Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7, D-14195 Berlin-Dahlem, Germany

Andreas Löbel

Recent Computational Developments for Large-Scale Multiple-Depot Vehicle Scheduling Problems

Preprint SC 97–17 (May 1997)

Recent Computational Developments for Large-Scale Multiple-Depot Vehicle Scheduling Problems*

Andreas Löbel**

Abstract

This paper presents an integer linear programming approach with delayed column generation for the $\mathcal{NP}$-hard Multiple-Depot Vehicle Scheduling Problem (MDVSP) in public mass transit. We describe in detail all basic ingredients of our approach that are indispensable to solve truly large-scale real-world instances to optimality, and we report on computational investigations that are based on real-world instances from the city of Berlin, the city of Hamburg, and the region around Hamburg. These real-world instances have up to 25 thousand timetabled trips and 70 million dead-head trips.

Computational tests using the data of the Hamburger Hochbahn AG indicate savings of several vehicles and a cost reduction of about 10 % compared with the solution provided by HOT II, the vehicle scheduling tool of the HanseCom GmbH, Hamburg. Parts of our algorithms are already integrated in the BERTA system of the Berliner Verkehrsbetriebe (BVG) and will soon be integrated in the MICROBUS system of the Gesellschaft für Informatik, Verkehrs- und Umweltplanung mbH (IVU), Berlin.

Mathematics Subject Classification (1991): 90B06, 90B10, 90C05. 90C06.

1 Introduction

Public transportation in Europe is subject to serious structural changes caused by market deregulation, liberalization, and globalization. In Germany, there is a law regulating public transportation with the result that there is usually a monopolistic market controlled by local governments. Governments issue concessions to local companies to provide public transportation. In many cases, the governments themselves are the owners of these companies. In the past, it was the task of these local monopolists to provide transportation offers such that a politically predefined maximum loss per year would not be exceeded. Following introduction of a deregulation law and forced by the permanent lack of public money, however, German cities and local authorities will, from the year 1999 on, invite tenders for public transportation or parts thereof. In order to prevent their complete extinction from the market, public transportation companies will therefore have to turn from a deficit orientated monopolist to a competitive market player. One important factor in facing the challenges of a competitive market is, of course, cost reduction [BW97, KL97], which can be obtained by making intelligent use of latest mathematical methods.

Vehicle scheduling is one important field of the planning process in public transportation, where a large amount of money can be wasted – or saved if the available resources are employed efficiently. 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, in-company, 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 pragmatic solution techniques for the $\mathcal{NP}$-hard MDVSP and special, often polynomially solvable cases of it. Review articles on this topic are for instance [DDSS95, DP95, BWZ97] giving complete overviews of

*This work has been supported by the German Federal Ministry of Education, Science, Research, and Technology grant no. 03-GR7ZIB -7. Responsibility for the contents lies with the author.

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

the available publications dealing with the MDVSP. Up to now, only relatively small real-world MDVSPs have been solved to proven optimality: For instance, [FHW94] report on numerical investigations with up to 3 depots and 600 timetabled trips, some are taken from real-world problems.

Approaches to heuristically solve MDVSPs in practice are often based on a single-commodity network flow formulation, e.g., the assignment approach within the HOT II system of the HanseCom GmbH, Hamburg. It can be shown that the solutions generated by methods based on network flow do not necessarily attain the optimum since, in general, they do not use all degrees of freedom. Moreover, these methods do not provide tight lower bounds or do not provide lower bounds at all.

Together with HanseCom, we have investigated in a current research project the MDVSP, see also [Löb97c]. One major product of HanseCom is the program package HOT II, which covers the complete field of vehicle and duty scheduling. Its vehicle scheduling approach is based on an assignment approach. This approach was state of the art for many years. Although the vehicle scheduling of HOT has been known to be one of the best available heuristics in practice, HanseCom decided in 1993 to start a common project with us to develop new vehicle scheduling tools that use all possible degrees of freedom. We have modelled the MDVSP correctly as a multicommodity flow problem and developed an exact mathematical method that is able to provide tight lower bounds quickly and can solve problems from practice to **proven** optimality. HanseCom has provided us access to real-world data of the Hamburger Hochbahn AG (HHA) as well as the Verkehrsbetriebe Hamburg-Holstein AG (VHH) serving the region around Hamburg. The problem instances from HHA and VHH have up to 8,563 passenger trips and about 11 million possibilities to link these trips. Computational tests using the data of HHA and VHH indicate savings of several vehicles and a cost reduction of about 10 % compared with the solution provided by the assignment approach of HOT. Our proposed solutions are currently evaluated by HHA and VHH.

In 1995, the Berliner Verkehrsbetriebe (BVG) together with the Gesellschaft für Informatik, Verkehrs- und Umweltplanung mbH (IVU) joined our research project. IVU provided us access to real-world data of BVG, which is the world's fourth largest public transportation company. Our problem instances from Berlin have up to 24,906 passenger trips and about 70 million possible links. BVG and IVU provide computer-aided tools for the planning process: BERTA, developed by IVU for BVG, and MICROBUS, developed by IVU for small and medium sized public transportation companies. The heuristical part of our tools is already integrated in BERTA and will be integrated in the next release of MICROBUS in the third quarter of this year. BVG has tested our method for the set of timetabled trips that have been serviced by one garage. These tests indicate savings of about 19 % of the vehicles and 14 % of the operational costs compared to the manual planning results. We will report in a forthcoming publication how much of these theoretical savings have been put into practice adapting the results obtained by our method for duty scheduling.

This paper describes a method that brings together academic and pragmatic requirements using new mathematical and computational knowhow to solve real-world and large-scale problems to optimality. Our method is based on improved scientific techniques from combinatorial optimization and integer programming and gains from drastically increased computing power. We have developed, implemented, and investigated various tools to solve MDVSP problems and subproblems such as primal heuristics, a network simplex algorithm with delayed column generation [Löb96a], Lagrangean relaxations [KL96], linear programming relaxations [Löb96b], Dantzig-Wolfe decomposition [Löb97a], and a branch-and-cut approach combining all the above concepts. A complete report about this project is our thesis [Löb97c].

We present our branch-and-cut approach with delayed column generation starting in Sect. 2 by describing our version of the problem and giving a multicommodity-flow formulation for it. This is followed by two reasonable integer linear programming formulations and (linear programming and Lagrangean) relaxations thereof. The next section discusses our model and compares it with solution approaches currently used in practice by many systems. Our algorithms are described in Sect. 4, followed by a description of the real-world data in Sec. 5 and a discussion of our computational results in Sect. 6.

2 The Multiple-Depot Vehicle Scheduling Problem

In the following, we give a problem description covering almost all practical requirements occurring at BVG, HHA, and VHH. A detailed problem description of our version of the MDVSP can be found in [Löb97c], we give here only a summary.

The vehicles of a transportation company are divided into so-called **depots**, denoted by $\mathcal{D}$. Each depot represents a set of vehicles that do not need to be distinguished for the operational planning process. 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, called **depot capacity**, of each depot d is denoted by κ_d.

A given set of lines together with a timetable define a set of so-called **timetabled trips** (or **passenger trips**), denoted by $\mathcal{T}$. We associate with each trip $t \in \mathcal{T}$ a first stop t^- together with a departure time s_t, a last stop t^+ together with an arrival time e_t, and a nonempty set of valid depots, called **depot-group** and denoted by $G(t) \subseteq \mathcal{D}$. Only the vehicles of the depots $G(t)$ are allowed to service the trip $t \in \mathcal{T}$. Let $\mathcal{T}_d := \{t \in \mathcal{T} | \ d \in G(t)\}$ denote all those timetabled trips being eligible for depot d, and let $\mathcal{T}^- := \{t^- | \ t \in \mathcal{T}\}$ and $\mathcal{T}^+ := \{t^+ | \ t \in \mathcal{T}\}$. There are further types of trips, all running without passengers: A **pull-out trip** connecting some start point d^+ with some first stop t^-, a **pull-in trip** connecting some last stop t^+ with some end point d^-, and a **dead-head trip** connecting some last stop t^+ with some succeeding first stop t'^-. For notational simplicity, we call them all **unloaded trips**.

For two trips t and $t' \in \mathcal{T}$, let $\Delta_{t,t'} \geqslant 0$ be given. The term $\Delta_{t,t'}$ can have different meanings: In the literature, it is mostly equivalent to the duration (travel plus layover time) from the location of the last stop of t to the location of the first stop of t', e. g., see [DP95, DFT93, RS94]. However, our partners from practice use such a definition of $\Delta_{t,t'}$ only for those dead-head trips for which the idle time or the difference $s_{t'} - e_t$ do not exceed a predefined maximum duration ranging from 40 to 120 minutes. Otherwise, $\Delta_{t,t'}$ is set to infinity. Dead-head trips exceeding the maximum allowed duration are currently not considered in practice for various practical reasons, e. g, the driver's idle time (e. g., break) would become too long etc. 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 nevertheless, we set $\Delta_{t,t'} := s_{t'} - e_t$ whenever it is possible to park a vehicle between t and t' at the depot. We call these special dead-head trips also **pull-in-pull-out trips**. They have first been described in 1980 by Bokinge and Hasselström in [BH80]; however, this concept has been considered to be an unsuited approach and has therefore been rejected.

Whenever $e_t + \Delta_{t,t'} \leqslant s_{t'}$ is satisfied, the corresponding dead-head trip is called **compatible**. Pull-in and pull-out trips are always considered to be compatible. The unloaded trips are used to interconnect the timetabled trips to **vehicle schedules**, each being 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 feasible if all its trips can be performed by at least one and the same depot. If no dead-head trip is a pull-in-pull-out trip, the vehicle schedule is also called a **block** (or a **rotation**). A block that contains only one single timetabled trip is also called a **tripper**.

It is the task to provide a set of vehicle schedules covering each trip $t \in \mathcal{T}$ exactly once by a vehicle schedule assigned to some depot of $G(t)$. The main objective – expecially of large transportation companies – is to use as less vehicles, i. e., vehicle schedules, as possible and, subordinate, with minimum operational costs among all fleet minimal solutions.

2.1 A Multi-Commodity Flow Model

We introduce for each depot $d \in \mathcal{D}$ the following sets: $A_d^{\text{t-trip}} := \{(t^-, t^+) | \ t \in \mathcal{T}_d\}$ and $A_d^{\text{u-trip}} := \{(d^+, t^-), (t^+, d^-) | \ t \in \mathcal{T}_d\} \cup \{(p^+, q^-) | \ e_p + \Delta_{p,q} \leqslant s_q\}$, where the superscripts "t-trip" and "u-trip" stand for timetabled trips and unloaded trips. For each depot $d \in \mathcal{D}$, let (d^-, d^+) denote an additional **backward arc** (on which depot capacities can be controlled) and let $A_d := A_d^{\text{t-trip}} \cup A_d^{\text{u-trip}} \cup \{(d^-, d^+)\}$. We define a digraph $D = (V, A)$ 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 gives an illustration of D for a small example with two depots $\mathcal{D} = \{\text{r,g}\}$, five timetabled trips

$\mathcal{T} = \{$a,b,c,d,e$\}$, and depot-wise trip sets $\mathcal{T}_{\mathrm{r}} = \{a,c,d\}$ and $\mathcal{T}_{\mathrm{g}} = \{a,b,c,e\}$. Note that the arc set A is a disjoint union including parallel arcs, which we have to handle carefully! For instance, addressing an arc (t^-, t^+) without knowing its depot $d \in G(t)$ may lead to confusion. If necessary, we explicitly distinguish such arcs by their corresponding depots.

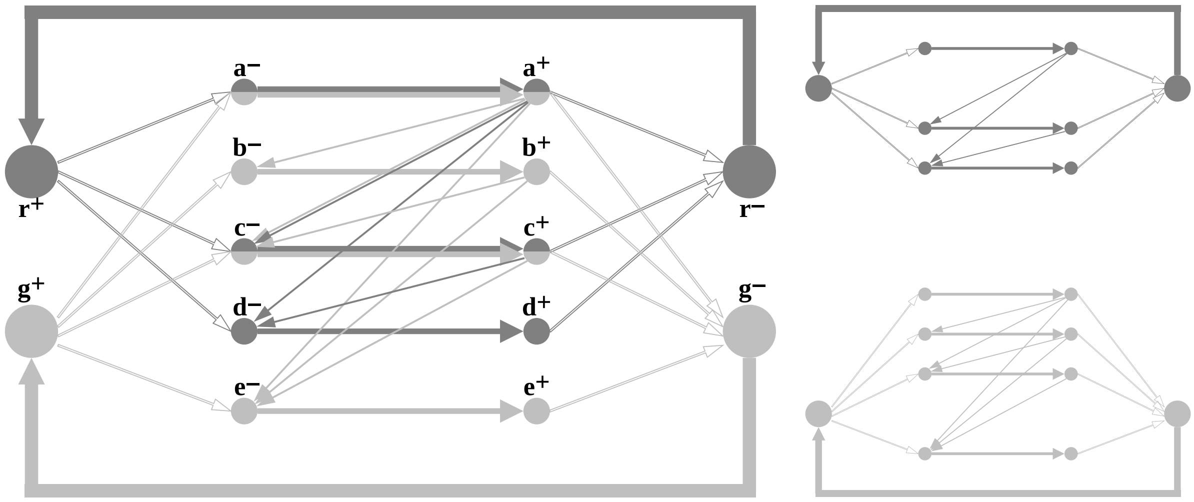

Figure 1: Digraph (V, A) and its single-depot graphs.

2.2 An Integer Linear Program

We will now state an integer linear programming (ILP) model for the MDVSP. For each $d \in \mathcal{D}$ and each $a \in A_d$, we introduce an integer variable x_a^d that denotes a decision variable indicating whether a vehicle of the depot d runs the trip a or not, except a denotes the backward arc of some depot d. In this case, x_a^d counts all employed vehicles of the depot d. The variables x_a^d are combined to vectors $x^d := (x_a^d)_{a \in A_d} \in \mathbb{R}^{A_d}$, $d \in \mathcal{D}$, and $x := (x^d)_{d \in \mathcal{D}} \in \mathbb{R}^A$.

Our two stage cost function is realized as follows: 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 **big M** standing for the capital costs and being larger than the operational costs of any feasible solution.

Given a node $v \in V$, let $\delta^+(v)$ denote all arcs of A having its tail in v and, accordingly, $\delta^-(v)$ denote all arcs having its 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 $x(\tilde{A}) := \sum_{d \in \mathcal{D}} x^d(\tilde{A})$. With respect to our multicommodity-flow model, the self-suggesting ILP reads

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

subject to

$$\begin{aligned}
x\big(\delta^+(t^-) \cap \delta^-(t^+)\big) &= 1, && \forall\, t \in \mathcal{T}, && \text{(1b)}\\
x^d\big(\delta^+(v)\big) - x^d\big(\delta^-(v)\big) &= 0, && \forall\, v \in \mathcal{T}_d \cup \{d^+, d^-\}\ \ \forall\, d \in \mathcal{D}, && \text{(1c)}\\
x^d_{(d^-,d^+)} &\leqslant \kappa_d, && \forall\, d \in \mathcal{D}, && \text{(1d)}\\
x_a^d &\geqslant 0, && \forall\, a \in A_d\ \ \forall\, d \in \mathcal{D}, && \text{(1e)}\\
& x \text{ integral.} && && \text{(1f)}
\end{aligned}$$

Note that $x\big(\delta^+(t^-) \cap \delta^-(t^+)\big) = \sum_{d \in G(t)} x^d_{(t^-,t^+)}$. Constraints (1b), the so-called **flow conditions**, ensure that each timetabled trip is serviced exactly once. Constraints (1c) are the so-called **flow conservations**.

They guarantee that any vehicle that leaves the start point d^+ must reach the end point d^- and returns via the backward arc (d^-, d^+).

From a computational point of view, the ILP (1) includes many redundant constraints that can be eliminated by performing some preprocessing steps as shown in [Löb97c]. The main idea is to shrink the two nodes t^- and t^+ to one node t, for all $t \in \mathcal{T}$, and to shrink the two nodes d^- and d^+ to one node d, for all $d \in \mathcal{D}$. This corresponds to an elimination of each arc that belongs not to some unloaded trip, see Fig. 2, and leads to the following equivalent ILP:

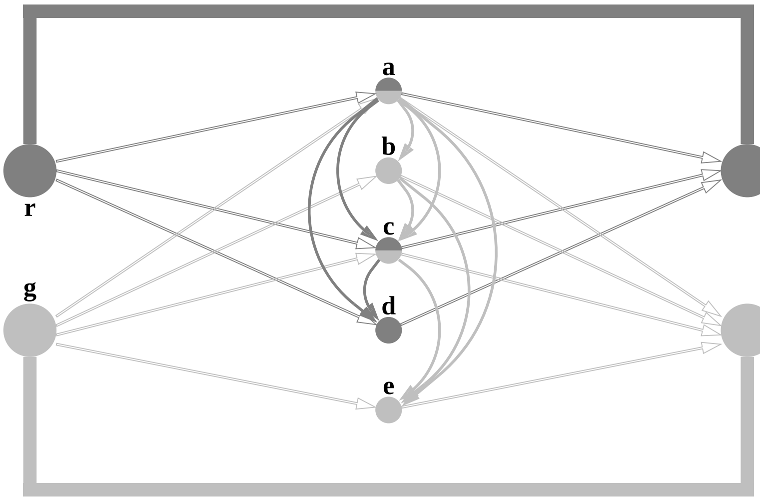

Figure 2: Contracted digraph.

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

subject to

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

2.3 Relaxations

The natural relaxation of (2) is of course its linear programming (LP) relaxation simply given by

$$\min_{\substack{x \text{ satisfying} \\ (2\text{b})-(2\text{e})}} \sum_{d \in \mathcal{D}} \sum_{a \in A_d^{\text{u-trip}}} c_a^d x_a^d. \tag{3}$$

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 (2b), (2c), and (2d). Consider a subset $\tilde{A} \subseteq A^{\text{u-trip}}$ of all unloaded trips. The LP containing just all columns corresponding with $\tilde{A}$ is called **restricted** LP of (3) and denoted by **RLP**.

We briefly give two possible Lagrangean relaxations for the MDVSP, which we use to compute tight lower bounds quickly, see [KL96], and which we use as new column generation techniques, called **Lagrangean pricing**, for a delayed column generation applied to (3), see [Löb96b]. For notational simplification, we use the same symbols for the dual variables of (3) and for the Lagrangean multipliers of the two following Lagrangean relaxations. With respect to the flow conservations (2 c) and the depot capacities (2d), the first Lagrangean function is

$$L_{\text{fcs}}(\pi,\gamma) := \min \sum_{d\in\mathcal{D}} \Big\{ \sum_{a\in A_d^{\text{u-trip}}} c_a^d\, x_a^d \; - \; \sum_{t\in\mathcal{T}_d} \pi_t^d \Big(x^d\big(\delta^+(t)\big) - x^d\big(\delta^-(t)\big)\Big) \\ - \; \gamma^d \Big(\kappa_d - x^d\big(\delta^+(d)\big)\Big)\Big\} \quad (4\,\text{a})$$

subject to

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

and the Lagrangean relaxation LR_{fcs} reads $\max_{\pi;\gamma\geqslant 0} L_{\text{fcs}}(\pi,\gamma)$. The subscript “fcs” of L_{fcs} and LR_{fcs} stands for **F**low-**C**on**S**ervation. Note, the Lagrangean function L_{fcs} is a minimum-cost flow problem. Instead of relaxing the flow conservations, it is also possible to relax the flow conditions. But unlike for the first relaxation, the second one in based on the larger ILP model (1) that also results in an efficiently solvable minimum-cost flow problem. This would not be the case if we relax the flow conditions of the ILP model (2). With respect to the flow conditions (1b), the second Lagrangean function is

$$L_{\text{fcd}}(\nu) := \nu^{\mathrm{T}}\mathbb{1} \; + \; \sum_{d\in\mathcal{D}} \min\Big\{ \sum_{a\in A_d^{\text{u-trip}}} c_a^d\, x_a^d \; - \; \sum_{t\in\mathcal{T}_d} \nu_t\, x^d_{(t^-,t^+)} \Big\}, \quad (5\,\text{a})$$

subject to

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

and the Lagrangean relaxation LR_{fcd} reads $\max_\nu L_{\text{fcd}}(\nu)$. The subscript “fcd” of L_{fcd} and LR_{fcd} stands for **F**low-**C**on**D**ition. Note that L_{fcd} decomposes into $|\mathcal{D}|$ independently solvable minimum-cost flow circulation problems.

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

3 Discussion of the Model

From a computational point of view, it is even nowadays by far still too hard to incorporate operational restrictions on the length, the duration, etc. of blocks or vehicle schedules. Such restrictions make just single-depot problems $\mathcal{NP}$-hard (e. g., see [FP95]) and can for large systems only be considered heuristically. Solution methods for the MDVSP, which are successfully employed in practice, are in most cases based on a single-commodity minimum-cost flow formulation within a **schedule first – cluster second** (SF-CS) approach (we refer the reader to [DP95] for a detailed description of this approach). This means that the multiple-depot formulation is, in principle, reduced to a single-depot formulation. However, those formulations have two significant weak points:

- Unlike to our approach, it is only possible to consider one depot independent dead-head trip between two trips. Such a dead-head trip (t,t') is considered to be feasible with respect to the depot-groups if $G(t)\cap G(t')\neq\emptyset$, i. e., depot-groups can only be considered implicitly by the definition of dead-head trips. But then, the average of the depot-groups can be empty for such a generated block leading to an infeasible solution, see Fig. 3. This infeasibility must be repaired, which may lead to suboptimal solutions.
- In most cases, dead-head trips are considered up to a maximum duration, e. g., see [DM88, DMS93], such that only blocks, but not complete vehicle schedules can be generated. Based on heuristic ideas,

Figure 3: Invalid block.

the main objective is to use as much dead-head trips as possible and, subordinate, to minimize operational costs. Obviously, this objective function does indeed minimize the number of blocks. 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 Figs. 4 and 5. The blocks, which have been determined by the SF-CS approach, are then divided to the depots and depot-wise linked to vehicle schedules. These links correspond to our pull-in-pull-out trips. It is obvious that such a problem decomposition into two successive steps inevitably leads to suboptimal solutions.

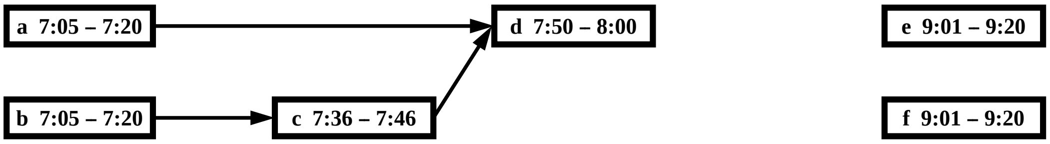

Figure 4: A **single-depot** instance for which a block minimal solution does not provide a fleet minimal solution: The maximum allowed duration of a dead-head trip is set to 60 minutes such that only the displayed dead-head trips (arrows) can be used. If we assume that the operational weight of (a, d) is significantly smaller than the weight of (c, d), it is easy to check that the blocks including the trips $\{a, d\}$, $\{b, c\}$, $\{e\}$, and $\{f\}$ define the unique block minimal solution with minimum operational weight. If we further assume that it is not possible to use a pull-in-pull-out trip to link "c" or "d" with "e" or "f", the block minimal solution requires four vehicle schedules. A fleet minimal solution, however, needs three vehicle schedules (e. g., including the timetabled trips $\{a, e\}$, $\{b, f\}$, and $\{c, d\}$), but has five blocks (including the timetabled trips $\{a\}$, $\{b\}$, $\{c, d\}$, $\{e\}$, and $\{f\}$).

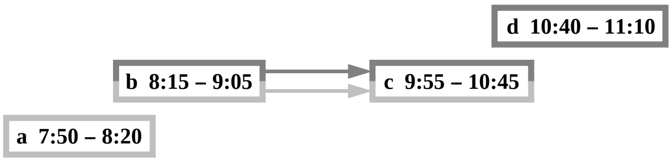

Figure 5: A **multiple-depot** instance for which a block minimal solution does not provide a fleet minimal solution: The first depot is allowed to service the trips "b, "c", and "d", and the second depot is allowed to service "a", "b", and "c". Two timetabled trips may be linked by a pull-in-pull-out 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 either to the first or to 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.

Together with the HanseCom, we have modelled the MDVSP correctly such that depot-groups can now be considered exactly and such that it is possible to simultaneously minimize the fleet size and the operational costs among all fleet minimal solutions. By the way, we independently came up with the same ILP model as already published in [FHW94].

4 Solving MDVSPs

Real-world problems of large cities 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 publically available standard software, even on the newest and fastest workstations or supercomputers. Nonetheless, with an intelligent combination of available LP and minimum-cost flow codes together with implementations of many concepts of combinatorial optimization and integer programming, it has become possible to solve such problems to optimality by delayed column generation and branch-and-cut on fast workstations. We describe below all basic ingredients that turned out to be indispensable to solve our test instances. Briefly, 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 to the integer optimum value as possible,
- primal opening heuristics as a schedule – cluster – reschedule (**SCR**) and/or a nearest depot heuristic (**ND**) to obtain a first integer feasible solution and a good starting point for
- the LP relaxation approach with a delayed column generation including new column generation techniques based on Lagrangean relaxation,
- and an LP based iterative rounding heuristic, called **LP-plunging**, to exploit information compiled in a current RLP and its solution.
- Our network simplex code MCF combined with a delayed column generation to solve single-depot instances and evaluate our Lagrangean functions, and the LP solver CPLEX to solve the RLPs.

Our basic method to solve the MDVSP is to solve the integer linear programming formulation by primal and dual heuristics, delayed column generation, and branch-and-cut, see also Fig. 6.

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 delayed 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 $\frac{c_U}{L_{\text{fcs}}(0)}$ 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 almost each test instance an optimal solution or a fleet minimal 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 a main memory limit) 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 occur 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 the important part of our method. Therefore, it is not the scope of this article to describe it here, but we refer the interested reader to [Löb97c].

We have also investigated a Dantzig-Wolfe decomposition. It turned out that such a decomposition approach is an unsuitable method to solve the MDVSP. The major obstacle here is that the continuous master problem relaxations become too hard to be solved 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. We refer the reader to [Löb97a] or [Löb97c] for a detailed description and discussion of this investigation.

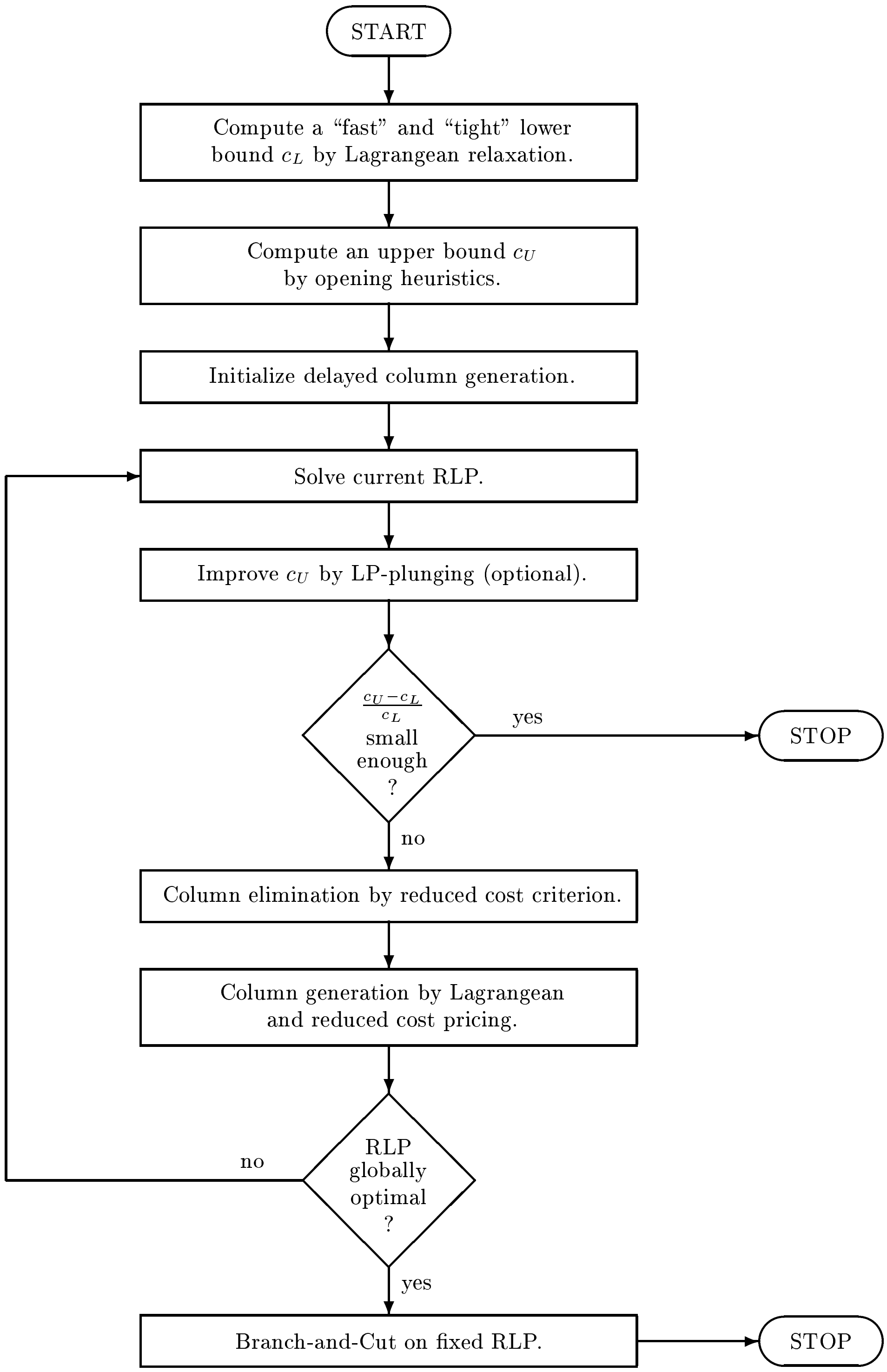

Figure 6: Solving MDVSPs: Flow chart.

4.1 Lagrangean Relaxations

The first important use of the Lagrangean relaxation $\mathrm{LR}_{\mathrm{fcs}}$ and $\mathrm{LR}_{\mathrm{fcd}}$ is to quickly compute lower bounds as close to the integer optimum value as possible. We have shown in [KL96] that our real-world test instances seem to be fairly well structured: For instance, the trivial problem relaxation $L_{\mathrm{fcs}}(0)$, which is equivalent to the neglection of the flow conservation constraints, gives already 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 let c^* denote the optimal integer solution value. Since $0 \leqslant L_{\mathrm{fcs}}(0) \leqslant c^* \leqslant c_U$, the percentage deviation between c_U and c^* can be approximated and estimated by

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

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

Another important use of $\mathrm{LR}_{\mathrm{fcs}}$ and $\mathrm{LR}_{\mathrm{fcd}}$ is Lagrangean pricing, which is a new column generation technique to solve the LP relaxation (3), see below.

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 that simply assigns each trip $t \in \mathcal{T}$ to one of its depots of $G(t)$ that provides the cheapest costs for its pull-out and pull-in trips (cluster part).
- 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 summarized (or contracted) to single depot nodes $\mathcal{D}$ and the depot individual flow conservations are aggregated to one common flow conservation. This relaxed system is solved to optimality. The resulting vehicle schedules may violate some individual flow conservations. These violations are repaired heuristically: The vehicle schedules are pieced to feasible blocks defining a cluster that considers depot capacities as good as possible. 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, which cause the infeasibility of a current (relaxed) solution.

Complexity theory tells us that it is $\mathcal{NP}$-hard to just find a feasible solution if depot capacities are considered, see [Löb97c]. However, it turned out that depot capacities are not always hard constraints and can thus sometimes be violated since vehicles can often be shifted from one garage (or depot) to another.

4.3 Solving the LP Relaxation

Surprisingly, 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 relaxations exactly. Standard software alone (e. g., CPLEX pure) as well as standard approaches from integer linear programming (e. g., a delayed column generation and column elimination schemes based on the reduced cost criterion) are unable to solve larger instances with several thousand timetabled trips. We are not aware of any application of a delayed column generation that uses other column generation schemes than those based on the reduced cost criterion. We have developed a delayed column generation method including to the best of our knowledge **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.

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

Let $\tilde{\nu}$, $\tilde{\pi}$, and $\tilde{\gamma}$ denote the values of the dual multipliers according to the flow conditions (2b), the flow conservations (2c), and the depot capacities (2d) of the last basis of the current RLP. The Lagrangean functions L_{fcs} and L_{fcd} can be solved very efficiently – even for the complete variable set – and give very good approximations of the LP relaxation. We evaluate L_{fcs} and L_{fcd} at $(\tilde{\pi}, \tilde{\gamma})$ and $\tilde{\nu}$, respectively. The solution of each Lagrangean function can be interpreted as a set of vehicle schedules 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 exceptionally new idea of our column generation strategies is to generate, in addition to columns with negative reduced costs, also those that have nonnegative reduced costs, but are just necessary to complete (almost) optimal solutions. For more details, we refer the reader to [Löb96b] or [Löb97c].

4.4 LP-Plunging

Our real-world MDVSP instances show a very nice integer property: Solutions x of the LP relaxation (3) or an RLP thereof include not so many fractional variables. Often, x is already integral or there exists some integral solution yielding (almost) the same objective value. Moreover, the gap between the optimal LP value and the optimal integer value is often small or already zero. This property of real-world problems was also observed and described by [FHW94]. LP-plunging makes use of this property by iteratively rounding up and fixing components of the LP solution and reoptimizing the enlarged LP.

Given some LP (3) or an RLP thereof 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 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 RLP 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 $\alpha\Delta$ is still greater than 0.5, we reset $\Delta := \alpha\Delta$ and go to 1. Otherwise, the value of every fractional variable is not larger than 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 iterative rounding succeeds, the clustering defined by the (last) integral x is depot-wise rescheduled to optimality using all possible unloaded trips of each depot.

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 exactly requires at several steps the efficient solution of minimum-cost flow problems and linear programs: 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 and combined it with a delayed column generation. This implementation, called MCF, allows to solve the single-depot problems and subproblems to optimality in a few seconds. The Lagrangean functions can, depending on the problem size, also be evaluated in a few seconds up to a few minutes. For instance, the Lagrangean function L_{fcs} of the problem with 70 million unloaded trips can be exactly evaluated in about 15 minutes. For detailed information, we refer the reader to [Löb97c]. MCF is available free of charge for academic use via WWW at URL: http://www.zib.de/Optimization, see [Löb97b].

We solve the linear programs with the primal as well as the dual simplex solver of CPLEX, currently version 4.0.9 [CPL95]. 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 test instances from the city of Berlin (BVG), the city of Hamburg (HHA), and the region around Hamburg (VHH). Table 1 gives the characteristics of our test set. The average size of the depot-group $G(t)$ for a timetabled trip $t \in \mathcal{T}$ is denoted by $\varnothing|G(t)|$. Besides the total number of arcs in $A^{\text{u-trip}}$, we also give the number of unloaded trips without pull-in-pull-out trips, which we denote as user-defined trips. The number of flow conditions and conservations #(2b) + #(2c) is simply given by $(\varnothing|G(t)|+1)|\mathcal{T}|$.

Test Sets	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert A^{\text{u-trip}}\rvert$		$\varnothing\lvert G(t)\rvert$	#(2b) + #(2c)
			User-def.[a]	All		
Berlin 1	44	24,906	846,000	69,700,000	4.03	125,255
Berlin 2	49	24,906	304,000	13,200,000	1.56	63,641
Berlin 3	3	1,313	77,000	2,300,000	2.33	4,370
Berlin-Spandau 1	9	2,424	164,000	3,700,000	4.94	14,418
Berlin-Spandau 2	9	3,308	327,000	8,800,000	5.49	21,470
Berlin-Spandau 3	13	2,424	39,000	590,000	1.92	7,103
Berlin-Spandau 4	13	3,308	72,000	1,530,000	2.25	10,753
Berlin-Spandau 5	13	3,331	75,000	1,550,000	2.25	10,834
Berlin-Spandau 6	13	1,998	28,000	380,000	1.90	5,798
Berlin-Spandau 7	7	2,424	145,000	3,300,000	4.16	12,506
Berlin-Spandau 8	7	3,308	283,000	7,800,000	5.02	18,376
Hamburg 1	12	8,563	1,322,000	10,900,000	2.23	27,696
Hamburg 2	9	1,834	99,000	1,000,000	2.02	5,549
Hamburg 3	2	791	30,000	200,000	1.32	1,835
Hamburg 4	2	238	2,000	23,000	1.04	487
Hamburg 5	2	1,461	85,000	580,000	1.31	3,379
Hamburg 6	2	2,283	176,000	1,600,000	1.33	5,323
Hamburg 7	2	341	6,000	34,000	1.32	795
Hamburg-Holstein 1	4	3,413	230,000	4,000,000	1.68	9,167
Hamburg-Holstein 2	19	5,447	1,054,000	9,400,000	3.65	25,334

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

Table 1: Real-world test sets.

BVG Problems.

Berlin 1: At present, BVG maintains 9 garages and runs 10 different vehicle types (2 double-decker types, 6 single-decker types, and 3 articulated bus types). Combining the garages with their available vehicle types results in 44 different depots. There are no depots containing vehicles of different garages. For a normal weekday, about 28,000 timetabled trips have to be serviced. This number reduces to 24,906

timetabled trips since, for instance, some of the timetabled trips are out-sourced to third-party companies. The number of unloaded trips for our largest Berlin instance is about 70 millions from which about 846 thousand are explicitly user-defined.
Berlin 2: Since the LP relaxation for Berlin 1 is extremely large – more than 125 thousand equations – we have generated a smaller problem with more depots, but much fewer unloaded trips. This problem is based on the same set of timetabled trips as the problem 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 relatively small test instance including 9 lines from the south of Berlin and 3 depots from one single garage.
Berlin-Spandau: 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 Problems.

At the moment, HHA together with other transportation companies maintain 14 garages with 9 different vehicle types. The existing garage-vehicle combinations define 40 depots. In Hamburg, more than 16,000 daily timetabled trips must be scheduled with about 15.1 million unloaded trips. The problem in Hamburg decomposes into a 12-depot problem, a 9-depot problem, five smaller 2-depot problems, and nine small 1-depot problems. We consider only the multiple-depot problems, denoted by Hamburg 1–7.

VHH Problems.

VHH currently plans 10 garages with 9 different vehicle types. The garage-vehicle combination defines 19 depots. This problem, denoted by Hamburg-Holstein 2, is our latest test set. Hamburg-Holstein 1 is an older test set containing a subset of the depots and trips of VHH.

6 Computational Results

All our 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 [Löb96a, Löb97b] combined with a delayed column generation.

Tables 2, 3, and 4 summarize our computational results. The objective values (fleet sizes and operational weights) are displayed in Tab. 2: First, the lower bounds obtained by the Lagrangean and LP relaxations; second, the upper bounds obtained by our opening heuristics as well as obtained by our method starting with SCR or ND and terminating after a maximum running time limit of 10 hours (and 16 hours for Berlin 1 starting with ND); third, the optimal integer values or, if the optimum is still unknown, the best integer solution values. 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. In Tab. 3, we give the lower bound values in proportion to the integer optimal values (or best known upper bound values) and the upper bound values in proportion to the integer optimal values (or the best known lower bound values). Note that the proportion for the operational weights is only considered for problems if the corresponding fleet size gap is equal to zero. The running times to solve the Lagrangean function $L_{\text{fcs}}(0)$, 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 delayed column generation) are given in Tab. 4

Lower Bounds.

The values obtained by Lagrangean relaxation, i. e., evaluating the Lagrangean function $L_{\text{fcs}}(0)$ in our case, give good approximations (on the average, 99.94 %) for the minimum integral fleet sizes. For 15 of our 20 instances, the fleet size can be approximated exactly. Ignoring for those problems the values for

Table 2: Vehicle demands and operational weights (optimal integer values are in bold face).

Test Sets	Lower bounds: Lagrangean relax. $L_{\mathrm{fcs}}(0)$[a]		LP relaxation[b]		Optimum or best int. solution		Upper bounds: Schedule – cluster – reschedule, pure[c]		Schedule – cluster – reschedule, + LP method[d]		Nearest depot heuristic, pure[c]		Nearest depot heuristic, + LP method[d]	
	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight
Berlin 1	1323	715714	1323	759162.0[e]	1329	850680	1347	1317379	1335	1118287	1575	3279774	1356	982914
Berlin 2	1350	715623	1353.7	797918.8	**1354**	777823	1366	1318085	1354	809611	1655	3900191	1354	788958
Berlin 3	**69**	14043	**69**	**14119**	**69**	**14119**	**69**	14122	**69**	**14119**	70	14366	**69**	**14119**
B-Spandau 1	**125**	65585	**125**	65610.5	**125**	**65611**	**125**	125786	**125**	65835	139	70092	**125**	65901
B-Spandau 2	184	78947	184.5	79110.0	**185**	79052	**185**	289262	**185**	80430	207	213333	**185**	92249
B-Spandau 3	**127**	90514	**127**	**93745**	**127**	**93745**	**127**	152109	**127**	**93745**	135	149629	**127**	**93745**
B-Spandau 4	**191**	195844	**191**	**230846**	**191**	**230846**	192	395891	**191**	**230846**	222	475552	**191**	**230846**
B-Spandau 5	**191**	191141	**191**	**227580**	**191**	**227580**	194	393922	**191**	**227580**	220	487944	**191**	**227580**
B-Spandau 6	**98**	91109	**98**	**101075**	**98**	**101075**	**98**	132650	**98**	**101075**	109	139593	**98**	**101075**
B-Spandau 7	**125**	65585	**125**	65610.5	**125**	**65611**	**125**	105853	**125**	**65611**	139	70092	**125**	65724
B-Spandau 8	184	78947	184.5	79110.0	**185**	79093	**185**	259406	**185**	79273	207	213333	**185**	79959
Hamburg 1	**432**	66874	**432**	71068.3	**432**	**71069**	446	70291	434	73066	489	76054	**432**	71270
Hamburg 2	**103**	15356	**103**	**16070**	**103**	**16070**	104	16792	**103**	**16070**	114	15849	**103**	**16070**
Hamburg 3	**39**	5557	**39**	**5860**	**39**	**5860**	**39**	6298	**39**	**5860**	41	5429	**39**	**5860**
Hamburg 4	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**
Hamburg 5	**62**	12092	**62**	**12502**	**62**	**12502**	**62**	13535	**62**	**12502**	65	12101	**62**	**12502**
Hamburg 6	**111**	15705	**111**	**15791**	**111**	**15791**	**111**	16588	**111**	**15791**	**111**	**15791**	**111**	**15791**
Hamburg 7	**15**	2832	**15**	**2961**	**15**	**2961**	16	2836	**15**	**2961**	16	2836	**15**	**2961**
H-Holstein 1	**201**	28697	**201**	**29027**	**201**	**29027**	**201**	30497	**201**	**29027**	213	32132	**201**	**29027**
H-Holstein 2	360	51084	**362**	52787.7	**362**	**52788**	363	72700	**362**	**52986**	393	62598	**362**	**53090**

[a] See also [KL96].

[b] See also [Löb96b].

[c] Results obtained with the opening heuristics SCR and ND, respectively.

[d] Results obtained with our LP method including branch-and-cut, starting with SCR or ND, using LP-plunging, and terminating after a running time limit of ten hours (and 16 hours for Berlin 1 starting with ND).

[e] Best known feasible LP value.

Table 3: Lower bounds relative to the integer optimal values (or best upper bounds), and upper bounds relative to the integer optimal values (or best lower bounds). Optimal values are in bold face.

Test Sets	Lower bounds: Optimum or Best integer solution				Upper bounds: Optimum or Best integer solution							
	Lagrangean relaxation $L_{\mathrm{fcs}}(0)$		LP relaxation		Schedule – cluster – reschedule: pure[a]		Schedule – cluster – reschedule: + LP method[b]		Nearest depot heuristic: pure[c]		Nearest depot heuristic: + LP method[d]	
	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight
Berlin 1	0.9954	—	0.9954	—	1.0181	—	1.0091	—	1.1851	—	1.0249	—
Berlin 2	0.9970	—	0.9997	—	1.0089	—	**1.0000**	—	1.2223	—	**1.0000**	—
Berlin 3	**1.0000**	0.9946	**1.0000**	**1.0000**	**1.0000**	1.0002	**1.0000**	**1.0000**	1.0145	—	**1.0000**	**1.0000**
B-Spandau 1	**1.0000**	0.9996	**1.0000**	1.0000	**1.0000**	1.9171	**1.0000**	1.0034	1.1120	—	**1.0000**	1.0044
B-Spandau 2	0.9946	—	0.9973	—	**1.0000**	—	**1.0000**	—	1.1189	—	**1.0000**	—
B-Spandau 3	**1.0000**	0.9655	**1.0000**	**1.0000**	**1.0000**	1.6225	**1.0000**	**1.0000**	1.0630	—	**1.0000**	**1.0000**
B-Spandau 4	**1.0000**	0.8484	**1.0000**	**1.0000**	1.0052	—	**1.0000**	**1.0000**	1.1623	—	**1.0000**	**1.0000**
B-Spandau 5	**1.0000**	0.8399	**1.0000**	**1.0000**	1.0157	—	**1.0000**	**1.0000**	1.1518	—	**1.0000**	**1.0000**
B-Spandau 6	**1.0000**	0.9014	**1.0000**	**1.0000**	**1.0000**	1.3124	**1.0000**	**1.0000**	1.1122	—	**1.0000**	**1.0000**
B-Spandau 7	**1.0000**	0.9996	**1.0000**	1.0000	**1.0000**	1.6133	**1.0000**	**1.0000**	1.1120	—	**1.0000**	1.0017
B-Spandau 8	0.9946	—	0.9973	—	**1.0000**	—	**1.0000**	—	1.1189	—	**1.0000**	—
Average	0.9983	0.9356	0.9991	1.0000	1.0044	—	1.0008	1.0005	1.1248	—	1.0023	1.0009
$\sqrt{\sigma_n}$	0.0023	0.0660	0.0015	0.0000	0.0065	—	0.0026	0.0012	0.0538	—	0.0072	0.0016
Hamburg 1	**1.0000**	0.9410	**1.0000**	1.0000	1.0324	—	1.0046	—	1.1319	—	**1.0000**	1.0028
Hamburg 2	**1.0000**	0.9556	**1.0000**	**1.0000**	1.0097	—	**1.0000**	**1.0000**	1.1068	—	**1.0000**	**1.0000**
Hamburg 3	**1.0000**	0.9483	**1.0000**	**1.0000**	**1.0000**	1.0747	**1.0000**	**1.0000**	1.0513	—	**1.0000**	**1.0000**
Hamburg 4	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Hamburg 5	**1.0000**	0.9672	**1.0000**	**1.0000**	**1.0000**	1.0826	**1.0000**	**1.0000**	1.0484	—	**1.0000**	**1.0000**
Hamburg 6	**1.0000**	0.9946	**1.0000**	**1.0000**	**1.0000**	1.0505	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Hamburg 7	**1.0000**	0.9564	**1.0000**	**1.0000**	1.0667	—	**1.0000**	**1.0000**	1.0667	—	**1.0000**	**1.0000**
H-Holstein 1	**1.0000**	0.9886	**1.0000**	**1.0000**	**1.0000**	1.0506	**1.0000**	**1.0000**	1.0597	—	**1.0000**	**1.0000**
H-Holstein 2	**0.9945**	—	**1.0000**	1.0000	1.0028	—	**1.0000**	1.0038	1.0856	—	**1.0000**	1.0057
Average	0.9994	0.9690	1.0000	1.0000	1.0124	1.0517	1.0005	1.0005	1.0612	—	1.0000	1.0009
$\sqrt{\sigma_n}$	0.0017	0.0211	0.0000	0.0000	0.0216	0.0288	0.0014	0.0013	0.0414	—	0.0000	0.0019
All Average	0.9988	0.9534	0.9995	1.0000	1.0080	—	1.0007	1.0005	1.0962	—	1.0012	1.0009
All $\sqrt{\sigma_n}$	0.0021	0.0505	0.0012	0.0000	0.0158	—	0.0021	0.0012	0.0580	—	0.0054	0.0018

[a]Results obtained by the opening heuristics SCR and ND, respectively.

[b]Results obtained with our LP method including branch-and-cut, starting with SCR or ND, using LP-plunging, and terminating after a running time limit of ten hours (and 16 hours for Berlin 1 starting with ND).

Table 4: Running times in seconds on a SUN Model 170 UltraSPARC with 512 MByte main memory.

Test Sets	Lower bounds			Upper bound (or optimum)					
	Lagrangean relaxation: $L_{\mathrm{fcs}}(0)$	LP relaxation[a] starting with		Schedule – cluster – reschedule			Nearest depot heuristic		
		SCR	ND	pure	Exact method LP-plunging on[b]	Exact method LP-plunging off[c]	pure	Exact method LP-plunging on[b]	Exact method LP-plunging off[c]
Berlin 1	916	—[d]	—[d]	12386	—[d]	—[d]	171	—[d]	—[d]
Berlin 2	229	34795	32767	3810	—[d]	35202[e]	79	30985[e]	33248[e]
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	—[d]	138212[e]	39	—[d]	240852[e]
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[e]	84725[e]	36	—[d]	63215[e]
Hamburg 1	185	—[d]	50246	2868	—[d]	—[d]	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

[a] LP times pure without LP-plunging.
[b] Our method using LP-plunging.
[c] Our method using LP-plunging only within branch-and-cut, but not between two successive RLPs.
[d] Not solved to optimality since, for instance, the objective progress was too small or stalling occurred.
[e] The problem has only be solved fleet minimally, but not to proven cost minimality.

the fleet size, the gap between the operational costs of $L_{\text{fcs}}(0)$ and the optimum is at most 16 % and on the average 5 %. From our point of view, these gaps are acceptable, but we have shown in [KL96] that they can be significantly decreased using in addition a computationally expensive subgradient method. The computing times are quite fast: For Berlin 1, $L_{\text{fcs}}(0)$ with 70 million variables can be evaluated in about 15 minutes; for all the other instances together, it can be computed in 15 minutes.

The values obtained by the LP relaxation give lower bounds quite close to the integer optimal values. For 16 of our 20 instances, the LP relaxation provides already the integer optimal value, and for 3 instances, rounding up the LP value to the next integer value provides the integer optimum value. For Berlin 1, we do not know the minimal number of vehicles and can, thus, not make any statement for this instance. On the average, the fleet size is almost exactly approximated by the LP relaxation. If the LP relaxation provides the exact fleet size for an instances of our test set, it also provides the minimal operational costs. A similar phenomenon is observed by [FHW94]: 22 of their 30 test instances have integral LP solutions, and for the other 8 problems the largest gap between the LP value and the integral optimum is 0.003 %. All LP relaxations, except for Berlin 1, have been solved to optimality. To find a fleet minimal LP value for Berlin 1, our delayed column generation requires about 200 hours cpu time.

For the two relaxations, we can make statements about the quality of the operational costs only if the fleet size equals the integer optimum. Otherwise, from a theoretical point of view, the operational cost of the lower bounds are pointless. However, we strongly believe that they nevertheless give good estimated values for the minimal operational costs.

Upper Bounds.

We have computationally investigated our two opening heuristics SCR and ND as well as our exact method starting with SCR and ND, using LP-plunging between two RLPs, and terminating after a given running time limit of 10 hours (and 16 hours for Berlin 1 starting with ND).

The trivial opening heuristic ND delivers already 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, we consider the operational costs of these solutions not as acceptable. Better results are obtained by 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 codes currently used in practice. We receive almost always optimal results if we apply our exact method. The fleet and cost gaps are, on the average, not larger than 0.12 %. It does not make any difference which opening heuristic we use for the exact method since the running times are for both comparable. The running times of our exact method may be improved if we use both opening heuristics together to determine the first RLP. This may be the scope of further computational tests.

Figures. 7–10 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 ND, the fleet size 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 for Berlin 1, see Fig. 7. Starting with SCR, the fleet size can be approximated in one hour with a gap less than 2 % and in 10 hours with a gap less than 1 %, see Fig. 9. There is also a positive development of the operational costs: Compared to the optimal integer costs of fleet minimal solutions, the operational costs can be approximated with a small gap, see Figs. 8 and 10. If the running time limit is 10 hours or more, the four figures conform that it does not make any difference which opening heuristic is uses, the results are in any case comparable. If, however, there is a stronger run time limit of two or three hours, starting with SCR provides the better results than starting with ND.

Optimal Solutions

With the exception of the problem Berlin 1, each instance of our test set can be solved to proven fleet minimality, and with the exception of the problems Berlin 2, Berlin-Spandau 2, and Berlin-Spandau 8, each instance can be solved to proven optimality for our two-stage objective function.

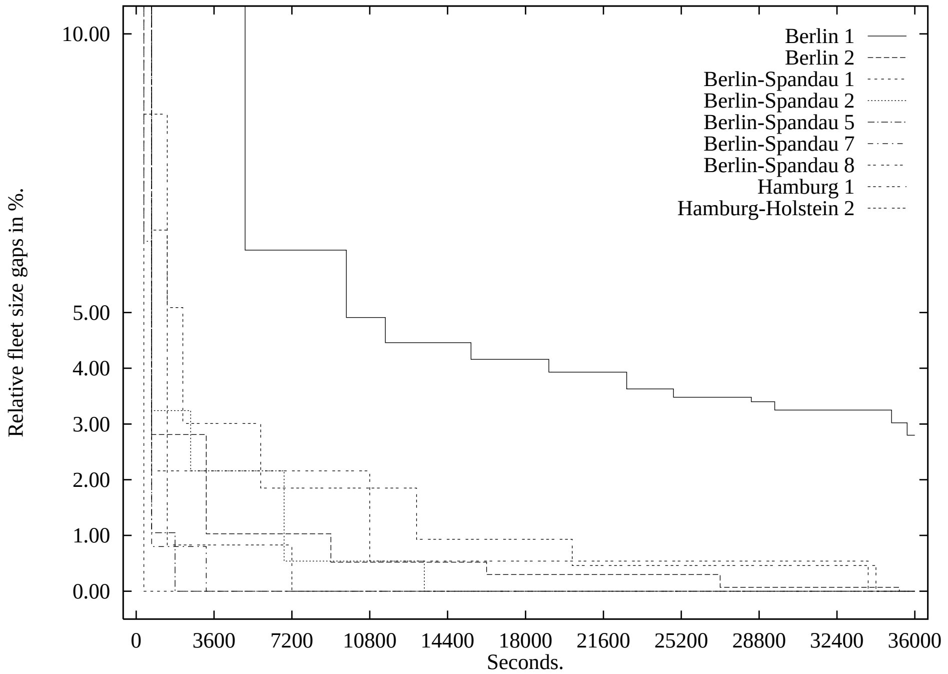

Figure 7: Development of upper bound values (**fleet size**) of problems requiring more than 2 hours running time to obtain a fleet minimal solution or the optimum, starting our method with the ND heuristic.

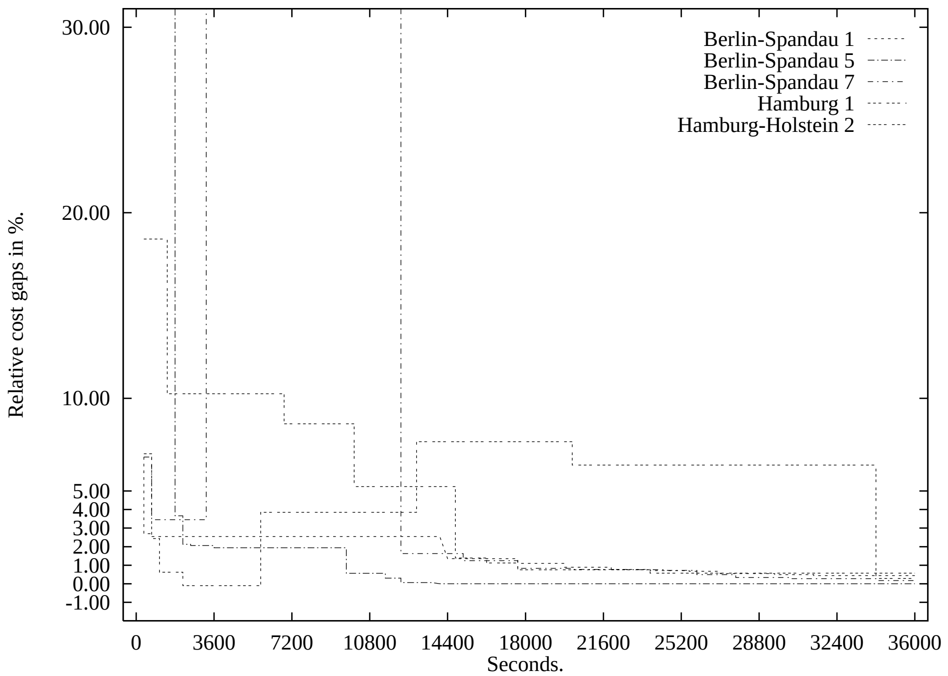

Figure 8: Development of upper bound values (**operational weights**) of problems (i) requiring more than 2 hours running time to obtain a fleet minimal solution or the optimum and (ii) knowing the minimum weight among all fleet minimally solutions; starting our method with the ND heuristic.

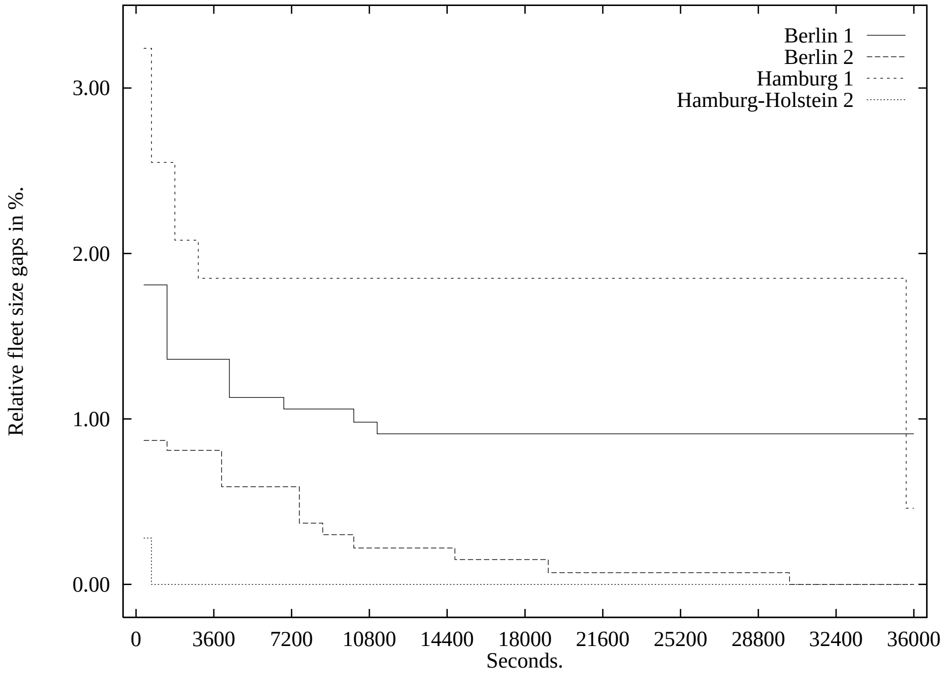

Figure 9: Development of upper bound values (**fleet size**) of problems requiring more than 2 hours running time to obtain the optimum, starting our method with the SCR heuristic.

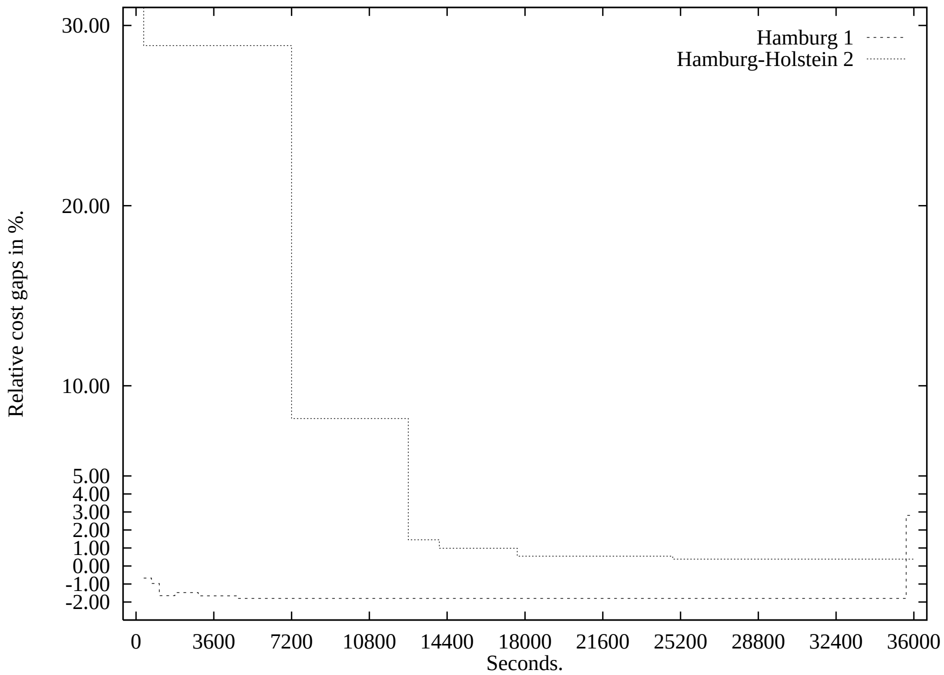

Figure 10: Development of upper bound values (**operational weights**) of problems (i) requiring more than 2 hours running time to obtain the optimum and (ii) knowing the minimum weight among all fleet minimally solutions; starting our method with the SCR heuristic.

Solving really large-scale problems to optimality still leads to unpractical running times. Especially Berlin 1 with 70 million variables is still a challenge to us to be solved to proven optimality. Nevertheless, the results obtained with our methods are currently the best that are possible. Solutions providing possibly a gap of few vehicles, but with reasonable operational weights can be computed in acceptable running times.

7 Conclusions

The Lagrangean relaxation provide tight lower bounds that can be quickly computed in few minutes. Public transportation companies can use this tool to simulate different scenarios in vehicle scheduling, e. g., different compatibility rules for or different weights of unloaded trips, and so easily find out a proper scenario.

With our delayed column generation and the new Lagrangean pricing scheme, it becomes possible to solve the LP relaxation and its integer version of large-scale problems within acceptable running times on a workstation to proven optimality or, at least, with an acceptable small gap. However, truly large problems as the complete BVG system using all degrees of freedom are still an open challenge to be solved to optimality. The solution generated by LP-plunging within few hours are currently the best possible.

The schedule – cluster – reschedule heuristic is already integrated in the BERTA system of the Berliner Verkehrsbetriebe and is being used since the beginning of this year 1997. Unfortunately, we can hitherto give no results of saving put into practice by using our method. Tests indicate savings of about 19 % of the vehicles and about 14 % of the operational costs compared to a manual planning and savings of several vehicles and about 10 % cost reduction compared to the solutions provided by the assignment approach of HOT. However, the rework of our solutions by BVG, HHA, and VHH to become a proper input for the duty scheduling have not been finished hitherto, but we will report on these results in a forthcoming publication.

8 Acknowledgements

This work has been supported by the German Federal Ministry of Education, Science, Research, and Technology grant no. 03-GR7ZIB -7. 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 Hamburger Hochbahn AG, the Verkehrsbetriebe Hamburg-Holstein AG, and the Berliner Verkehrsbetriebe for their kind permission to use and publish their data. We are indebted to our colleagues Ralf Borndörfer and Alexander Martin (both at ZIB) and Bob Bixby (Rice University, Houston, TX, and CPLEX Optimization, Inc.) for their helpful discussions. We are also grateful to Bob Bixby and CPLEX Optimization, Inc., for regularly providing us access to the newest beta versions of CPLEX and the CPLEX Callable Library.

References

[BH80] U. Bokinge and D. Hasselström. Improved vehicle scheduling in public transport through systematic changes in the time-table. *European Journal of Operational Research*, 5:388–395, 1980.

[BMMN95] M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors. *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*. Elsevier Science B.V., Amsterdam, 1995.

[BW97] Thomas Böhmig and Malte Wolter. Kostenträgerrechnung für ÖPNV-Betriebe. *Der Nahverkehr*, 1–2/97:14–19, 1997. In German.

[BWZ97] Michael Bussieck, Thomas Winter, and Uwe T. Zimmermann. Discrete optimization in public rail transport. Preprint, Technical University of Braunschweig, 1997. Available at URL: http://www.math.tu-bs.de/mo/research/preprints.html.

[CPL95] CPLEX Optimization, Inc., Suite 279, 930 Tahoe Blvd., Bldg 802, Incline Village, NV 89451, USA. *Using the CPLEX Callable Library*, 1995. Information available via WWW at URL: http://www.cplex.com/.

[DBP95] Joachim R. Daduna, Isabel Branco, and José M. Pinto Paixão, editors. *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag, 1995.

[DDSS95] Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and Francois Soumis. *Time Constrained Routing and Scheduling*, chapter 2, pages 35–139. Volume 8 of Ball et al. [BMMN95], 1995.

[DFT93] M. Dell'Amico, M. Fischetti, and P. Toth. Heuristic algorithms for the multiple depot vehicle scheduling problem. *Management Science*, 39(1):115–125, 1993.

[DM88] Joachim R. Daduna and Miodrag Mojsilovic. Computer-aided vehicle and duty scheduling using the HOT programme system. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems, pages 133–146, 1988.

[DMS93] Joachim R. Daduna, Miodrag Mojsilovic, and Peter Schütze. Practical experiences using an interactive optimization procedure for vehicle scheduling. In Ding-Zhu Du and Panos M. Pardalos, editors, *Network Optimization Problems: Algorithms, Applications and Complexity*, volume 2 of *Series on Applied Mathematics*, pages 37–52. World Scientific Publishing Co. Pte. Ltd., Singapore, New York, London, 1993.

[DP95] Joachim R. Daduna and José M. Pinto Paixão. Vehicle scheduling for public mass transit – an overview. In Daduna et al. [DBP95].

[FHW94] M. A. Forbes, J. N. Holt, and A. M. Watts. An exact algorithm for multiple depot bus scheduling. *European Journal of Operational Research*, 72(1):115–124, 1994.

[FP95] Richard Freling and José M. Pinto Paixão. Vehicle scheduling with time constraint. In Daduna et al. [DBP95].

[KL96] Andreas Kokott and Andreas Löbel. Lagrangean relaxations and subgradient methods for multiple-depot vehicle scheduling problems. Preprint SC 96-22, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996. Available at URL: http://www.zib.de/ZIBbib/Publications/.

[KL97] Caroline Kretschmann and Rainer Lawerentz. Produktmanagement im ÖPNV. *Der Nahverkehr*, 1–2/97:8–13, 1997. In German.

[Löb96a] Andreas Löbel. Solving large-scale real-world minimum-cost flow problems by a network simplex method. Preprint SC 96-7, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996. Available at URL: http://www.zib.de/ZIBbib/Publications/.

[Löb96b] Andreas Löbel. Solving the LP relaxations of large-scale multiple-depot vehicle scheduling problems exactly. Preprint SC 96-26, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996. Available at URL: http://www.zib.de/ZIBbib/Publications/.

[Löb97a] Andreas Löbel. Experiments with a Dantzig-Wolfe-decomposition for multiple-depot vehicle scheduling problems. Preprint SC 97-16, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997. Available at URL: http://www.zib.de/ZIBbib/Publications/.

[Löb97b] Andreas Löbel. *MCF Version 1.0 – A network simplex implementation*, 1997. Available for academic use free of charge via WWW at URL: http://www.zib.de/Optimization.

[Löb97c] Andreas Löbel. *Optimal Vehicle Scheduling in Public Transit*. PhD thesis, Technische Universität Berlin, 1997. To appear.

[RS94] Celso C. Ribeiro and Francois Soumis. A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, 42(2):41–52, 1994.