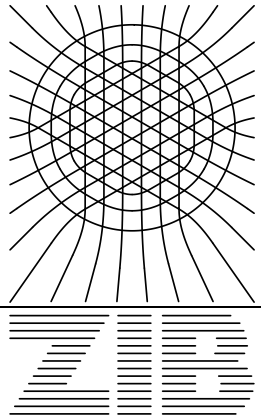

Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7, D-14195 Berlin-Dahlem, Germany

Andreas Löbel

Vehicle Scheduling in Public Transit and Lagrangean Pricing

Revised Preprint SC 96–26 (December 15, 1997)

Vehicle Scheduling in Public Transit and Lagrangean Pricing*

Andreas Löbel**

December 15, 1997

Abstract

This paper investigates the solution of the linear programming (LP) relaxation of the multicommodity flow formulation of the multiple-depot vehicle scheduling problems arising in public mass transit. We develop a column generation technique that makes it possible to solve the huge linear programs that come up there. The technique, which we call *Lagrangean pricing*, is based on two different Lagrangean relaxations.

We describe in detail the basic ingredients of our approach and give computational results for large-scale test data (with up to 70 million variables) from three German public transportation companies. Because of these results, we propose Lagrangean pricing as one of the basic ingredients of an effective method to solve multiple-depot vehicle scheduling problems to *proven* optimality.

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

1 Introduction

The multiple-depot vehicle scheduling problem comes up as one step of a *hierarchical planning process in public transit* consisting of line planning, timetabling, vehicle scheduling, and crew scheduling and rostering. Each of these individual subproblems is itself hard and of large scale. The economic significance of these problems requires their solution to optimality. For a survey on optimization in public transit see, e. g., Daduna, Branco, and Paixão [1995] and the references therein.

This paper deals with solving large-scale *multiple-depot vehicle scheduling problems* in public transit (MDVSP). The literature on this topic discusses a large number of possible approaches, both heuristic and exact, see, e. g., Daduna and Paixão [1995] for an overview.

The most successful solution approaches are based on *network flow models* and their integer programming analogues. In the literature, there are two basic mathematical models of this type for the MDVSP. 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 by a 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. For the direct methods, column generation can be seen as an implicit

*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, URL: `http://www.zib.de`

pricing technique, see Schrijver [1989]: one works on restricted subsets of active arcs, which are generated and eliminated in a dynamic process. For the Dantzig-Wolfe 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 Dantzig-Wolfe 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, Dantzig-Wolfe 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.

Dantzig-Wolfe decomposition models are needed for problems that involve path constraints. They apply not only to vehicle scheduling problems, but also to applications of similar flavour, e. g., to crew and airline scheduling. For a survey on set partitioning approaches to such problems, we refer the reader to Desrosiers, Dumas, Solomon, and Soumis [1995], Barnhart, Hane, and Vance [1996], Barnhart, Johnson, Nemhauser, and Vance [1997], and Soumis [1997].

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. Techniques of this kind have been discussed in various articles. For instance, Carpaneto, Dell'Amico, Fischetti, and Toth [1989] describe a certain integer LP (ILP) formulation based on an assignment formulation 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 and Soumis [1994] show that this additive lower bounding is a special case of Lagrangean relaxation and its corresponding subgradient method. Forbes, Holt, and 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.

We investigate in this paper the solution of the LP relaxation of the multicommodity flow formulation by means of column generation techniques. The standard column generation approach in the literature is based on generating and eliminating columns based on the reduced cost criterion. We propose here a new technique that is based on Lagrangean relaxations of the multicommodity flow model. The method, which we call *Lagrangean pricing*, activates the arcs of complete paths and not only individual arcs. In particular, it is not only possible, but *essential* that columns with positive reduced costs are generated. Lagrangean pricing has been developed independently at the same time by Fischetti and Toth [1996] and Fischetti and Vigo [1996] for solving the Asymmetric Traveling Salesman Problem and the Resource-Constrained Arborescence Problem, respectively.

Solving an MDVSP instance to optimality using LP based approaches requires to solve the LP relaxation to optimality. With Lagrangean pricing, it becomes possible to solve the huge linear programs that come up here. Therefore, we propose Lagrangean pricing as one of the basic ingredients of an effective method to solve this kind of problems to *proven* optimality, see also Löbel [1997] for a comprehensive discussion of this method.

Our computational investigations were performed on large-scale data from the German public transportation companies Berliner Verkehrsbetriebe (BVG), Hamburger Hochbahn AG, and Verkehrsbetriebe Hamburg-Holstein AG. These instances involve problems with up to 49 depots, about 25 thousand timetabled trips, and about 70 million unloaded trips. The results show that our method is capable of solving problems of this size – orders of magnitude larger than the instances successfully solved with other approaches, as far as we know.

In the recent years, considerable research has gone into the design of pseudo-polynomial time approximation algorithms for multicommodity flow *feasibility* problems, e. g., see Leighton, Makedon, Plotkin, Stein, Tardos, and Tragoudas [1991], Plotkin, Shmoys, and Tardos [1991], and Klein, Plotkin, Stein, and Tardos [1994]. We have investigated these approaches and did not see how they could substantially help solving the *optimization* problems that we investigate here. In particular, the results reported in Leong, Shor, and Stein [1993] and Borger, Kang, and Klein [1993] on rather small problem instances do not look encouraging from a computational point of view.

In the following, we assume the reader to be familiar with integer linear programming and network flows, e. g., see Schrijver [1989] and Ahuja, Magnanti, and Orlin [1993].

2 The Multiple-Depot Vehicle Scheduling Problem

The following section refers to some basic terminology for MDVSPs that we quickly resume here. For more details see Löbel [1997] or Grötschel, Löbel, and Völker [1997].

The fleet of a public 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 $\mathcal{T}^- := \{t^- |\ t \in \mathcal{T}\}$, $\mathcal{T}^+ := \{t^+ |\ t \in \mathcal{T}\}$, and $\mathcal{T}_d := \{t \in \mathcal{T} |\ d \in G(t)\}$.

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 all these trips **unloaded trips**. Let $\Delta_{t,t'}$ denote the duration of the dead-head trip from t^+ to t'^- including some layover time. Whenever $e_t + \Delta_{t,t'} \leqslant s_{t'}$, we call the corresponding dead-head trip **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 trips and unloaded trips occur alternately. A vehicle schedule is called **valid** if all its trips belong to the same depot.

We denote the disjoint union by $\dot\cup$. We introduce the following sets of trips for each depot $d \in \mathcal{D}$: $A_d^{\text{t-trip}} := \{(t^-, t^+) |\ t \in \mathcal{T}_d\}$ and

$$A_d^{\text{u-trip}} := \{(d^+, t^-), (t^+, d^-) |\ t \in \mathcal{T}_d\} \dot\cup \bigdot\bigcup_{\substack{\text{compatible dead-head}\\ \text{trips of depot } d\\ \text{connecting } p \text{ with } q}} \{(p^+, q^-)\}.$$

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. The minimization of this "two-stage" objective function first minimizes the fleet size and, subordinate, the operational costs among all fleet minimal 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 are

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 gives an illustration of D for 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 $\mathcal{T}_\text{g} = \{\text{a,b,c,e}\}$.

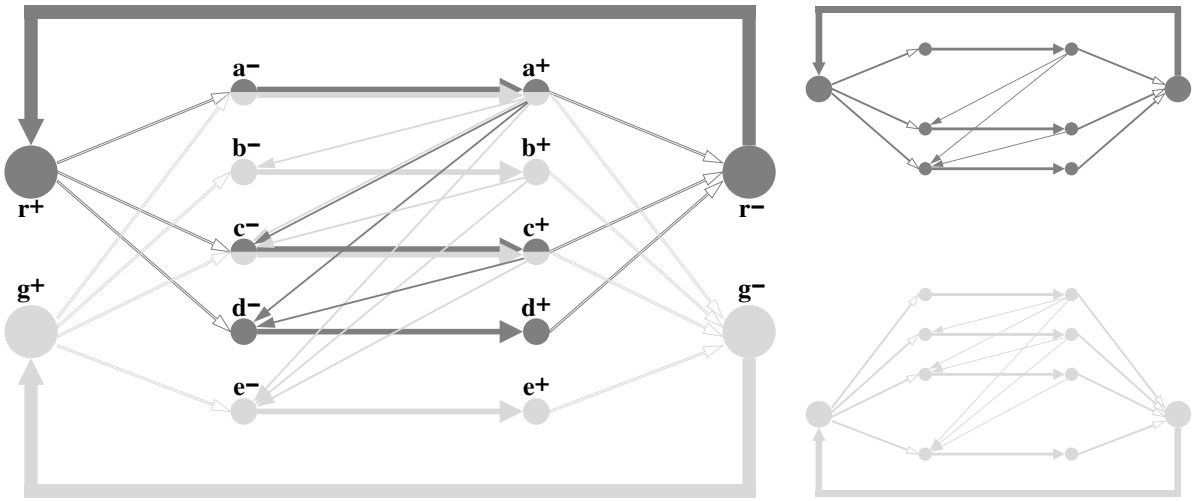

Figure 1: Digraphs (V, A), (V_r, A_r), and (V_g, A_g).

We introduce an integer variable x_a^d for each arc $a \in A_d$ and each depot $d \in \mathcal{D}$. x_a^d denotes a decision variable indicating whether a vehicle of depot d runs trip a or not, unless 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 these are combined into a vector $x := (x^d)_{d \in \mathcal{D}} \in \mathbb{R}^A$. Given a node $v \in V$, let $\delta^+(v)$ denote all arcs of A with tail v and, accordingly, $\delta^-(v)$ denote all arcs with head 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) integer linear programming (ILP) formulation of the MDVSP reads:

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

subject to

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

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 $v \in V$ also leaves it.

The ILP (1) includes many redundant constraints that can be eliminated by performing some preprocessing steps as shown in Löbel [1997]. The main idea is to shrink both nodes t^- and t^+ to one node t, for all $t \in \mathcal{T}$, and to shrink both nodes d^- and d^+ to one node d, for all $d \in \mathcal{D}$. This corresponds to an elimination of each arc not belonging to some unloaded tripand 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

$$x\big(\delta^+(t^+)\big) = 1, \qquad \forall\, t \in \mathcal{T}, \tag{2b}$$

$$-x\big(\delta^-(t^-)\big) = -1, \qquad \forall\, t \in \mathcal{T}, \tag{2c}$$

$$x^d\big(\delta^+(t^+)\big) - x^d\big(\delta^-(t^-)\big) = 0, \qquad \forall\, t \in \mathcal{T}_d \;\; \forall\, d \in \mathcal{D}, \tag{2d}$$

$$x^d\big(\delta^+(d^+)\big) \leqslant \kappa_d, \qquad \forall\, d \in \mathcal{D}, \tag{2e}$$

$$x_a^d \geqslant 0, \qquad \forall\, a \in A_d^{\text{u-trip}} \;\; \forall\, d \in \mathcal{D}, \tag{2f}$$

$$x \text{ integral.} \tag{2g}$$

Note that equations (2c) are a linear combination of (2b) and (2d), but helpful for one of the following relaxations.

Relaxations.

The natural relaxation of (2) is of course the LP relaxation. We also give two well known Lagrangean relaxations, which are the basis of our new Lagrangean pricing. For notational simplification, we use the same symbols for the dual variables of the LP relaxation and for the multipliers of the Lagrangean relaxations.
The LP relaxation of (2) is simply

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

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), (2d), and (2e), respectively.

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 (2d) and the depot capacities (2e). Relaxing (2d) and (2e), we obtain a Lagrangean dual LR_{fcs} with respect to the flow conservations reading $\max_{\pi;\gamma \geqslant 0} L_{\text{fcs}}(\pi, \gamma)$ with inner minimization problem

$$L_{\text{fcs}}(\pi,\gamma) := \min_{\substack{x \text{ satisfying} \\ (2\,\text{b}),\,(2\,\text{c}),\,(2\,\text{f}), \\ \text{and } (2\,\text{g})}} \sum_{d \in \mathcal{D}} \Bigg(\sum_{a \in A_d^{\text{u-trip}}} c_a^d \, x_a^d \; - \sum_{t \in \mathcal{T}_d} \pi_t^d \Big(x^d\big(\delta^+(t^+)\big) - x^d\big(\delta^-(t^-)\big) \Big) \\ - \; \gamma^d \Big(\kappa_d - x^d\big(\delta^+(d^+)\big) \Big) \Bigg) \tag{4}$$

The subscript "fcs" of L_{fcs} and LR_{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 (1b). Relaxing (1b), we obtain a Lagrangean dual $\mathrm{LR}_{\mathrm{fcd}}$ with respect to the flow conditions reading $\max_\nu L_{\mathrm{fcd}}(\nu)$ with inner minimization problem

$$L_{\mathrm{fcd}}(\nu) := \min_{\substack{x \text{ satisfying} \\ (1\,\mathrm{c})\text{–}(1\,\mathrm{g})}} \Bigg(\sum_{d\in\mathcal{D}} \sum_{a\in A_d^{\text{u-trip}}} c_a^d\, x_a^d \;-\; \sum_{t\in\mathcal{T}} \nu_t \Big(\sum_{d\in G(t)} x^d_{(t^-,t^+)} \;-\; 1 \Big) \Bigg) \tag{5}$$

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

3 Implementation Details

The instances of the MDVSP we encountered in practice have up to 70 million variables and 125 thousand equations. Ignoring the integrality stipulation, we obtain linear programs, which are way out of reach for even the best LP codes currently available.

We will show in this section how the LP relaxation (3) can be solved to optimality using Lagrangean pricing techniques. In particular, our implementation combines robust LP software, a minimum-cost flow code, and parts of Lagrangean relaxations codes for the MDVSP. In our case, we use the CPLEX Callable Library (CPLEX [1997]), the network simplex code MCF (Löbel [1997a]), and parts of the Lagrangean relaxation code presented in Kokott and Löbel [1996].

In a first try, we have tried to apply a standard column generation and elimination technique based on the reduced cost criterion, see Sect. 3.1. With such a standard approach, however, only rather small instances have been solved successfully. *Stalling* in the objective value occurred for larger instances. Within the column generation process, many new columns have been generated, but none of them could help to improve the objective value. Moreover, almost all active columns have reduced costs near to zero and, therefore, none of them could be eliminated resulting in too large RLPs.

The new Lagrangean pricing techniques can help to improve the column generation process. We will describe Lagrangean pricing in Sect. 3.2. The right composition of all employed ingredients is given in Sect. 3.3.

3.1 Column Generation

The basic idea of a column generation is to provide only a relatively small subset of the columns, which includes some optimal basis, and to ignore all the other ones. One starts with a subset of columns that, in addition, includes at least some primal feasible basis. The reduced LP, defined by this subset of columns, is called *restricted* LP (RLP). It is the task to solve a sequence of RLPs until it is proved that the last RLP contains the columns of some basis, which is optimal for the complete LP. The global optimality condition of an RLP is described below.

An exact description of the column generation is as follows. Assume that we have already determined a subset $\tilde{A} \subset A$ such that $\tilde{A}$ includes some (primal) feasible solution. Consider the RLP including only the columns according to this subset $\tilde{A}$. In addition, assume that a primal feasible starting basis is determined. In general, the RLP is resolved to optimality, but it is sufficient to perform only some (primal) simplex iterations. Let $\tilde{\nu}$, $\tilde{\pi}$, and $\tilde{\gamma}$ denote the value of the dual multipliers associated with the last basis of the current RLP. For notational simplicity,

let $\tilde{\nu}_d := 0$ and $\tilde{\pi}^d_d := 0$, for all $d \in \mathcal{D}$, denote *artificial* variables. We compute for each variable the reduced costs

$$\overline{c}^d_{ij} := c^d_{ij} - \tilde{\nu}_i - \tilde{\pi}^d_i + \tilde{\pi}^d_j + \left\{ \begin{array}{c} 0 \\ \tilde{\gamma}_d \end{array} \right\} \quad \forall\ (i,j) \in A \text{ such that } i \left\{ \begin{array}{c} \notin \\ \in \end{array} \right\} \mathcal{D}. \tag{6}$$

Note that $\overline{c}_a \geqslant 0$ for all active columns $a \in \tilde{A}$ if the last RLP was solved to optimality. If $\overline{c}_a \geqslant 0$ for all $a \in A$, the global optimality of the current basis is proved and we can stop. Otherwise, we search for some (inactive) variables $a \in A \setminus \tilde{A}$ and generate their corresponding columns.

Standard column generation schemes generate only those columns that violate the reduced cost criterion $\overline{c} \geqslant 0$, i.e., variables with negative reduced costs. But, as we will see below, it turned out that adding also columns with nonnegative reduced costs may be advantageous. Having selected the variables that become active, $\tilde{A}$ and the corresponding RLP are redefined appropriately. The enlarged LP is reoptimized or a limited number of simplex iterations is performed, and we iterate until we prove optimality, i.e., $\overline{c}_a \geqslant 0$.

Obviously, to achieve any progress, at least one variable having negative reduced cost must be activated between two consecutive RLPs. Tests in practice have shown that it is impossible to generate all inactive columns with negative reduced costs since the next RLP gets far too large and cannot be handled at all. Therefore, we restrict the number of new arcs to some parameter controlled limit. This limit ranges from 200 to 3000 variables for each depot, depending on the problem size.

For the standard column generation scheme, we use Dantzig's pricing rule. We select the variables with most invalid reduced costs as candidates to become activated. With this approach, it is also possible to prove the global optimality of some RLP, provided that the last RLP has been solved to optimality and includes some optimal basis. We have also tried to use more advanced pricing rules such as Devex or steepest-edge pricing. Similar to Dantzig's rule, these rules generate only columns with negative reduced costs, but we could not observe better computational results. Therefore, we have rejected those advanced pricing rules and apply only Dantzig's rule. Lagrangean pricing.

To avoid that the RLPs become too large, we must also remove obsolete columns in each iteration of the column generation process. All columns whose reduced costs exceed some predefined parameter controlled positive threshold are therefore eliminated.

3.2 Lagrangean pricing

In a first version, we have tried to solve large MDVSP instances using only standard column generation and elimination schemes. But this approach failed. One main obstacle is the completely degenerate LP relaxation. A second reason for the difficulties is as follows: The standard column generation scheme activates only variables with negative reduced cost. These variables can locally promise some progress in the objective value, but it is not clear whether they may have any influence on the solution and the objective value without an interaction with some other related nonactive variables. Therefore, we came up with the idea that the nonactive variables should be not only evaluated alone by its reduced costs, but also in interaction with all the other active and inactive variables. However, how can this be done efficiently? We have to find a method that determines good (nonactive) variables that may give progress in the objective value as best as possible. To use the information already compiled within the previous RLPs, this method should also use dual information as pricing methods do. It may also be a good idea

to invoke also Lagrangean relaxation techniques that turned out to give good approximations of our hard solvable LP relaxations.

The answer to all these questions is *Lagrangean pricing*: The inner minimization problems L_{fcs} (4) and L_{fcd} (5) of the presented Lagrangean relaxations LR_{fcs} and LR_{fcd} can be solved efficiently – even for the complete variable set – and give excellent approximations of the LP relaxation. So, we evaluate for LR_{fcs} and LR_{fcd} the linear programs $L_{\text{fcs}}(\tilde{\pi}, \tilde{\gamma})$ and $L_{\text{fcd}}(\tilde{\nu})$. Remember, $\tilde{\nu}$, $\tilde{\pi}$, and $\tilde{\gamma}$ denote the value of the dual multipliers associated with the flow conditions (2b), the flow conservations (2d), and the depot capacities (2e) of the last basis of the current RLP.

Obviously, both relaxations approximate the LP relaxation with all active and inactive variables, use dual information given by the last RLP, are based on good relaxations of the LP relaxation, and can be evaluated efficiently. We still have to show how good nonactive variables can be determined. The solution of each inner minimization problem can be interpreted as a set of vehicle schedules that seem to be advantageous for the given shadow prices of the current RLP relaxation. In the case of the Lagrangean relaxation L_{fcd}, these vehicle schedules may include unloaded trips of different depots. Consider all the vehicle schedules defined by the optimal solutions attaining the values of $L_{\text{fcs}}(\tilde{\pi}, \tilde{\gamma})$ and $L_{\text{fcd}}(\tilde{\nu})$. Each still nonactive variable according to some unloaded trip of some of these vehicle schedules determines a candidate to become active.

3.3 The Basic Ingredients

We have made many computational experiments to find out the right mixture of the techniques presented above. The basic ingredients, each being indispensable to solve large-scale instances at all, are as follows:

Initial RLP relaxation: The initial RLP should contain at least some primal feasible solution yielding a value as close to the LP optimum as possible. A very efficient way to heuristically determine some solution is a *schedule – cluster – reschedule heuristic* (SCR). Heuristics of this kind are described, e. g., in Grötschel, Löbel, and Völker [1997], Dell'Amico, Fischetti, and Toth [1993], and Daduna and Mojsilovic [1988]. A faster method is a *nearest depot heuristic* (ND), which assigns each timetabled trip to some depot with the smallest sum of the pull-out and pull-in costs. This kind of opening heuristic, however, yields rather poor starting points, see Löbel [1997].

As soon as each timetabled trip is assigned to some depot, the problem decomposes into $|\mathcal{D}|$ independently solvable single-depot subproblems. We solve for each depot its single-depot instances according to all its heuristically assigned timetabled trips. Each unloaded trip that corresponds to some basic variable becomes active and its column is generated for the initial RLP. Thus, the first RLP includes at least the feasible solution defined by the union of the solutions of all subproblems together. A further idea is to use the union of all columns generated by any primal (opening heuristic) and dual (Lagrangean relaxation) method. Unfortunately, we have not tested such a combination of different heuristics.

The Workhorses: Minimum-Cost Flow and LP: Solving the LP relaxation with our approach exactly, requires at several steps the efficient solution of minimum-cost flow problems and linear programs: The minimum-cost flow problems stem from the Lagrangean relaxations, the LPs are RLPs. All minimum-cost flow problems have been solved with MCF. This is an

implementation in C of the primal and the dual network simplex algorithm and is available for academic use free of charge via WWW at URL http://www.zib.de/Optimization, see Löbel [1997a]. The linear programs have been solved with the primal as well as the dual simplex solver of the CPLEX Callable Library, version 4.0.9. CPLEX turned out to be a reliable and robust method for our degenerate (R)LP problems.

For our computations, an important feature of CPLEX 4.0 is the new and more gentle perturbation method. In previous version of CPLEX, the bounds of all variables have been relaxed when perturbing a problem. This perturbation approach led often to numerical problems when we have solved our test instances. With the current version of CPLEX, only all basic variables are perturbed whenever the perturbation starts. As soon as some nonbasic variable has been selected to become basic it will also be perturbed if not already done in some previous iteration. This simple alteration of the perturbation strategy has significantly improved the efficiency of our implementation for large MDVSPs.

The column generation is divided into two phases: First, a *Lagrangean phase* where we apply standard and Lagrangean pricing, and, second, a *standard phase* in which we apply only the standard column generation approach.

Lagrangean phase: This phase precedes always the standard phase and is applied as long as the objective value declines between two consecutive RLPs at least by some predefined parameter controlled threshold (10.0 is used as default). The last basis of the last RLP is always neglected, and each RLP is reduced by LP preprocessing. The columns of each RLP obtained in this phase are, at least for large MDVSPs, far too many for the primal simplex solver. We use here the dual simplex solver. We have also tried to use CPLEX's primal-dual logarithmic barrier solver. It turned out, however, that numerical problems often prevent the barrier solver from proceeding.

As long as there is a sufficiently large gap between the optimal LP value and the value of the current RLP, the Lagrangean phase works well. However, stalling occurs when the current RLP value approaches the LP optimum. This phase is unable to converge to an optimal variable set: Although the objective has been become almost optimal, the standard column generation between two consecutive RLPs finds always thousands up to millions of unloaded trips that do not satisfy the reduced cost criterion. This effect is maybe a result of neglecting always the last basis (i. e., all dual information) of the previous RLP, but we cannot provide any other reasonable explanation.

Thus, we came up with the idea to use at this point only the standard column generation scheme: We switch to the standard phase when the objective progress becomes too small and, therefore, some "approximation of optimality" has been reached.

Standard phase: When we start this phase, we believe that our current RLP contains some almost optimal basis of the complete LP relaxation. The occurring RLPs are now solved with the primal simplex solver and each RLP starts with the last basis of the preceding RLP. This approach iterates until the (global) optimality of some RLP can be proved with the reduced cost criterion.

4 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 that are displayed in Tab. 1. The term $\varnothing G := \sum_{t \in \mathcal{T}} G(t)/|\mathcal{T}|$ 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.

Test Sets[a]	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert A\rvert/1,000$	$\varnothing\lvert G(t)\rvert$	number of equations
Berlin 1	44	24,906	69,700	4.03	125,255
Berlin 2	49	24,906	13,200	1.56	63,641
Berlin 3	3	1,313	2,300	2.33	4,370
Berlin-Spandau 1	9	2,424	3,700	4.94	14,418
Berlin-Spandau 2	9	3,308	8,800	5.49	21,470
Berlin-Spandau 3	13	2,424	590	1.92	7,103
Berlin-Spandau 4	13	3,308	1,530	2.25	10,753
Berlin-Spandau 5	13	3,331	1,550	2.25	10,834
Berlin-Spandau 6	13	1,998	380	1.90	5,798
Berlin-Spandau 7	7	2,424	3,300	4.16	12,506
Berlin-Spandau 8	7	3,308	7,800	5.02	18,376
Hamburg 1	12	8,563	10,900	2.23	27,696
Hamburg 2	9	1,834	1,000	2.02	5,549
Hamburg 3	2	791	200	1.32	1,835
Hamburg 4	2	238	23	1.04	487
Hamburg 5	2	1,461	580	1.31	3,379
Hamburg 6	2	2,283	1,600	1.33	5,323
Hamburg 7	2	341	34	1.32	795
Hamburg-Holstein 1	4	3,413	4,000	1.68	9,167
Hamburg-Holstein 2	19	5,447	9,400	3.65	25,334

[a]Compared to the problems presented in Grötschel, Löbel, and Völker [1997], the number of timetabled and unloaded trips and the weights for some unloaded trips have been changed for some instances due to slightly different rules for the depot generation and compatibility of dead-head trips.

Table 1: Real-world test sets.

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

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

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

5 Computational Results

All presented 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 during all our test runs.

In Tab. 2 we summarize the objective values (fleet size and operational weight) of the lower bounds obtained with $L_{\text{fcs}}(0,0)$ and the LP relaxation, the integer optimum (or best known integer value), and the upper bounds obtained with the SCR and ND heuristics (taken from Löbel [1997]). Note that evaluating L_{fcs} at zero is equivalent to neglect all flow conservation constraints (2d) and all depot capacities (2e).

Test Sets	$L_{\text{fcs}}(0,0)$		LP relax.		Optimum or best int. solution		Heuristics			
							SCR		ND	
	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight	Fleet	Weight
Berlin 1	1323	715714	1323	759162.0[1]	1329	850680	1347	1317379	1575	3279774
Berlin 2	1350	715623	1353.7	797918.8	**1354**	777823	1366	1318085	1655	3900191
Berlin 3	**69**	14043	**69**	**14119**	**69**	**14119**	**69**	14122	70	14366
B-Spandau 1	**125**	65585	**125**	65610.5	**125**	**65611**	**125**	125786	139	70092
B-Spandau 2	184	78947	184.5	79110.0	**185**	79052	**185**	289262	207	213333
B-Spandau 3	**127**	90514	**127**	**93745**	**127**	**93745**	**127**	152109	135	149629
B-Spandau 4	**191**	195844	**191**	**230846**	**191**	**230846**	192	395891	222	475552
B-Spandau 5	**191**	191141	**191**	**227580**	**191**	**227580**	194	393922	220	487944
B-Spandau 6	**98**	91109	**98**	**101075**	**98**	**101075**	**98**	132650	109	139593
B-Spandau 7	**125**	65585	**125**	65610.5	**125**	**65611**	**125**	105853	139	70092
B-Spandau 8	**184**	78947	184.5	79110.0	**185**	79093	**185**	259406	207	213333
Hamburg 1	**432**	66874	**432**	71068.3	**432**	**71069**	446	70291	489	76054
Hamburg 2	**103**	15356	**103**	**16070**	**103**	**16070**	104	16792	114	15849
Hamburg 3	**39**	5557	**39**	**5860**	**39**	**5860**	**39**	6298	41	5429
Hamburg 4	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**	**6**	**1358**
Hamburg 5	**62**	12092	**62**	**12502**	**62**	**12502**	**62**	13535	65	12101
Hamburg 6	**111**	15705	**111**	**15791**	**111**	**15791**	**111**	16588	**111**	**15791**
Hamburg 7	**15**	2832	**15**	**2961**	**15**	**2961**	16	2836	16	2836
H-Holstein 1	**201**	28697	**201**	**29027**	**201**	**29027**	**201**	30497	213	32132
H-Holstein 2	360	51084	**362**	52787.7	**362**	**52788**	363	72700	393	62598

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

The quality of the lower bounds obtained by $L_{\text{fcs}}(0,0)$ and the LP relaxation are quite good. First, let us consider the fleet size values: On the average, $L_{\text{fcs}}(0,0)$ approximates the integer

optimal fleet sizes (or best known integer upper bounds) by a factor of 0.9988 with a standard deviation of 0.0021. The LP relaxation gives for the fleet size an approximation of 0.9995 with a standard deviation of 0.0012. If we ignore the test set Berlin 1, which is currently the only instance that we could not solve optimally, and round up each fractional fleet size value, the LP relaxation approximates the optimal integer fleet size exactly for all the other test instances. Let us now consider the operational weights. Obviously, we can estimate the quality of a given lower bound for the operational weight only if the lower bound for the fleet size is exact. For all those test instances, $L_{\text{fcs}}(0,0)$ approximates, on the average, the optimal operational weights by a factor of 0.9534 with a standard deviation of 0.0505. The LP relaxation produces always the optimal integer value whenever the lower bound for the fleet size was tight. The opening heuristics SCR and ND approximate, on the average, the integer optimal fleet size (or best known integer lower bound) by a factor of 1.0080 and 1.0962 with a standard deviation of 0.0158 and 0.0580, respectively. These results support the following conclusions:

- The LP relaxations yield quite tight lower bounds. It is often the case that rounding up the LP value to the next integer value yields already the integer optimum. A similar phenomenon is observed by Forbes, Holt, and Watts [1994]: 22 of their 30 test instances 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 values obtained by $L_{\text{fcs}}(0,0)$ are close to the optimal LP values. Let ν^+ and ν^- denote the dual variables associated with the flow conditions (2b) and (2c) in $L_{\text{fcs}}(0,0)$. We have shown in Löbel [1997] that $L_{\text{fcd}}(\nu^+ - \nu^-)$ and $L_{\text{fcs}}(0,0)$ yield the same optimal value. Moreover, these lower bounds can be improved if we apply a subgradient method, see Kokott and Löbel [1996]. Thus, LR_{fcs} and LR_{fcd} give an excellent approximation of the LP relaxation.

- The SCR heuristic produces (significantly) better solutions than the ND heuristic. From this point of view, SCR would be the better opening heuristic. But surprisingly, we can observe from Tab. 3 that starting with the columns provided by ND needs less running time for about 60 % of our test instances. So, it is an open question whether it might be advantageous to initialize the column generation with the columns provided by both heuristics.

If we are only interested in obtaining tight lower bounds quickly, we would only evaluate $L_{\text{fcs}}(0,0)$. Simply neglecting all flow conservation constraints provides, for our largest instance Berlin 1, a lower bound within 15 minutes running time. This lower bound provides the same fleet size lower bound as the LP relaxation does, and the operational weight gap between $L_{\text{fcs}}(0,0)$ and the best known LP value is only about 6 %. Our LP method needs about 200 hours cpu time to find a fleet minimal value for Berlin 1, see Tab. 3.

In the following, we report on some specific observations we made in solving the LP relaxation. Table 4 shows, for each test instance and for each opening heuristic, the number of RLPs that have been solved until optimality has been proved, the total number of CPLEX iterations that have been performed, and the number of columns that have been generated and eliminated within the column generation process. We shall describe some features of our column generation

Test Sets	$L_{\text{fcs}}(0)$	LP relax.			
		SCR		ND	
		Init.	Total	Init.	Total
Berlin 1	916	12386	—	171	—
Berlin 2	229	3810	34795	79	32767
Berlin 3	17	25	431	7	311
B-Spandau 1	27	343	43777	15	66501
B-Spandau 2	93	1939	112337	39	165048
B-Spandau 3	9	42	975	7	739
B-Spandau 4	25	334	6014	14	4384
B-Spandau 5	31	354	5264	15	5618
B-Spandau 6	17	7	162	6	227
B-Spandau 7	23	259	24717	14	46597
B-Spandau 8	67	1238	82284	36	62041
Hamburg 1	185	2868	—[a]	29	50246
Hamburg 2	12	103	875	7	685
Hamburg 3	4	16	35	2	31
Hamburg 4	2	1	3	1	3
Hamburg 5	10	86	258	5	155
Hamburg 6	18	30	148	11	84
Hamburg 7	2	2	8	1	9
H-Holstein 1	40	199	2619	18	2087
H-Holstein 2	101	1696	46673	40	64489

[a] After 32 hours, we have stopped the run since stalling of the objective progress occurs; the last objective value in the standard phase was 432.25 vehicles with a weight of 71169.125. The Lagrangean phase alone takes more than 63000 seconds, which is more than the total running time of ND!

Table 3: Running times in seconds on a SUN Model 170 UltraSPARC with 512 MByte main memory.

method with the example of Hamburg 1 starting with ND. The behaviour of our implementation starting with SF-CS or ND is similar for all the other problems.

We have observed the following: The dominating part of the objective value, the fleet sizes, converge quickly to the minimum value. At the *crossover* from the standard to the Lagrangean phase, the objective values are almost optimal. For our test set, neither the standard nor the Lagrangean phase dominates the total running time. Figure 2 shows a typical development of the fleet size values (left picture) and their operational weights (right picture) in respect to the running time.

The number of generated and eliminated columns are almost always about the same for each iteration of the column generation process, see the left picture of Fig. 3. Therefore, the LP sizes are relatively constant during the solution process. The right picture in this figure shows a typical development of the number of rows, columns, and nonzero elements of the k^{th} RLP and of

Table 4: Running statistics for our column generation and elimination.

Test Sets	Starting with SCR								Starting with ND							
	RLPs			CPLEX			Col. gen.		RLP's			CPLEX			Col. gen.	
	Number of			Iterations / 1000			Cols. / 1000		Number of			Iterations / 1000			Cols. / 1000	
		Lgr.	Stn.		Lgr.	Stn.	gen.	elim.		Lgr.	Stn.		Lgr.	Stn.	gen.	elim.
Berlin 1	—	24	—	—	6.333	—	(967)	(830)	—	—	—	—	—	—	—	—
Berlin 2	97	25	72	501	307	194	691	634	106	27	79	561	371	190	716	658
Berlin 3	24	2	22	24	1	23	41	36	23	5	18	18	6	12	55	50
B-Spandau 1	23	12	11	881	615	266	250	227	30	16	14	1.399	1.091	308	293	274
B-Spandau 2	26	15	11	1.260	628	632	458	430	28	14	14	1.762	890	872	453	412
B-Spandau 3	34	13	21	70	57	13	211	202	30	11	19	61	49	12	186	177
B-Spandau 4	40	17	23	230	188	42	348	335	46	17	29	178	130	48	336	324
B-Spandau 5	47	14	33	191	135	56	323	310	49	17	32	223	161	62	349	337
B-Spandau 6	20	4	16	13	5	8	70	63	25	7	18	21	15	6	99	92
B-Spandau 7	34	12	22	594	403	191	207	190	30	16	14	1.069	869	200	247	227
B-Spandau 8	28	13	15	1.073	512	561	378	350	26	15	11	966	575	391	279	248
Hamburg 1	—	38	—	—	1.167	—	(296)	(273)	98	24	74	873	656	217	478	451
Hamburg 2	26	9	17	38	21	17	128	122	20	8	12	35	21	14	113	107
Hamburg 3	13	2	11	1	1	0	19	17	23	2	21	1	0	1	19	17
Hamburg 4	3	2	1	0	0	0	1	0	3	2	1	0	0	0	1	0
Hamburg 5	25	6	19	7	4	3	71	67	44	2	42	11	1	10	126	212
Hamburg 6	12	2	10	4	1	3	37	32	2	2	0	1	1	0	12	9
Hamburg 7	13	3	10	1	0	1	9	8	21	3	18	1	0	1	13	12
H-Holstein	20	13	7	97	79	18	190	177	16	8	8	79	63	16	125	111

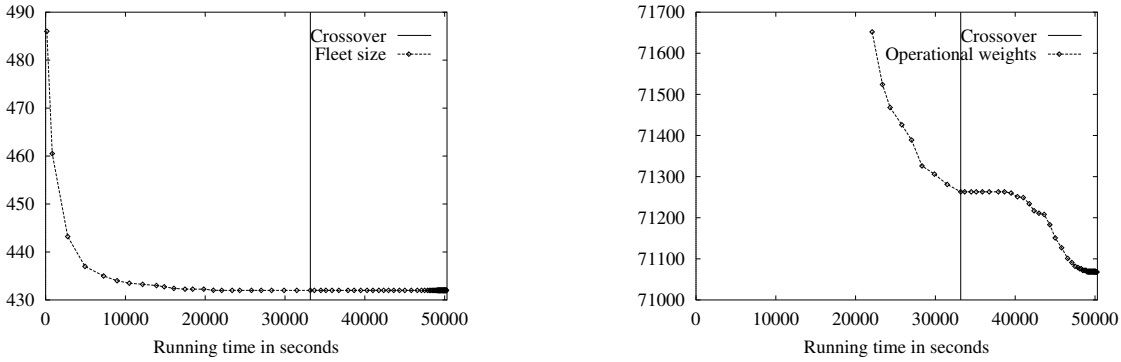

Figure 2: Typical development of the fleet size values (left picture) and their operational weights (right picture) in respect to the running time.

the k^{th} RLP in the Lagrangean phase after LP preprocessing. We can observe from this picture the importance of LP preprocessing within the Lagrangean phase: Without this preprocessing, neither the primal nor the dual simplex solver would not be applicable here because of intolerable long running times. But preprocessing reduces the sizes of these RLPs significantly such that it becomes possible to solve the occurring RLPs within acceptable running times.

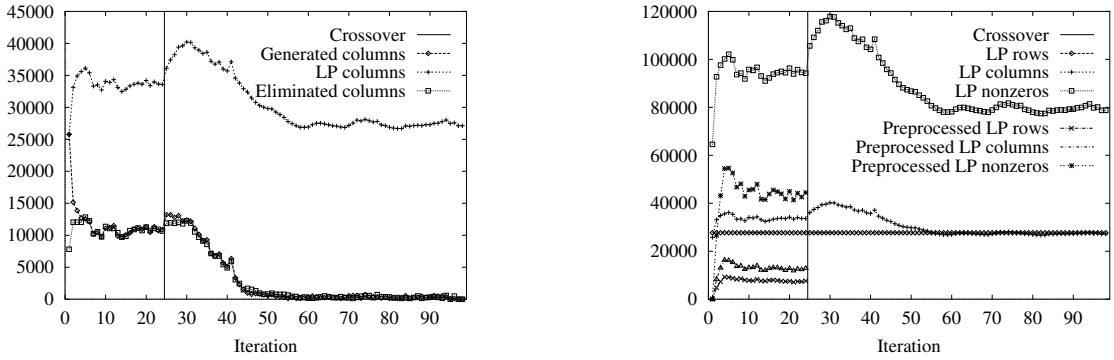

Figure 3: Typical number of generated, eliminated, and active LP columns (left picture), number of LP columns, rows, and nonzero elements per iteration (right picture).

6 Conclusions

We have presented a column generation method to solve the LP relaxation of the multicommodity flow formulation of the MDVSP. The key ingredients here are Lagrangean pricing and the right combination of available LP and minimum-cost flow codes as, e. g., CPLEX and MCF. With this new technique, it becomes possible, for the first time, to solve not only the LP relaxations, but the ILP formulation of MDVSP instances with up to 8,563 timetabled trips to proven optimality and instances with up to 24,906 timetabled trips almost optimal with a gap of less than 0.5 %. Therefore, we propose Lagrangean pricing as one of the basic ingredients of an effective method to solve multiple-depot vehicle scheduling problems.

From our computational results on Lagrangean pricing for large-scale real-world instances, we draw the following conclusions: First, our method can solve degenerate LP relaxations of

large-scale problems from practice to optimality or, at least, with an acceptable small gap. Fleet minimal LP solutions that do not necessarily yield minimum operational weights can be generated quite fast. Without Lagrangean pricing, it is not possible to solve even smaller test instances. Second, if the column generation process is combined with some heuristic(s) exploiting the solutions of the solved restricted LPs, it is possible to compute a feasible solution yielding the minimum fleet size and a relatively small gap for the operational costs for almost all of our test problems within few iterations and few hours of running time, see Löbel [1997]. Third, LP methods are not the right tools to produce lower bounds quickly for this kind of problems; Lagrangean relaxations and subgradient methods, as presented for instance in Kokott and Löbel [1996], are the right methods.

7 Acknowledgements

We are grateful to Uwe Strubbe of IVU GmbH and Manfred Völker and Anna Neufeld of HanseCom GmbH for providing us with real-world problems of the German transportation companies Berliner Verkehrsbetriebe, Hamburger Hochbahn AG, and Verkehrsbetriebe Hamburg-Holstein AG. We are grateful to these three transportation companies for their kind permission to use and publish their data. We are grateful to Martin Grötschel, our colleagues Ralf Borndörfer and Alexander Martin (both at ZIB), and Robert Bixby for their helpful discussions. We are also grateful to Robert 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

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Barnhart, C., Hane, C. A., and Vance, P. H. (1996). Integer multicommodity flow problems. In Cunningham, W. H., McCormick, S. T., and Queyranne, M., editors. *Integer Programming and Combinatorial Optimization*, Proc. of the 5th Int. IPCO Conf., Vancouver, British Columbia, Canada.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., and Vance, P. H. (1997). Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200.

Bertossi, A. A., Carraresi, P., and Gallo, G. (1987). On some matching problems arising in vehicle scheduling models. *Networks*, 17:271–181.

Borger, J. M., Kang, T. S., and Klein, P. N. (1993). Approximating concurrent flow with unit demands and capacities: an implementation. In Johnson and McGeoch [1993].

Carpaneto, G., Dell'Amico, M., Fischetti, M., and 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 available at URL: http://www.cplex.com.

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

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

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

Dell'Amico, M., Fischetti, M., and 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., and Soumis, F. (1995). *Time Constrained Routing and Scheduling.* Chapter 2 in Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L., editors. *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science.* Elsevier Science B.V., Amsterdam.

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

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

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

Grötschel, M., Löbel, A., and Völker, M. (1997). Optimierung des Fahrzeugumlaufs im öffentlichen Nahverkehr. In Hoffmann, K.-H., Jäger, W., Lohmann, T., and Schunck, H., editors, *Mathematik – Schlüsseltechnologie für die Zukunft*, pages 609–624. Springer Verlag.

Johnson, D. S. and McGeoch, C. C., editors (1993). *Network Flows and Matching*, volume 12 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society.

Klein, P., Plotkin, S., Stein, C., and Tardos, É. (1994). Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Comput.*, 23(3):446–487.

Kokott, A. and 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 via WWW at URL: http://www.zib.de/ZIBbib/Publications.

Leighton, T., Makedon, F., Plotkin, S., Stein, C., Tardos, É., and Tragoudas, S. (1991). Fast approximation algorithms for multicommodity flow problems. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing.*

Leong, T., Shor, P., and Stein, C. (1993). Implementation of a combinatorial multicommodity flow algorithm. In Johnson and McGeoch [1993].

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

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

Plotkin, S., Shmoys, D. B., and Tardos, É. (1991). Fast approximation algorithms for fractional packing and covering problems. In *32nd Annual Symposium on Foundations of Computer Science*, pages 495–504. IEEE Computer Society Press.

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

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., and Martello, S., editors. *Annotated Bibliographies in Combinatorial Optimization.* John Wiley & Sons Ltd.