Solving Large-Scale Multiple-Depot Vehicle Scheduling Problems

Andreas Löbel

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Takustraße 7,
D-14195 Berlin, Germany, E-mail: loebel@zib.de, URL: www.zib.de

Abstract: This paper presents an integer linear programming approach with column generation for the $\mathcal{NP}$-hard Multiple-Depot Vehicle Scheduling Problem (MDVSP) in public mass transit. We describe in detail the basic ingredients of our approach that seem indispensable to solve truly large-scale problems to optimality, and we report on computational investigations that are based on real-world instances of three large German public transportation companies. These instances have up to 25 thousand timetabled trips and 70 million integer decision variables.

Compared to the results obtained with one of the best planning system currently available in practice, our test runs indicate savings of several vehicles and a cost reduction of about 10 %. Parts of our presented implementations are already integrated in the planning systems BERTA of the Berliner Verkehrsbetriebe and MICROBUS II of the IVU Gesellschaft für Informatik, Verkehrs- und Umweltplanung mbH, Berlin. Moreover, this system has also been purchased by the research department of the SIEMENS AG, Munich.

1 Introduction

Solving transportation problems was and still is one of the driving forces behind the development of mathematical disciplines such as optimization and operations research (see Borndörfer/Grötschel/Löbel (1995)). Truly large transportation problems have to be solved, for instance, in airline traffic (airline and crew scheduling) and public mass transit (vehicle and duty scheduling). In the past, the corresponding transportation markets have often been protected by monopolistic structures. However, deregulation of such monopolistic markets has led to a world-wide competition. It is therefore obvious that competitive participants in these markets must use computer-aided tools for their operational planning process to employ their resources as efficiently as possible. Modern and sophisticated mathematical optimization techniques can help to solve such planning problems.

For instance, public transportation in the European Community is subject to such market deregulation. Monopolistic markets have become more liberal or will soon be broken up. In order to prevent their complete extinction from the market, monopolistic transportation companies will therefore have

to change from deficit-oriented monopolies to competitive market players. One important factor in facing the challenges of a competitive market is, of course, cost reduction, which can be obtained by making intelligent use of the latest mathematical knowhow.

Vehicle scheduling is one important step in the hierarchical planning process in public transportation. The *Multiple-Depot Vehicle Scheduling Problem* (MDVSP) is to assign a fleet of vehicles, possibly stationed at several garages, to a given set of passenger trips such that operational, company-specific, technical, and further side constraints are satisfied and the available resources are employed as efficiently as possible. In the last three decades, considerable research has gone into the development of academic as well as practice-oriented solution techniques for the $\mathcal{NP}$-hard MDVSP and special, often polynomially solvable cases of it. Review articles on this topic are, for instance, Desrosiers/Dumas/Solomon/Soumis (1995), Daduna/Paixão (1995), and Bussieck/Winter/Zimmermann (1997).

The most successful solution approaches for the MDVSP are based on *network flow models* and their integer programming analogues. In the literature, there are two basic mathematical models of this type: First, a direct *arc-oriented model* leading to a *multicommodity flow problem* and, second, a *path-oriented model* leading to a *set partitioning problem*. The latter can also be derived from Dantzig-Wolfe decomposition applied to the first. Both approaches lead to large-scale integer programs, and *column generation techniques* are required to solve their LP relaxations. We shall explicitly discuss the differences between these two models in Section 3.

We investigate in this paper the solution of the *multicommodity flow formulation*. Solution techniques for models of this flavour have been discussed in various articles: Carpaneto/Dell'Amico/Fischetti/Toth (1989) describe an integer LP (ILP) formulation based on an arc-oriented assignment problem with additional path-oriented flow conservation constraints. They apply a so-called "additive lower bounding" procedure to obtain a lower bound for their ILP formulation. Ribeiro/Soumis (1994) show that this additive lower bounding is a special case of Lagrangean relaxation and its corresponding subgradient method. Forbes/Holt/Watts (1994) solve the integer linear programming formulation of the multicommodity flow model by branch-and-bound. The sizes of the problems that have been solved to optimality in these publications are relatively small involving up to 600 timetabled trips and 3 depots.

The *contribution of this paper* is the efficient solution of the ILP (derived from the multicommodity flow formulation) by means of LP column generation techniques. We use a new technique, called *Lagrangean pricing*, that is based on Lagrangean relaxations of the multicommodity flow model. Embedded within a branch-and-cut frame, this method makes it possible to solve problems from practice to *proven* optimality. Lagrangean pricing has been developed independently at the same time by Fischetti/Vigo (1996)

and Fischetti/Toth (1996) for solving the Asymmetric Travelling Salesman Problem and the Resource-Constrained Arborescence Problem.

Our computational investigations are performed on large-scale data from three German public transportation companies: the Berliner Verkehrsbetriebe (BVG), the Hamburger Hochbahn AG (HHA), and the Verkehrsbetriebe Hamburg-Holstein AG (VHH). These instances involve problems with up to 49 depots, about 25 thousand timetabled trips, and about 70 million integer decision variables. These test instances have been provided by our partners HanseCom GmbH, Hamburg, and IVU Gesellschaft für Informatik, Verkehrs- und Umweltplanung mbH (IVU), Berlin. The test runs on this test set show that our method is able to solve problems of this size optimally. These problems are orders of magnitude larger than the instances successfully solved with other approaches, as far as we know.

In the following, we present our branch-and-cut approach with column generation. We start in the next section by describing our multicommodity flow version of the problem and present an ILP formulation and relaxations thereof. In Sect. 3 follows a discussion of the model, e. g., we compare it with problem relaxations often used in practice. Our algorithm is described in Sect. 4 presenting various tools to solve MDVSP problems and subproblems such as primal heuristics, a network simplex algorithm with column generation (Löbel (1996)), Lagrangean relaxations (Kokott/Löbel (1996)), linear programming relaxations (Löbel (1997d)), and the branch-and-cut approach composing all these ingredients. The tested real-world data are presented in Sect. 5, and the computational results are discussed in Sect. 6. A comprehensive report about this project can be found in our thesis (see Löbel (1997c)).

2 The MDVSP

The following section refers to some basic terminology for MDVSPs that we quickly resume here. For more details see Löbel (1997c).

The fleet of a transportation company is subdivided into **depots**. The set of depots is denoted by $\mathcal{D}$. With each depot $d \in \mathcal{D}$, we associate a start point d^+ and an end point d^- where its vehicles start and terminate their daily duty. Let $\mathcal{D}^+ := \{d^+ |\ d \in \mathcal{D}\}$ and $\mathcal{D}^- := \{d^- |\ d \in \mathcal{D}\}$. The number of available vehicles, the **depot capacity**, of each depot d is denoted by κ_d. A given timetable defines a set of **timetabled trips**, denoted by $\mathcal{T}$, that are used to carry passengers. We associate with each $t \in \mathcal{T}$ a first stop t^-, a last stop t^+, a departure time s_t, an arrival time e_t, and a **depot-group** $G(t) \subseteq \mathcal{D}$. Each $G(t)$ includes those depots whose vehicles are allowed and able to service trip t. Let $\mathcal{T}_d := \{t \in \mathcal{T} |\ d \in G(t)\}$, $\mathcal{T}^- := \{t^- |\ t \in \mathcal{T}\}$, and $\mathcal{T}^+ := \{t^+ |\ t \in \mathcal{T}\}$.

There are further types of trips, which do not carry passengers: A **pull-out trip** connects a start point d^+ with a first stop t^-, a **pull-in trip** connects a last stop t^+ with an end point d^-, and a **dead-head trip** connects

a last stop t^+ with a succeeding first stop t'^-. For simplicity, these trips are all called **unloaded trips**.

For two trips $p, q \in \mathcal{T}$, let $\Delta_{p,q} \geqslant 0$ be given. In the literature, $\Delta_{p,q}$ is often be used as the duration (travel plus layover time) from the last stop p^+ to the first stop q^- (e. g., see Dell'Amico/Fischetti/Toth (1993), Ribeiro/Soumis (1994), and Daduna/Paixão (1995). However, our operating partners use such a definition of $\Delta_{p,q}$ only for those dead-head trips for which the idle time or the difference $s_q - e_p$ does not exceed a predefined maximum ranging from 40 to 120 minutes. Otherwise, $\Delta_{p,q}$ is set to infinity. We will show that such a restriction in the degree of freedom can lead to a higher vehicle demand and, therefore, to suboptimal solutions. To make it possible to use such links in spite of this, we set $\Delta_{p,q} := s_q - e_p$ whenever it is possible to park a vehicle between p and q at the depot. We call these special dead-head trips **pull-in-pull-out trips**. They were first described in Bokinge/Hasselström (1980). (This concept has been considered unacceptable and has therefore been rejected.) Whenever $e_p + \Delta_{p,q} \leqslant s_q$ is satisfied, the corresponding dead-head trip is called **compatible**.

A **vehicle schedule** or (duty) is a chain of trips such that the first trip is a pull-out trip, the last trip is a pull-in trip, and the timetabled and unloaded trips occur alternately. A vehicle schedule is called **feasible** if all its trips belong to the same depot. A circulation is also called a **block** (or **rotation**) if it includes no pull-in-pull-out trip.

For each depot $d \in \mathcal{D}$, we introduce the following sets of trips: $A_d^{\text{t-trip}} := \{(t^-, t^+) |\ t \in \mathcal{T}_d\}$ (timetabled trips) and $A_d^{\text{u-trip}} := \{(d^+, t^-), (t^+, d^-) |\ t \in \mathcal{T}_d\} \cup \{(p^+, q^-) |\ p, q \in \mathcal{T}_d \wedge e_p + \Delta_{p,q} \leqslant s_q\}$ (unloaded trips). With each unloaded trip $a \in A_d^{\text{u-trip}}$, we associate a weight $c_a^d \in \mathbb{Q}$ representing its operational costs. In addition, we add to the weight of each pull-out trip a sufficiently large $\mathbf{M}$ representing the capital costs and being larger than the operational costs of any feasible solution. The minimization of this "two-stage" objective function first minimizes the fleet size and, subordinate, the operational costs among all minimal fleet solutions. With this terminology, the MDVSP is to find a weight minimal set of feasible vehicle schedules such that each timetabled trip is covered by exactly one vehicle schedule.

The MDVSP can be stated as an integer multicommodity flow problem as follows. For each depot $d \in \mathcal{D}$, let (d^-, d^+) denote an additional **backward arc** (on which depot capacities can be controlled) and let $A_d := A_d^{\text{t-trip}} \cup A_d^{\text{u-trip}} \cup \{(d^-, d^+)\}$. Let $D = (V, A)$ be a digraph with node set $V := \mathcal{D}^+ \cup \mathcal{D}^- \cup \mathcal{T}^- \cup \mathcal{T}^+$ and arc set $A := \dot{\bigcup}_{d \in \mathcal{D}} A_d$. Figure 1 illustrates a small example with $\mathcal{D} = \{\text{r,g}\}$, $\mathcal{T} = \{\text{a,b,c,d,e}\}$, $\mathcal{T}_\text{r} = \{\text{a,c,d}\}$, and $\mathcal{T}_\text{g} = \{\text{a,b,c,e}\}$.

An Integer Linear Program.

We introduce an integer variable x_a^d for each $a \in A_d$ and each $d \in \mathcal{D}$. x_a^d denotes a decision variable indicating whether a vehicle of depot d runs trip a

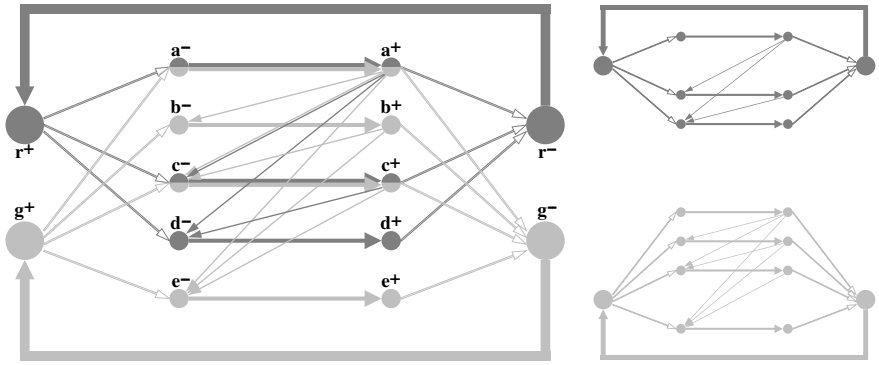

Figure 1: Digraph (V, A) and its single-depot graphs.

or not, unless a denotes the backward arc. In this case, x_a^d counts all employed vehicles of the depot d. The variables x_a^d are combined into vectors $x^d := (x_a^d)_{a \in A_d} \in \mathbb{R}^{A_d}$, $d \in \mathcal{D}$, and these into $x := (x^d)_{d \in \mathcal{D}} \in \mathbb{R}^A$.

Given a node $v \in V$, let $\delta^+(v)$ denote all arcs of A with tail in v and, accordingly, $\delta^-(v)$ denote all arcs with head in v. For a given set $\tilde{A} \subset A$, we define $x^d(\tilde{A}) := \sum_{a \in \tilde{A} \cap A_d} x_a^d$ and $x(\tilde{A}) := \sum_{d \in \mathcal{D}} x^d(\tilde{A})$. The self-suggesting ILP formulation of the MDVSP is

$$\min \sum_{d \in \mathcal{D}} \sum_{a \in A_d^{\text{u-trip}}} c_a^d \, x_a^d \tag{1a}$$

subject to

$$x\big(\delta^+(t^-) \cap \delta^-(t^+)\big) = 1, \quad \forall\, t \in \mathcal{T}, \tag{1b}$$

$$x^d\big(\delta^+(v)\big) - x^d\big(\delta^-(v)\big) = 0, \quad \forall\, v \in \mathcal{T}_d \cup \{d^+, d^-\} \;\; \forall\, d \in \mathcal{D}, \tag{1c}$$

$$x^d_{(d^-, d^+)} \leqslant \kappa_d, \quad \forall\, d \in \mathcal{D}, \tag{1d}$$

$$x_a^d \geqslant 0, \quad \forall\, a \in A_d \;\; \forall\, d \in \mathcal{D}, \tag{1e}$$

$$x \text{ integral.} \tag{1f}$$

Note that $x\big(\delta^+(t^-) \cap \delta^-(t^+)\big) = \sum_{d \in G(t)} x^d_{(t^-, t^+)}$. Constraints (1b), the **flow conditions**, ensure that each timetabled trip is serviced exactly once. Constraints (1c), the **flow conservations**, guarantee that the total flow value of each depot d entering some node $v \in V$ must also leave v.

The ILP (1) includes many redundant constraints that can be eliminated by performing some preprocessing steps as shown in Löbel (1997c). The main idea is to shrink t^- and t^+ to one node t, for all $t \in \mathcal{T}$, and to shrink d^- and d^+ to one node d, for all $d \in \mathcal{D}$. This corresponds to eliminating each arc not belonging to some unloaded trip and leads to the following equivalent ILP:

$$\min \sum_{d \in \mathcal{D}} \sum_{a \in A_d^{\text{u-trip}}} c_a^d \, x_a^d \tag{2a}$$

subject to

$$\begin{aligned} x\big(\delta^+(t)\big) &= 1, && \forall\, t \in \mathcal{T}, && (2\,\mathrm{b})\\ x^d\big(\delta^+(t)\big) - x^d\big(\delta^-(t)\big) &= 0, && \forall\, t \in \mathcal{T}_d \ \forall\, d \in \mathcal{D}, && (2\,\mathrm{c})\\ x^d\big(\delta^+(d)\big) &\leqslant \kappa_d, && \forall\, d \in \mathcal{D}, && (2\,\mathrm{d})\\ x^d_a &\geqslant 0, && \forall\, a \in A^{\text{u-trip}}_d \ \forall\, d \in \mathcal{D}, && (2\,\mathrm{e})\\ & x \text{ integral.} && && (2\,\mathrm{f}) \end{aligned}$$

Relaxations.

The natural relaxation of (2) is of course its LP relaxation

$$\min_{\substack{x \text{ satisfying}\\ (2\,\mathrm{b})\text{–}(2\,\mathrm{e})}} \quad \sum_{d\in\mathcal{D}} \sum_{a\in A^{\text{u-trip}}_d} c^d_a\, x^d_a. \tag{3}$$

Let $\nu \in \mathbb{R}^{\mathcal{T}}$, $\pi := (\pi^d \in \mathbb{R}^{\mathcal{T}_d})_{d\in\mathcal{D}}$, and $0 \leqslant \gamma \in \mathbb{R}^{\mathcal{D}}$ denote the dual multipliers for (2 b), (2 c), and (2 d). Consider a subset $\tilde{A} \subseteq A^{\text{u-trip}}$. The LP containing just the columns corresponding to $\tilde{A}$ is called **restricted** LP of (3) and is denoted by **RLP**.

We briefly give two possible Lagrangean relaxations for the MDVSP. For notational simplicity, we use the same symbols for the dual variables of (3) and for the Lagrangean multipliers of the two following Lagrangean relaxations.

Let $\pi := (\pi^d \in \mathbb{R}^{\mathcal{T}_d})_{d\in\mathcal{D}}$ and $0 \leqslant \gamma := (\gamma^d)_{d\in\mathcal{D}} \in \mathbb{R}^{\mathcal{D}}$ denote the Lagrangean multipliers according to the flow conservations (2 c) and the depot capacities (2 d). Relaxing (2 c) and (2 d), we obtain a Lagrangean dual $\mathrm{LR}_{\mathrm{fcs}}$ reading $\max_{\pi;\gamma\geqslant 0} L_{\mathrm{fcs}}(\pi,\gamma)$ with inner minimization problem

$$\begin{aligned} L_{\mathrm{fcs}}(\pi,\gamma) := \min \sum_{d\in\mathcal{D}} \Bigg\{ & \sum_{a\in A^{\text{u-trip}}_d} c^d_a\, x^d_a \;-\; \sum_{t\in\mathcal{T}_d} \pi^d_t \Big(x^d\big(\delta^+(t)\big) - x^d\big(\delta^-(t)\big) \Big) \\ & -\; \gamma^d \Big(\kappa_d - x^d\big(\delta^+(d)\big) \Big) \Bigg\} \end{aligned} \tag{4a}$$

subject to

$$x \text{ satisfies } (2\,\mathrm{b}),\ (2\,\mathrm{e}),\ (2\,\mathrm{f}), \text{ and } -x\big(\delta^-(t)\big) = -1,\ \forall\, t \in \mathcal{T}. \tag{4b}$$

The subscript "fcs" of L_{fcs} and $\mathrm{LR}_{\mathrm{fcs}}$ stands for **F**low-**C**on**S**ervation. Note, for fixed arguments, L_{fcs} is a minimum-cost flow problem.

The second Lagrangean relaxation is based on the ILP (1). Let $\nu := (\nu_t)_{t\in\mathcal{T}} \in \mathbb{R}^{\mathcal{T}}$ denote the Lagrangean multipliers for to the flow conditions (1 b). Relaxing (1 b), we obtain a Lagrangean dual $\mathrm{LR}_{\mathrm{fcd}}$ reading $\max_{\nu} L_{\mathrm{fcd}}(\nu)$ with inner minimization problem

$$L_{\mathrm{fcd}}(\nu) := \nu^{\mathrm{T}}\mathbb{1} \;+\; \sum_{d\in\mathcal{D}} \min \Bigg\{ \sum_{a\in A^{\text{u-trip}}_d} c^d_a\, x^d_a \;-\; \sum_{t\in\mathcal{T}_d} \nu_t\, x^d_{(t^-,t^+)} \Bigg\} \tag{5a}$$

subject to

$$x \text{ satisfies } (1\,\mathrm{c})\text{–}(1\,\mathrm{f}) \text{ and } x^d_{(t^-,t^+)} \leqslant 1, \quad \forall\, t \in \mathcal{T}_d \;\; \forall\, d \in \mathcal{D}. \qquad (5\,\mathrm{b})$$

The subscript "fcd" of L_{fcd} and $\mathrm{LR}_{\mathrm{fcd}}$ stands for **F**low-**C**on**D**ition. Note that L_{fcd} decomposes a constant part $\nu^{\mathrm{T}}\mathbb{1}$ and into $|\mathcal{D}|$ independently solvable minimum-cost flow circulation problems.

It is easy to see that the additional constraints in (4b) and (5b) are redundant in (2) and (1), respectively, but necessary to receive convenient inner minimization problems that are efficiently solvable minimum-cost flow problems.

3 Discussion of the Model

In this section, we discuss the relation and differences between the arc-oriented multicommodity flow and the path-oriented set partitioning formulations. We will also distinguish our multicommodity flow formulation from some other (arc-oriented) models that have been presented in the literature.

Multicommodity Flow and Set Partitioning Models.

Arc-oriented multicommodity flow and path-oriented Dantzig-Wolfe (DW) set partitioning formulations are usually used to model the MDVSP. Applied to vehicle scheduling problems from practice, their corresponding ILP formulations can provide several million integer variables. Solving such large ILPs requires column generation techniques.

For the arc-oriented model, column generation can be seen as an implicit pricing technique (see Schrijver (1989)): one works on restricted subsets of active arcs that are generated and eliminated in a dynamic process. For the DW decomposition, column generation usually leads to pricing problems in the form of constraint shortest path problems. Many researchers automatically associate the term "column generation" with the solution process used in a DW decomposition (e. g., see Soumis (1997)). To distinguish this use of the term "column generation" from those as a general LP pricing technique in the sense of Schrijver, DW column generation is also called *delayed column generation* as proposed in Chvátal (1980). To avoid misunderstandings, we will use in this paper the term "column generation" as a general LP pricing technique in the sense of Schrijver.

Direct approaches to the multicommodity flow formulation can be used if all side constraints can be formulated solely in terms of the arcs of the network. This is the case for the MDVSP considered here. *DW decomposition* is in particular needed for problems that involve path constraints. It applies not only to vehicle scheduling problems, but also to applications of similar flavour, e. g., to crew and airline scheduling. For a survey on DW

set partitioning approaches to such problems, we refer the reader, e. g., to Desrosiers/Dumas/Solomon/Soumis (1995) and Soumis (1997).

Differences to Other Arc Oriented Models.

Practice-oriented methods for the MDVSP are in most cases based on a single-commodity minimum-cost flow relaxation within a **schedule first – cluster second** approach (see Daduna/Paixão (1995) for a detailed description of this approach). The multiple-depot formulation is reduced to a single-depot relaxation. Unlike multicommodity flow formulations, however, those single-depot relaxations yield two significant drawbacks:

Depot-groups and flow conservation: It is only possible to consider a single (depot independent) dead-head trip – we better call it *link* – between two timetabled trips. Such a link (t, t') is considered to be feasible with respect to the depot-groups if $G(t) \cap G(t') \neq \emptyset$. But if depot-groups must only be satisfied locally between two trips, the intersection of the depot-groups of a generated block may be empty. In other words, the solution would be infeasible, see Fig. 2. Splitting such infeasible block into its feasible parts can lead to suboptimal solutions.

Figure 2: Invalid block.

To avoid falling in such traps, the MDVSP should be modelled as a multicommodity flow problem. Many research groups have considered the MDVSP as a multicommodity flow problem long before we started our investigations. The requirement of many real-world applications to consider different depot-groups, however, was just realized in the last years, e. g., by Forbes/Holt/Watts (1994). It is obvious that multicommodity flow formulations are natural for this kind of scheduling problems.

Limited duration for dead-head trips: It is often the case that single-depot relaxations consider dead-head trips with a limited duration (e. g., see Daduna/Mojsilovic/Schütze (1993)). It is therefore only possible to generate blocks that must be linked to vehicle schedules in a succeeding step. Based on heuristic ideas, the main objective is to use as many dead-head trips as possible and, subordinate, to minimize operational costs. Obviously, this objective function does indeed minimize the number of blocks if depot-groups are handled correctly. At the same time, it is assumed that a block minimal solution provides also a fleet minimal solution. It can be shown, however, that this is not true in general, see Fig. 3. The blocks, which have been determined by this strategy are subdivided to the depots and depot-wise linked to vehicle schedules. These links correspond to pull-in-pull-out trips.

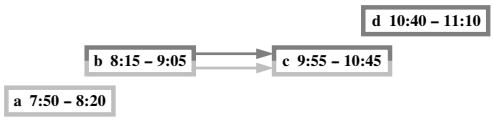

Figure 3: Minimal block solution is not fleet minimal.

It is clear that such a problem decomposition into two successive steps can lead to suboptimal solutions.

Figure 3 displays a multiple-depot instance with two depots for which the fleet minimal solution cannot be obtained with a block minimal solution: The first depot can service trips "b, "c", and "d", and the second depot can service "a", "b", and "c". Two timetabled trips may be linked by a pull-in-pull-out trip if the depot-groups are satisfied and if the two timetabled trips do not overlap. The maximum allowed duration of a dead-head trip is set to 60 minutes such that for both depots just the dead-head trip between "b" and "c" is possible. The block minimal number is three ("d" is assigned to the first depot, "a" is assigned to the second depot, and "b→c" is assigned to the first or the second depot) and requires three vehicles, but two vehicles are optimal ({a,c} and {b,d}) if each timetabled trip defines its own block.

Since it is insufficient to generate a fleet minimal solution in such a two step approach, linking blocks optimally and selecting user-defined unloaded trips must be done simultaneously. Pull-in-pull-out trips translate the decision of linking blocks into the terminology of dead-head trips. Therefore, using pull-in-pull-out trips makes it possible to generate a fleet minimal solution with minimum operational costs in one step.

Each pull-in-pull-out trip stands for a pull-in trip followed by a pull-out trip. The set of all pull-in-pull-out trips represents all feasible possibilities to link blocks to vehicle schedules. If we enlarge the user-defined unloaded trips by the pull-in-pull-out trips, the number of necessary vehicles is nothing but the number of used pull-out trips (or, equivalently, pull-in trips). Vice versa, if we replace each pull-in-pull-out trip of a vehicle schedule by the corresponding pull-in and pull-out trip, it is always possible to assign all resulting blocks of this vehicle schedule to a single vehicle.

4 Solving MDVSPs

The following section sketches the branch-and-cut method to solve the MDVSP. We give here a brief summary of the basic ingredients that we have required to solve our test instances. Because of a limited space, it is here not possible to explain each detail. Therefore, we refer the reader to our thesis (see Löbel (1997c)) which gives a full description of the following items.

Real-world problems of large cities such as Berlin have up to 25 thousand daily timetabled trips and 70 million unloaded trips. At first glance, it

seems impossible to solve such large ILPs and their LP relaxations exactly using commercial or publicly available standard software, even on the newest and fastest workstations or supercomputers. Nonetheless, with an intelligent combination of available LP and minimum-cost flow software together with implementations of many concepts of combinatorial optimization and integer programming, it has become possible to solve such problems to optimality by column generation and branch-and-cut on fast workstations. The basic components and concepts are:

- Lagrangean relaxations to quickly obtain tight lower bounds for the minimum fleet size and the minimum operational costs thereof as close as possible to the integer optimum value.
- Primal opening heuristics to obtain a first integer feasible solution and a good starting point for the LP relaxation.
- The LP relaxation approach with a column generation scheme including Lagrangean pricing.
- *LP-plunging* to exploit the information compiled in each (R)LP and its optimal solution.
- Branch-and-cut to solve a problem to proven optimality.
- The workhorses: MCF combined with a column generation and the LP solver CPLEX.

Our basic method to solve the MDVSP is to solve the integer linear programming formulation by primal and dual heuristics, column generation, and branch-and-cut (see Fig. 4).

First, we determine a "fast" and "tight" lower bound c_L by Lagrangean relaxation and compute an upper bound c_U using opening heuristics. Second, the LP relaxation is solved to optimality using a column generation and column elimination scheme. The column generation procedure is based on new Lagrangean pricing and on standard reduced cost pricing, the column elimination procedure uses only the reduced cost criterion. Within this iterative process, we optionally call an LP-plunging heuristic to find a better integer feasible solution. If the upper bound has been improved by the LP-plunging, we check whether $\frac{c_U}{L_{fcs}(0,0)}$ is "small enough" from a practical point of view and stop in this case.

Up to the point where the LP relaxation has been solved to optimality, our method generates for most test instances an optimal solution or a minimal fleet solution with a small gap in the operational costs. For many instances, the current solution obtained by LP-plunging is (almost) optimal, and we have already terminated the optimization process. Otherwise, let c_{LP} denote the optimal LP value. We generate as many nonactive columns as possible (respecting the main memory limit of the used computer) that have reduced

costs smaller than $c_U - c_{LP}$. Note that none of the other inactive variables can have a positive value in an integer solution yielding a smaller objective value than c_U. The resulting RLP is then fixed and solved by branch-and-cut. Of course, branch-and-cut solves the complete problem to proven optimality only if all necessary variables have been generated that may be included in an optimal integer solution. Otherwise, branch-and-cut is only a heuristic that solves the integer version of the last RLP to optimality. Our branch-and-cut approach turned out to be not a very important part of our method. Therefore, it is not described in this paper, but we refer the interested reader to Löbel (1997c).

We have also investigated a Dantzig-Wolfe decomposition. It turned out that such a decomposition approach is unsuitable for the MDVSP, at least for the test set that we have investigated. The major obstacle here is that the continuous master problem relaxations become too hard to solve efficiently. Especially for problems with more than one thousand timetabled trips, the LU factorization in solving a restricted master problem takes far too much time.

4.1 Lagrangean Relaxations

The first important use of the Lagrangean relaxations $\mathrm{LR}_{\mathrm{fcs}}$ and $\mathrm{LR}_{\mathrm{fcd}}$ is to compute quickly lower bounds as close as possible to the integer optimum value. The trivial problem relaxation $L_{\mathrm{fcs}}(0,0)$, which is equivalent to the neglection of the flow conservation constraints and the depot capacities, already gives very good lower bounds. These lower bounds can be improved using a computationally expensive subgradient method.

Let c_U denote the value of an integer feasible solution and c^* denote the optimal integer solution value. Since $0 \leqslant L_{\mathrm{fcs}}(0,0) \leqslant c^* \leqslant c_U$, the percentage deviation between c_U and c^* can be approximated and estimated by

$$0 \leqslant \frac{c_U - c^*}{c^*} \leqslant \frac{c_U - L_{\mathrm{fcs}}(0,0)}{L_{\mathrm{fcs}}(0,0)}.$$

Thus, as long as the LP relaxation is not solved to optimality, the somewhat weaker lower bound $L_{\mathrm{fcs}}(0,0)$ can be used to estimate the quality of integer feasible solutions that have been generated by the opening or LP-plunging heuristics.

4.2 Opening Heuristics

We have implemented two opening heuristics considering depot capacities only heuristically:

- ND: a cluster first – schedule second method based on a nearest depot heuristic. It simply assigns each trip $t \in \mathcal{T}$ to one of its depots of $G(t)$ that provides the cheapest pull-out and pull-in trips.

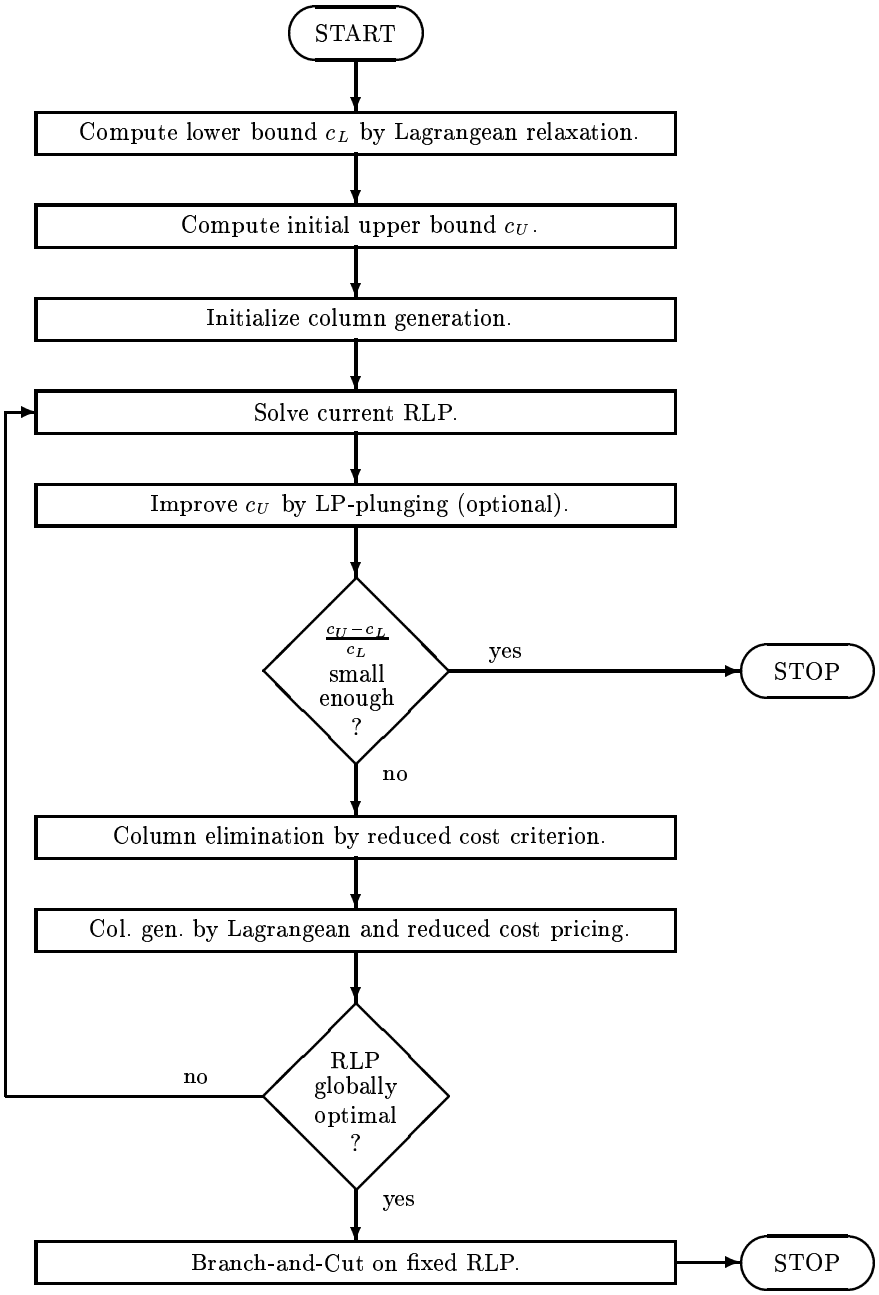

Figure 4: Solving MDVSPs: Flow chart.

- SCR: a schedule – cluster – reschedule method. This heuristic is not based on an assignment approach, but rather on minimum-cost flow that retains all degrees of freedom. The individual depot nodes in $\mathcal{D}^+$ and $\mathcal{D}^-$ are contracted to single depot nodes $\mathcal{D}$, and the depot individual flow conservations are aggregated to one flow conservation. This relaxed system is solved to optimality. The resulting vehicle schedules may violate some individual flow conservations, and these violations can be repaired heuristically: the vehicle schedules are pieced together into feasible blocks defining a cluster that considers depot capacities. This cluster can than be (re-)scheduled optimally. This procedure is embedded in a tabu search that forbids the use of certain dead-head trips that create the infeasibilities of a current (relaxed) solution.

Complexity theory tells us that it is $\mathcal{NP}$-hard just to find a feasible solution if depot capacities are considered (see Löbel (1997c)). However, it turned out that depot capacities are often soft constraints and can thus sometimes be violated somewhat since vehicles can often be shifted from one garage (or depot) to another.

4.3 Solving the LP Relaxation

Our computational investigations on real-world test data have shown that *the hard part* in solving the MDVSP to proven optimality *is to solve the LP relaxation.* Standard software alone (e. g., pure CPLEX) as well as standard approaches from integer linear programming (e. g., column generation and column elimination schemes based on the reduced cost criterion) are unable to solve larger instances with several thousand timetabled trips. We have developed a column generation method including **new** techniques based on the two Lagrangean relaxations $\mathrm{LR}_{\mathrm{fcs}}$ and $\mathrm{LR}_{\mathrm{fcd}}$. Lagrangean pricing is a new idea that makes it possible to solve the LP relaxation (3) of even large-scale instances.

The basic idea of Lagrangean pricing is to approximate the LP relaxation with all active and inactive variables. It is important that dual information compiled in the last RLP are used as it is done by the reduced cost pricing method: Let $\tilde{\nu}$, $\tilde{\pi}$, and $\tilde{\gamma}$ denote the values of the dual LP multipliers according to the flow conditions (2b), the flow conservations (2c), and the depot capacities (2d) of the last basis of the current RLP. We evaluate L_{fcs} and L_{fcd} at $(\tilde{\pi}, \tilde{\gamma})$ and $\tilde{\nu}$, respectively, using the complete variable set. The solutions L_{fcs} and L_{fcd} can be interpreted as a set of vehicle schedules and/or unloaded trips that seem to be advantageous for the given shadow prices of the current RLP relaxation. Each still nonactive variable according to such an unloaded trip is therefore generated and added to the next RLP. The new idea of Lagrangean pricing is to generate, in addition to columns with negative reduced costs, also those that have nonnegative reduced costs, but are necessary to complete (almost) optimal solutions.

Computational tests have shown that the LP relaxations of our problems can only be solved if we use advanced starting solutions that yield a value as close as possible to the LP optimum. Therefore, the set of all unloaded trips, used hitherto by the solutions of $L_{\text{fcs}}(0,0)$ and the opening heuristics, define the first restricted arc set $\tilde{A}$ of the initial RLP.

4.4 LP-Plunging

Our real-world MDVSP instances exhibit in practice a nice "almost-integrality property": solutions x of the LP relaxation (3) or an RLP include few fractional variables. It is often the case that x is integral or there exists some integral solution already yielding (almost) the same objective value. Moreover, the gap between the optimal LP or RLP value and its optimal integer value is often small or zero. LP-plunging makes use of this property by iteratively rounding up and fixing components of the LP solution and reoptimizing the enlarged LP.

Given an LP (3) or an RLP and a nonintegral feasible vector x. Let $\Delta \in (0.5, 1.0)$ denote some threshold value for which all fractional variables having a value within $(\Delta, 1)$ are rounded up and fixed to one, and let $\alpha \in (0.5, 1.0)$ denote some shrink factor for Δ. The standard values for Δ and α are 0.95 and 0.9. As long as the current x is nonintegral and the current (R)LP is primal feasible, the following steps are performed:

1. All variables $x_a^d \in (\Delta, 1)$ are rounded up and fixed to 1.
2. If no variable was fixed to 1 and if $\alpha\Delta > 0.5$, we reset $\Delta := \alpha\Delta$ and go to 1. Otherwise, each fractional variable yields $x_a^d \leqslant 0.5$, and we fix the first variable x_a^d to 1 yielding the largest fractional value.
3. Logical implications are performed, i. e., for each variable x_{ij}^d being fixed to one, we fix the variables of all arcs $(\delta^+(i) \cup \delta^-(j)) \cap A_d^{\text{u-trip}}$ and $(\delta(i) \cup \delta(j)) \setminus A_d^{\text{u-trip}}$ to zero.
4. The LP enlarged by the variable fixings is reoptimized with the dual simplex algorithm.

If the LP-plunging succeeds, the clustering defined by the last (integral) x is depot-wise rescheduled to optimality using all possible unloaded trips.

Since the restricted column set of an RLP generally includes only a small part of $A^{\text{u-trip}}$, the LP-plunging generates in many cases only poor or infeasible integer solutions. If this is the case, we enlarge the current RLP parameter controlled by inactive columns (such that the probability to find a better integer solution is presumably increased, but the dual feasibility of the optimal basis of the RLP is not destroyed and the main memory limit of the workstation is not exceeded) and apply the LP-plunging a second time.

4.5 The Workhorses: Minimum-Cost Flow and LP

Solving the MDVSP with our algorithm requires the efficient solution of minimum-cost flow problems and LPs at several steps: the minimum-cost flow problems stem from single-depot subproblems and Lagrangean functions (4) and (5), the LPs are RLPs from the LP relaxation (3).

Standard tools in vehicle scheduling are, of course, network flow models and algorithms, which have been profoundly investigated and are well understood. We have implemented a network simplex algorithm, called MCF, and *combined it with column generation.*[1] This implementation allows solving the single-depot problems and subproblems to optimality in a few seconds. The Lagrangean functions can also be evaluated in a few seconds up to a few minutes, depending on the problem size. For instance, the Lagrangean function L_{fcs} of the problem with 70 million unloaded trips can be exactly evaluated in about 15 minutes. MCF (without column generation) is available free of charge for academic use via WWW at www.zib.de/Optimization (see Löbel (1997b)).

We solve the linear programs with the primal as well as the dual simplex solver of CPLEX, currently version 4.0.9 CPLEX (1997). CPLEX turned out to be a reliable and robust method for our degenerate (R)LP problems.

5 Test Data

Our computational investigations are based on real-world data from the city of Berlin (BVG), the city of Hamburg (HHA), and the region around Hamburg (VHH). Different parameter settings and optimization aspects yielded in the test instances, which are illustrated in Tab. 1 ($\varnothing G := \sum_{t \in \mathcal{T}} G(t)/|\mathcal{T}|$ denotes the average depot-group size). Note that the number of equations of (3) is equal to the number of flow conditions and flow conservations.

Currently, BVG maintains 9 garages and runs 10 different vehicle types resulting in 44 depots. For a normal weekday, about 28,000 timetabled trips

[1] Due to the very special ("almost transportation") structure of the considered minimum cost flow problems and the importance of fast solutions, an anonymous referee proposed to try also more specialized algorithms like augmenting path methods. We have doubts that such methods could improve the performance of our branch-and-cut method: First, the portion of the total run time spend in the minimum cost flow subroutines can be neglected. Second, the considered minimum cost flow problems are *not* assignment or transportation problem although they include such substructures. Therefore, very specialized assignment or transportation algorithms *cannot* be used. Third, the cost coefficients of the inner minimizations problems L_{fcs} and L_{fcd} can also be negative. Augmenting path methods require a nonnegative objective function, otherwise, nontrivial network transformations are necessary. A general purpose network simplex code, however, can handle arbitrary objective function easily. Last, we believe that augmenting path methods *cannot* handle up to 70 million variables efficiently. We have also compared MCF with other efficient network flow solver such as RELAX IV and the cost scaling code CS 2 (see Löbel (1996)). For our special minimum-cost flow problems, MCF turned out to be, on the average, the fastest code.

Test Sets	$\|\mathcal{D}\|$	$\|\mathcal{T}\|$	$\|A^{\text{u-trip}}\|/1,000$		$\varnothing G$	no. of equations
			User[a]	All		
Berlin 1	44	24,906	846	69,700	4.03	125,255
Berlin 2	49	24,906	304	13,200	1.56	63,641
Berlin 3	3	1,313	77	2,300	2.33	4,370
Berlin-Spandau 1	9	2,424	164	3,700	4.94	14,418
Berlin-Spandau 2	9	3,308	327	8,800	5.49	21,470
Berlin-Spandau 3	13	2,424	39	590	1.92	7,103
Berlin-Spandau 4	13	3,308	72	1,530	2.25	10,753
Berlin-Spandau 5	13	3,331	75	1,550	2.25	10,834
Berlin-Spandau 6	13	1,998	28	380	1.90	5,798
Berlin-Spandau 7	7	2,424	145	3,300	4.16	12,506
Berlin-Spandau 8	7	3,308	283	7,800	5.02	18,376
Hamburg 1	12	8,563	1,322	10,900	2.23	27,696
Hamburg 2	9	1,834	99	1,000	2.02	5,549
Hamburg 3	2	791	30	200	1.32	1,835
Hamburg 4	2	238	2	23	1.04	487
Hamburg 5	2	1,461	85	580	1.31	3,379
Hamburg 6	2	2,283	176	1,600	1.33	5,323
Hamburg 7	2	341	6	34	1.32	795
Hamburg-Holstein 1	4	3,413	230	4,000	1.68	9,167
Hamburg-Holstein 2	19	5,447	1,054	9,400	3.65	25,334

[a]The unloaded trips without pull-in-pull-out trips.

Table 1: Real-world test sets.

have to be serviced. Since BVG outsources some trips to third-party companies, this number reduces to 24,906. Using all degrees of freedom, these 25 thousand trips can be linked with about 70 million unloaded trips.

Berlin 1: This is the complete BVG problem with all possible degrees of freedom.

Berlin 2: This problem is based on the timetabled trip set of Berlin 1, but the depots and the dead-head trips are generated with different rules resulting in fewer degrees of freedom.

Berlin 3: This is a small test instance including 9 lines from the south of Berlin and 3 depots from one single garage.

Berlin-Spandau 1 – 8: All the test sets denoted by Berlin-Spandau are defined on the data of the district of Spandau for different weekdays and different depot generation rules.

HHA together with some other transportation companies maintain 14 garages with 9 different vehicle types resulting in 40 depots. More than 16,000 daily timetabled trips must be scheduled with about 15.1 million unloaded trips. This problem decomposes into a 12-depot problem, a 9-depot problem, five smaller 2-depot problems, and nine small 1-depot problems.

Hamburg 1 – 7: Here we consider the multiple-depot subproblems of HHA.

VHH currently plans 10 garages with 9 different vehicle types. The garage-vehicle combinations define 19 depots. The 5,447 timetabled trips of VHH can be linked with about 10 million unloaded trips.

Hamburg-Holstein 1: This is a subset of VHH containing not all its depots and trips.

Hamburg-Holstein 2: This test set is based on the complete data of VHH.

6 Computational Results

In the following, we want to prove the effectiveness of our developed and implemented method to solve large MDVSP instances from practice. All the computational tests have been performed on a SUN Model 170 UltraSPARC with 512 MByte main memory and 1.7 MByte virtual memory. We have been the only user on this machine during our test runs. All linear programs have been solved with CPLEX, version 4.0.7 and 4.0.9, all minimum-cost flow problems and single-depot subproblems have been solved with our network simplex code MCF combined with a column generation.

The following objective values (fleet sizes and operational weights) are given in Tab. 2: (i) the lower bounds obtained with $L_{\mathrm{fcs}}(0,0)$ and the LP relaxations; (ii) the integer optimum or, if the optimum is still unknown, the best integer solution values; (iii) the upper bounds obtained by our opening heuristics as well as by our branch-and-cut method starting with SCR or ND and terminating after a maximum run time limit of 10 hours (and 16 hours for Berlin 1 starting with ND). The largest problem, Berlin 1, has not been solved to optimality. Berlin 2 and Berlin-Spandau 2 and 8 have been solved fleet minimally, but not to proven cost minimality.

The run times that have been required to solve the function $L_{\mathrm{fcs}}(0,0)$, the LP relaxation pure without LP-plunging, the opening heuristics SCR and ND, and our exact method (with and without using the optional LP-plunging within the column generation) are given in Tab. 3.

Lower Bounds.

To obtain lower bounds by Lagrangean relaxations, we have only considered $L_{\mathrm{fcs}}(0,0)$. For $L_{\mathrm{fcs}}(0,0)$, let ν^+ and ν^- denote optimal dual variables associated with the flow conditions $x\big(\delta^+(t)\big) = 1$ and $-x\big(\delta^-(t)\big) = -1$, respectively. We have shown in Löbel (1997c) that $L_{\mathrm{fcd}}(\nu^+ - \nu^-)$ and $L_{\mathrm{fcs}}(0,0)$ yield the same optimal value. The values obtained by $L_{\mathrm{fcs}}(0,0)$ give excellent approximations. The minimum integral fleet sizes can be approximated, on the average, by 99.94%. It is remarkable that the trivial problem relaxation – simply neglecting the flow conservations – gives such tight approximations. For 15 out of our 20 instances, the fleet sizes can be exactly approximated. Ignoring for those problems the values for the fleet size, the gap between the

Table 2: Fleet sizes and operational weights (optimal int. values are in bold).

Test Sets[b]	Lower bounds				Optimum or best solution		Upper bounds						
	$L_{\text{fcs}}(0,0)$		LP relaxation				SCR heuristic[a]				ND heuristic		
							pure		+ LP method		pure[c]	+ LP method	
	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Fleet	Weight
B 1	1323	715714	1323	759162[d]	1329	850680	1347	1317379	1335	1118287	1575	1356	982914
B 2	1350	715623	1353.7	797919	**1354**	777823	1366	1318085	**1354**	809611	1655	**1354**	788958
B 3	**69**	14043	**69**	**14119**	**69**	**14119**	**69**	14122	**69**	**14119**	70	**69**	**14119**
BS 1	**125**	65585	**125**	65611	**125**	**65611**	**125**	125786	**125**	65835	139	**125**	65901
BS 2	184	78947	184.5	79110	**185**	79052	**185**	289262	**185**	80430	207	**185**	92249
BS 3	**127**	90514	**127**	**93745**	**127**	**93745**	**127**	152109	**127**	**93745**	135	**127**	**93745**
BS 4	**191**	195844	**191**	**230846**	**191**	**230846**	192	395891	**191**	**230846**	222	**191**	**230846**
BS 5	**191**	191141	**191**	**227580**	**191**	**227580**	194	393922	**191**	**227580**	220	**191**	**227580**
BS 6	**98**	91109	**98**	**101075**	**98**	**101075**	**98**	132650	**98**	**101075**	109	**98**	**101075**
BS 7	**125**	65585	**125**	65611	**125**	**65611**	**125**	105853	**125**	**65611**	139	**125**	65724
BS 8	184	78947	184.5	79110	**185**	79093	**185**	259406	**185**	79273	207	**185**	79959
H 1	**432**	66874	**432**	71068	**432**	**71069**	446	70291	434	73066	489	**432**	71270
H 2	**103**	15356	**103**	**16070**	**103**	**16070**	104	16792	**103**	**16070**	114	**103**	**16070**
H 3	**39**	5557	**39**	**5860**	**39**	**5860**	**39**	6298	**39**	**5860**	41	**39**	**5860**
H 4	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**6**	**1358**
H 5	**62**	12092	**62**	**12502**	**62**	**12502**	**62**	13535	**62**	**12502**	65	**62**	**12502**
H 6	**111**	15705	**111**	**15791**	**111**	**15791**	**111**	16588	**111**	**15791**	**111**	**111**	**15791**
H 7	**15**	2832	**15**	**2961**	**15**	**2961**	16	2836	**15**	**2961**	16	**15**	**2961**
HH 1	**201**	28697	**201**	**29027**	**201**	**29027**	**201**	30497	**201**	**29027**	213	**201**	**29027**
HH 2	360	51084	**362**	52788	**362**	**52788**	363	72700	**362**	52986	393	**362**	53090

[a] Results obtained with SCR and ND: using only the heuristics ("pure"), and using each as the opening method within our exact LP method ("+ LP method"). In addition, we used a run time limit of ten hours and 16 hours for Berlin 1 starting with ND.

[b] B = Berlin, BS = Berlin-Spandau, H = Hamburg, HH = Hamburg-Holstein

[c] The results of the operational weights are not satisfying and, because of a lack of space, omitted. They are given in Löbel (1997c).

[d] Best known feasible LP value.

Test Sets	Lower bounds			Upper bound (or optimum)					
	Lagrangean relaxation: $L_{\mathrm{fcs}}(0,0)$	Pure LP times starting with		SCR heuristic			Nearest depot heuristic		
				pure	+ Exact method		pure	+ Exact method	
					LP-plunging[a]			LP-plunging[b]	
		SCR	ND		always	b&c		always	b&c
Berlin 1	916	—[b]	—[b]	12386	—[b]	—[b]	171	—[b]	—[b]
Berlin 2	229	34795	32767	3810	—[b]	35202[c]	79	30985[c]	33248[c]
Berlin 3	17	431	311	25	389	435	7	249	330
B-Spandau 1	27	43777	66501	343	59337	44053	15	68487	134386
B-Spandau 2	93	112337	165048	1939	—[b]	138212[c]	39	—[b]	240852[c]
B-Spandau 3	9	975	739	42	1626	990	7	953	758
B-Spandau 4	25	6014	4384	334	6228	6077	14	6773	4433
B-Spandau 5	31	5264	5618	354	6896	5304	15	13821	5666
B-Spandau 6	17	162	227	7	211	173	6	188	244
B-Spandau 7	23	24717	46597	259	26606	24793	14	45810	52031
B-Spandau 8	67	82284	62041	1238	146284[c]	84725[c]	36	—[b]	63215[c]
Hamburg 1	185	—[b]	50246	2868	—[b]	—[b]	29	88767	53971
Hamburg 2	12	875	685	103	732	902	7	926	708
Hamburg 3	4	35	31	16	37	41	2	34	38
Hamburg 4	2	3	3	1	3	3	1	3	3
Hamburg 5	10	258	155	86	288	279	5	119	174
Hamburg 6	18	148	84	30	158	181	11	124	86
Hamburg 7	2	8	9	2	11	10	1	8	11
H-Holstein 1	40	2619	2087	199	2166	2695	18	2445	2158
H-Holstein 2	101	46673	64489	1696	55915	54604	40	68534	71562

Table 3: Run times in seconds.

[a] LP-plunging is only used in the branch-and-cut part on a fixed RLP or always whenever an RLP has been solved.
[b] Not solved to optimality since, for instance, the objective progress was too small or stalling occurred.
[c] The problem has been solved fleet minimally, but not to proven cost minimality.

operational costs of $L_{\text{fcs}}(0,0)$ and the optimum is at most 16 % and 5 % on the average. The computing times are quite fast: For Berlin 1, $L_{\text{fcs}}(0,0)$ with 70 million variables can be evaluated in about 15 minutes; for all the other instances together, it can be computed in 15 minutes.

All LP relaxations, except for Berlin 1, have been solved to optimality. To find a fleet minimal LP value for Berlin 1, our column generation requires about 200 hours cpu time. The values obtained by the LP relaxation give lower bounds quite close to the integer optimal values. For 12 out of the 20 considered instances, the LP relaxation already provides the integer optimal value, and for 3 instances, it can be obtained by rounding up the LP value to the next integer value. For Berlin 1, we do not know the minimal number of vehicles, but expect that the fleet size lower bound provided by the LP relaxation is also tight. Whenever the LP relaxation provides an exact fleet size, it also provides the minimal operational weights.

We have seen that the LP values are quite tight. A similar phenomenon is observed by Forbes/Holt/Watts (1994): 22 of their 30 test instances with up to 600 trips have integral LP solutions, and the largest gap between the LP value and the integral optimum is at most 0.003 % for the remaining problems. So, this observation does not seem to be a *small scale phenomenon*.

The value of the operational weights in the objective value of the lower bounds do not necessarily define lower bounds for the integer optimal weights among all minimal fleet solutions. To estimate the quality of the operational weights requires that the lower bound of the fleet size is tight. For all problems that do not satisfy this condition, however, we believe that they nevertheless give good estimated values for the minimal operational weights.

Comparing the run times of the Lagrangean and LP relaxation, it is obvious that Lagrangean relaxations $L_{\text{fcs}}(0,0)$ are the faster method to obtain good lower bounds quickly. The better lower bounds provided by the LP relaxation require long run times that are only justified by a succeeding branch-and-cut method. The solution produced by SCR are always significantly better than those of ND. On the average, however, SCR used as the opening heuristic for the branch-and-cut algorithm does not provide better starting points. It is worth mentioning that starting without any heuristically generated solution, our LP method is unable to solve any of our larger problem instances at all.

Upper Bounds.

We will now consider the upper bounds obtained by the two opening heuristics (SCR and ND) and obtained by the exact branch-and-bound method starting with SCR and ND, using LP-plunging between two RLPs, and terminating after a given run time limit of 10 hours (and 16 hours for Berlin 1 starting with ND).

The trivial opening heuristic ND already delivers good results: The fleet size gap is, on the average, about 10 % with a standard deviation of 6 %. From

a practical point of view, however, the operational costs of these solutions are not acceptable. The better results are obtained from the SCR heuristic: The average fleet size gap is 0.8 % with a standard deviation of 1.6 %. The operational costs of these solutions are comparable to the results obtained by the best codes currently used in practice.

We almost always obtain optimal results if we apply our exact branch-and-cut method with a time limit of 10 hours. The objective gaps are, on the average, less than 0.12 %. It does not make any difference which opening heuristic we use for the exact method since the run times are comparable for both. The run times of our exact method may be decreased if we use both opening heuristics together to determine the first RLP. This may be the basis for further computational tests.

Figures 5–8 display the development of the upper bound values (fleet sizes and operational weights) obtained by the LP-plunging heuristic in proportion to the integer optimal (or lower bound) values. Starting our method with the solution obtained with ND, the fleet sizes can be approximated in two hours with a gap less than 3 %, in 4 four hours with gap of about 1 %, and in 6 hours with a gap less than 1 % for all problems except Berlin 1, see Fig. 5. Starting with the solution obtained with SCR, the fleet sizes can be approximated in one hour with a gap less than 2 % and in 10 hours with a gap less than 1 %, see Fig. 7. There is also a positive development of the operational costs: compared with the optimal integer costs of fleet minimal solutions, the operational costs can be approximated with a small gap, see Figs. 6 and 8. If the run time limit is 10 hours or more, the four figures show that it is meaningless which opening heuristic is used, the results are in any case comparable. However, if there is a stronger time limit of two or three hours, starting with SCR provides better results.

Optimal Solutions

Without any run time limit, each instance of our test set, with the exception of the problem Berlin 1, can be solved to proven fleet minimality. With the exceptions of the problems Berlin 2, Berlin-Spandau 2, and Berlin-Spandau 8, each instance can be solved to proven fleet and cost optimality.

With the current version of our branch-and-cut method, solving really large-scale problems to proven optimality leads to impractical run times. In particular, solving Berlin 1 with 70 million variables to optimality is still a challenge to us. Nevertheless, the results obtained with our methods are currently the best obtainable. Solutions providing possibly a gap of a few vehicles, but with reasonable operational weights can be computed in acceptable run times.

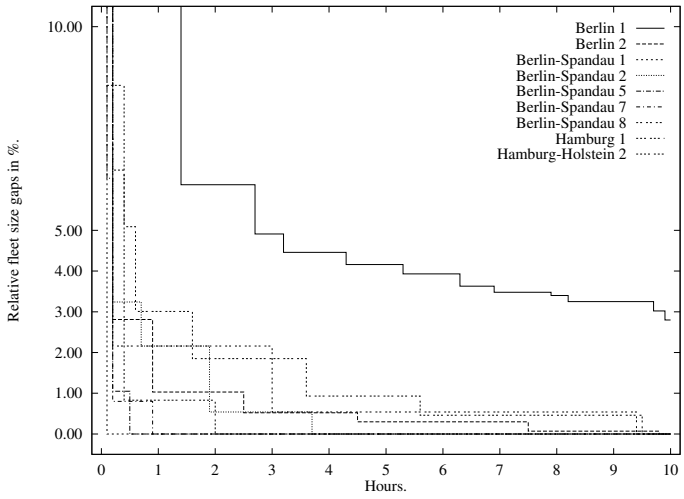

Figure 5: Development of fleet size upper bounds of problems requiring more than 2 hours run time to obtain a fleet minimal solution; starting with the ND heuristic.

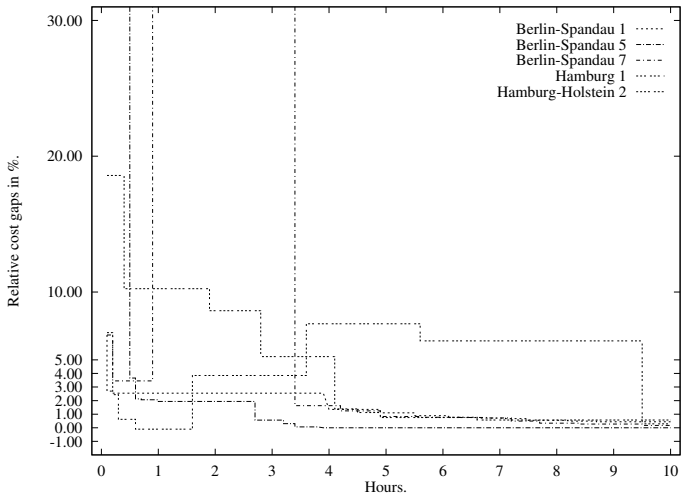

Figure 6: Development of operational weight upper bounds of problems requiring more than 2 hours run time to obtain a fleet minimal solution and knowing the minimum weight among all minimal fleet solutions; starting with the ND heuristic.

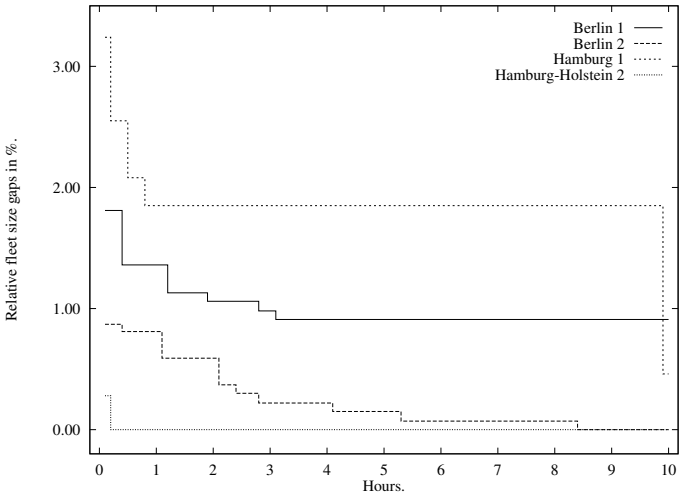

Figure 7: Development of fleet size upper bounds of problems requiring more than 2 hours run time to obtain the optimum; starting with the SCR heuristic.

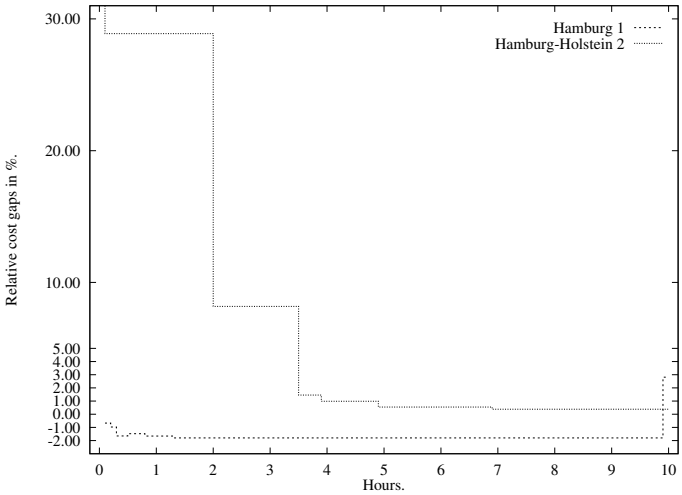

Figure 8: Development of operational weight upper bounds of problems requiring more than 2 hours run time to obtain the optimum and knowing the minimum weight among all minimal fleet solutions; starting with the SCR heuristic.

7 Conclusions

This paper is devoted to the Multiple-Depot Vehicle Scheduling Problem (MDVSP). We have presented a branch-and-cut method for its solution. A well-chosen combination of these methods turned out to be able to solve (almost) all problems of practical interest in accepatable running times. The success of the implementations, of course, gains from the (in the recent years drastically increased) computing power of modern workstations and sophisticated commercial optimization software (such as the LP solver CPLEX). We summarize some of our findings:

Upper bounds that have been generated with the schedule – cluster – reschedule heuristic (SCR) can be computed quickly and are of high quality. Compared with the optimal integer solutions, SCR provides solutions with a fleet size and operational weight gap of less than 1.25 % and 5.2 %, respectively.

Lagrangean relaxations allow to compute tight lower bounds even for large multiple-depot instances. Lagrangean relaxations can be used to quickly simulate fleet and cost effects of different parameter settings and, thus, to easily find out a useful scenario.

Branch-and-cut is capable of solving even very large multiple-depot instances to optimality, see Tab. 2.

Lagrangean pricing is a good idea to solve the large degenerate LPs that come up in solving multiple-depot instances with branch-and-cut. Our initial code used the well known standard reduced cost pricing techniques. However, this did not work at all because of stalling. To cure stalling, we introduced (what we call) Lagrangean pricing. We propose it as one of the basic ingredients of an effective method to solve multiple-depot vehicle scheduling problems. Similar positive results have been observed by Fischetti/Toth (1996) and Fischetti/Vigo (1996) also dealing with large degenerate LPs. We believe that variable pricing based on Lagrangean relaxation is a useful tool that can help to solve many combinatorial optimization problems.

Computational breakthrough: to our knowledge, at present no other implementation is able to solve MDVSPs with more than 1,000 timetabled trips to optimality. Our code has successfully produced optimal solutions of various real-world problem instances with up to 25 thousand timetabled trips. The integer multicommodity flow problems arising this way are orders of magnitude larger than what other codes are able to handle. The largest real instance we encountered gave rise to an integral multicommodity flow problem with about 125 thousand equations and 70 million integer variables. We could not produce an optimal solution, but found a solution with a fleet size gap of less than 0.5 %.

Possible savings indicated by our test runs are immense. Compared with a manual planning process, the SCR heuristic indicates savings of about 19 % of the vehicles and about 14 % of the operational costs. Compared with an assignment heuristic, our branch-and-cut method indicates savings of several vehicles and about 10 % cost reduction. However, the final evaluations of the SCR generated solutions have not been finished by BVG, HHA, and VHH yet. It still has to be checked whether our vehicle schedules provide a useful input for duty scheduling, the next step in the hierarchical planning process. It is therefore not clear how much of these indicated savings can be obtained in practice. Nonetheless, our methods can solve large problems optimally. The Berliner Verkehrsbetriebe, for instance, expect to save about DM 100 million per year with our SCR heuristic (see Schmidt (1997)).

There is a high demand within industry for efficient methods for the MDVSP. Parts of our system have been purchased by BVG for their planning system BERTA, by IVU for MICROBUS II, and by the research department of the SIEMENS AG in Munich.

8 Acknowledgements

We are grateful to Manfred Völker and Anna Neufeld of HanseCom GmbH for their support to model the MDVSP and providing us with real-world problems from the Hamburger Hochbahn AG and the Verkehrsbetriebe Hamburg-Holstein AG. We are grateful to Uwe Strubbe of IVU GmbH for providing us with real-world problems from the Berliner Verkehrsbetriebe. We are grateful to the Berliner Verkehrsbetriebe, the Hamburger Hochbahn AG, and the Verkehrsbetriebe Hamburg-Holstein AG for their kind permission to use and publish their data. We are indebted to Martin Grötschel, Ralf Borndörfer, and Alexander Martin (all at ZIB), and Bob Bixby for their helpful discussions. We are also grateful to Bob Bixby and ILOG CPLEX Division for regularly providing us access to the newest versions of CPLEX and the CPLEX Callable Library. This work has been supported by the German Federal Ministry of Education, Science, Research, and Technology grant no. 03-GR7ZIB -7.

References

Ball, M.O. / Magnanti, T.L. / Monma, C.L. / Nemhauser, G.L. (editors) **(1995):** *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*. Elsevier Science B.V.

Bokinge, U. / Hasselström, D. (1980): Improved vehicle scheduling in public transport through systematic changes in the time-table. *European Journal of Operational Research*, 5:388–395.

Borndörfer, R. / Grötschel, M. / Löbel, A. (1995): Alcuin's transportation problems and integer programming. Preprint SC 95-27, Konrad-Zuse-Zentrum für Informationstechnik Berlin. Available at www.zib.de. To appear in Butzer, P.L. / Jongen, H.T. / Oberschelp, W. (editors), *Charlemagne and his Heritage: 1200 Years of Civilization and Science in Europe, Volume II: The Mathematical Arts*, Brepols Publishers.

Bussieck, M. / Winter, T. / Zimmermann, U.T. (1997): Discrete optimization in public rail transport. In Liebling, T.M. / de Werra, D. (editors), *Mathematical Programming: A Publication of the Mathematical Programming Society*, pages 415–444. Elsevier Science B.V.

Carpaneto, G. / Dell'Amico, M. / Fischetti, M. / Toth, P. (1989): A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19:531–548.

Chvátal, V. (1980): *Linear programming.* W. H. Freeman and Company, New York.

CPLEX (1997): *Using the CPLEX Callable Library.* ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA. Information about CPLEX available at www.cplex.com.

Daduna, J.R. / Branco, I. / Paixão, J.M.P. (editors) **(1995):** *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Daduna, J.R. / Mojsilovic, M. / Schütze, P. (1993): Practical experiences using an interactive optimization procedure for vehicle scheduling. In Du, D.-Z. / Pardalos, P.M. (editors), *Network Optimization Problems: Algorithms, Applications and Complexity*, volume 2 of *Series on Applied Mathematics*, pages 37–52. World Scientific Publishing Co. Pte. Ltd.

Daduna, J.R. / Paixão, J.M.P. (1995): Vehicle scheduling for public mass transit – an overview. In Daduna/Branco/Paixão (1995).

Dell'Amico, M. / Fischetti, M. / Toth, P. (1993): Heuristic algorithms for the multiple depot vehicle scheduling problem. *Management Science*, 39(1):115–125.

Desrosiers, J. / Dumas, Y. / Solomon, M.M. / Soumis, F. (1995): *Time Constrained Routing and Scheduling.* In Ball/Magnanti/Monma/Nemhauser (1995), chapter 2, pages 35–139.

Fischetti, M. / Toth, P. (1996): A polyhedral approach to the asymmetric traveling salesman problem. Technical report, University of Bologna. To appear in *Management Science.*

Fischetti, M. / Vigo, D. (1996): A branch-and-cut algorithm for the resource-constrained arborescence problem. *Networks*, 29:55–67.

Forbes, M.A. / Holt, J.N. / Watts, A.M. (1994): An exact algorithm for multiple depot bus scheduling. *European Journal of Operational Research*, 72(1):115–124.

Freling, R. / Paixão, J.M.P. (1995): Vehicle scheduling with time constraint. In Daduna/Branco/Paixão (1995).

Kokott, A. / Löbel, A. (1996): Lagrangean relaxations and subgradient methods for multiple-depot vehicle scheduling problems. Preprint SC 96-22, Konrad-Zuse-Zentrum für Informationstechnik Berlin. Available at www.zib.de.

Löbel, A. (1996): Solving large-scale real-world minimum-cost flow problems by a network simplex method. Preprint SC 96-7, Konrad-Zuse-Zentrum für Informationstechnik Berlin. Available at www.zib.de.

Löbel, A. (1997a): Experiments with a Dantzig-Wolfe decomposition for multiple-depot vehicle scheduling problems. Preprint SC 97-16, Konrad-Zuse-Zentrum für Informationstechnik Berlin. Available at www.zib.de.

Löbel, A. (1997b): *MCF Version 1.0 – A network simplex implementation.* Available for academic use free of charge at www.zib.de.

Löbel, A. (1997c): *Optimal Vehicle Scheduling in Public Transit.* PhD thesis, Technische Universität Berlin.

Löbel, A. (1997d): Vehicle scheduling in public transit and Lagrangean pricing. Revised Preprint SC 96-26, Konrad-Zuse-Zentrum für Informationstechnik Berlin. Available at www.zib.de.

Ribeiro, C.C. / Soumis, F. (1994): A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, 42(1):41–52.

Schmidt, V. A. (1997): *Auf Sparkurs zum Ziel.* Rheinischer Merkur, number 39, page 37, 26th September 1997. In German.

Schrijver, A. (1989): *Theory of Linear and Integer Programming.* John Wiley & Sons Ltd., Chichester.

Soumis, F. (1997): *Decomposition and Column Generation.* Chapter 8 in Dell'Amico, M. / Maffioli, F. / Martello, S. (editors), *Annotated Bibliographies in Combinatorial Optimization*, pages 115–126. John Wiley & Sons Ltd, Chichester.