and-price algorithm for the generalized assignment problem", *Operations Research* **45**, 831-841.

[31] Savelsbergh, M.W.P., and Sol, M. (1995), "The general pickup and delivery problem", *Transportation Science* **29**, 17-29.

[32] Savelsbergh, M.W.P., and Sol, M. (1998), "DRIVE: Dynamic routing of independent vehicles", *Operations Research* **46**, 474-490.

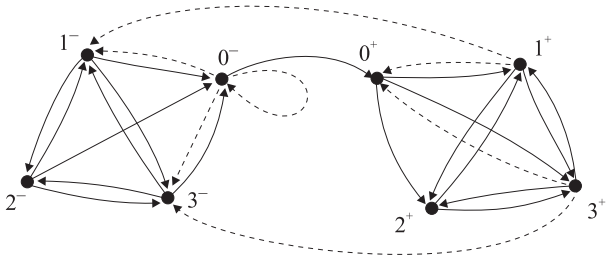

Figure 1: Network (V, A^f) for a three depot problem and a vehicle f available at depot 1 and 3. Backward arcs are displayed as dotted arcs.

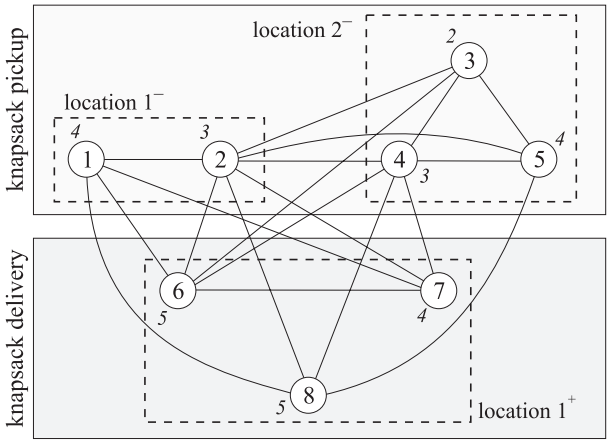

Figure 2: Combined clique, knapsack problem with additional constraints

request r	type	location $v(r)$	time window $[a_r, b_r]$	quantity q_r	hub time window $\mathrm{htw}(r, p, f)$
1	pickup	1^-	$[01:50, 08:00]$	4	$[04:20, 08:00]$
2	pickup	1^-	$[05:10, 11:00]$	3	$[07:40, 11:00]$
3	pickup	2^-	$[07:30, 10:00]$	2	$[09:10, 10:00]$
4	pickup	2^-	$[06:40, 12:40]$	3	$[08:20, 12:40]$
5	pickup	2^-	$[08:10, 10:20]$	4	$[09:50, 10:20]$
6	delivery	1^+	$[08:10, 11:20]$	5	$[08:10, 09:20]$
7	delivery	1^+	$[08:10, 10:30]$	4	$[08:10, 08:30]$
8	delivery	1^+	$[10:10, 16:20]$	5	$[10:10, 14:20]$

Table 1: Example for a route/vehicle combination with corresponding requests and computing hub time windows

	hub A	hub B	hub C	hub D	hub E	hub F
locations N	6	6	8	8	13	22
pickup requests R_p	13	26	40	57	102	130
quantity $\sum_{r \in R_p} q_r$ $[LTB]$	2136	4139	5565	8767	12702	10153
delivery requests R_d	21	23	43	57	55	112
quantity $\sum_{r \in R_d} q_r$ $[LTB]$	3683	3518	5831	9346	6744	8102

Table 2: Hubs with specific data

	scenario 1	scenario 2	scenario 3	scenario 4	scenario 5
vehicle types F	1	2	3	4	6
smallest capacity Q^f $[LTB]$	320	200	200	200	200
largest capacity Q^f $[LTB]$	320	320	400	640	1040

Table 3: Vehicle fleet scenarios

instance	property	scenario 1	scenario 2	scenario 3	scenario 4	scenario 5
	route/vehicle comb.	89	178	250	322	466
	relevant trips	242	266	408	1001	2204
	trips in solution	27	28	17	13	10
A	overall cost	3471	3274	2798	2269	2017
	lower bound	1822	1822	1643	1251	930
	time model	0.12	0.09	0.33	0.54	1.00
	time SCP best sol.	0.21	0.05	0.04	0.07	0.19
	time SCP solver	0.21	0.23	1.50	1.80	1.52
	route/vehicle comb.	34	68	92	116	164
	relevant trips	389	456	617	821	1074
	trips in solution	32	32	22	17	14
B	overall cost	5051	4682	4025	3600	3357
	lower bound	2125	2125	1932	1531	1202
	time model	0.09	0.13	0.18	0.27	0.41
	time SCP best sol.	1.03	1.33	0.01	0.01	0.01
	time SCP solver	1.13	1.37	1.71	2.29	2.68
	route/vehicle comb.	158	316	431	546	776
	relevant trips	2680	3075	4753	9239	16339
	trips in solution	34	34	27	22	17
C	overall cost	7792	7703	7165	6889	6473
	lower bound	4656	4656	4316	3612	3030
	time model	0.7	1.0	1.5	3.4	11.1
	time SCP best sol.	6.1	6.8	1.3	9.1	62.3
	time SCP solver	19.5	10.9	16.5	63.4	81.0
	route/vehicle comb.	70	140	193	246	352
	relevant trips	1679	1884	4222	12439	42375
	trips in solution	72	74	51	39	33
D	overall cost	12668	11994	10200	9219	8637
	lower bound	5784	5784	5364	4508	3802
	time model	0.8	1.2	2.4	13.8	107.5
	time SCP best sol.	1.3	1.2	8.6	24.9	90.8
	time SCP solver	10.3	19.5	29.6	205.1	263.7
	route/vehicle comb.	208	416	614	812	1208
	relevant trips	33522	38281	85961	153832	253993
	trips in solution	56	59	46	28	19
E	overall cost	13685	13685	11446	8598	6602
	lower bound	8653	8653	7773	5564	3925
	time model	19.0	21.9	62.8	283.9	1268.2
	time SCP best sol.	1077.1	46.5	1805.0	1269.5	1067.7
	time SCP solver	1439.0	1186.8	2022.2	2003.4	2087.9
	route/vehicle comb.	369	738	1053	1368	1998
	relevant trips	100603	152971	244514	326378	357797
	trips in solution	39	38	34	26	19
F	overall cost	14598	14399	13676	11394	8749
	lower bound	10621	10621	9635	7322	5403
	time model	405.6	479.3	810.2	1811.1	2905.6
	time SCP best sol.	1269.0	788.1	3349.5	3232.5	1851.9
	time SCP solver	4126.6	6108.8	7102.0	7103.1	3347.4

Table 4: Computational result