Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER MARTIN GRÖTSCHEL ANDREAS LÖBEL

Scheduling Duties by Adaptive Column Generation

This research has been supported with funds of the German Ministry for Research and Education, Grant No. 03-GR7ZI1.
Responsibility for the contents of this article is with the authors.

ZIB-Report 01-02 (January 2001)

Duty Scheduling in Public Transit*

Ralf Borndörfer, Martin Grötschel, and Andreas Löbel

Konrad-Zuse-Zentrum Berlin, Takustr. 7, 14195 Berlin, Germany, www.zib.de

Abstract. This article is about *adaptive column generation techniques* for the solution of duty scheduling problems in public transit. The current optimization status is exploited in an adaptive approach to guide the subroutines for duty generation, LP resolution, and schedule construction toward relevant parts of a large problem. Computational results for three European scenarios are reported.

1 Introduction

Duty scheduling is the activity of operational planning in public transport that deals with the construction of the daily shifts of driving work. Duty scheduling is a central operational issue. It has also economic significance: The average German bus company spends half of its operating budget on driver salaries, see [41,42, both articles in German]. The duty scheduling problem has been studied extensively in the OR literature, see the proceedings of the last four CASPT conferences ([56], [18], [26], and [19]) for an overview.

This article deals with *column generation* methods for duty scheduling. Column generation is one of the best established optimization approaches to duty scheduling. The methodology is based on techniques such as

- duty generation by constrained shortest path computations ([32], [24], [27–29]),
- large scale techniques such as sifting, core, and active set strategies ([12], [16], [15], [51]),
- LP and other acceleration techniques ([33], [23]),
- heuristics ([5], [8], [36], [54], [16], [15]),
- and other contributions (see [7], [30], [52], [21] for surveys).

Likewise, column generation modules form the optimization cores of several commercial duty scheduling systems, among them (in alphabetical order) the GENCOL optimizer of the HASTUS CREW-OPT system (see [25], [31], [22]), the PROB1 solver in the CARMEN system (see [55], [2]), the TURNI system (see [34]), and our optimizer DS-OPT, which is available in the BERTA and MICROBUS systems (see [14, in German] and [40, in German]).

This article discusses the three main algorithmic modules of our column generation system for duty scheduling. These modules use *relaxations* to

* This research has been supported with funds of the German Ministry for Research and Education, Grant No. 03-GR7ZI1. Responsibility for the contents of this article is with the authors.

speed up the identification of "improving" duties, the computation of shadow prices, and the construction of feasible schedules. The techniques are "adaptive" in the sense that they resort to the current optimization status in order to identify relevant parts of a problem. This helps to tackle large instances without ad hoc eliminations of degrees of freedom.

The article is subdivided into five parts. Section 2 lists notation and concepts on column generation and introduces an approximation algorithm that provides the basis for our duty scheduling system. Section 3 studies the constrained shortest path problem that comes up in the pricing subroutine of the algorithm. We propose an integer linear programming formulation for this problem that we solve using Lagrangean relaxation techniques. Section 4 investigates the solution of the master LP using coordinate ascent methods. Section 5 deals with the construction of feasible schedules. We describe a variable fixing heuristic that is based on scoring and probing techniques. Computational experience for three European duty scheduling scenarios is reported in Sect. 6.

2 Duty Scheduling

We list in this section notation, terminology, and concepts on duty scheduling that we will use in this paper. This notation is quite extensive. We have therefore tried to present a structured listing that is easy to reference. We do not discuss application and modeling aspects here; such information can be found in [50], [30], and [14, in German].

2.1 Terminology

The mathematical terminology follows [38]. We use the following extensions:

- We often scale by a quantity $1+\delta$ and denote that by
$$\tilde{x} = (1+\delta)x. \tag{1}$$
- $[\cdot]_a^b : \mathbb{R} \to \mathbb{R},\ x \mapsto \max\ \{a, \min\ \{x, b\}\}$, $[\cdot]^+ = [\cdot]_0^{+\infty}$.
- $\min_i S$ denotes the i-th minimum of a multiset $S = \{s_1, \ldots, s_k\}$, i.e., given some ordering $s_{j_1} \leq s_{j_2} \leq \cdots \leq s_{j_k}$, we define $\min_i S = s_{j_i}$, $i = 1, \ldots, k$, and $\min_i S = \infty$, $i > k$.
- $\operatorname{argmin}_{x \in D} f(x)$ denotes some optimal solution of the optimization problem $\min_{x \in D} f(x)$, ties are broken arbitrarily. argmax is analogous.
- We denote arcs in digraphs by uv and (u, v); the latter notation is used in cases such as $(m, m+1)$. We write $\operatorname{tail}(uv) = u$, $\operatorname{head}(uv) = v$.
- We sometimes interpret matrices as sets of column vectors and use notation such as $(a_{ij}) \cup b$ to add a column b to a matrix (a_{ij}) if an identical column is not already present.

We use Latin symbols in a network context, and Greek ones in a set partitioning context.

2.2 Network Model

The duty scheduling problem can be formulated in terms of a network model. This model involves an acyclic digraph $D = (V, A)$, the *duty scheduling network*. We number the nodes $V = \{0, 1, 2, \dots, m, m+1\}$. Nodes $M = \{1, \dots, m\}$ are the *duty elements* or *tasks*. They correspond to indivisible units of work that arise from cutting vehicle blocks at relief points. Two tasks u and v are joined by an arc or *link* if the same driver can conduct v immediately after u. Nodes 0 and $m+1$ serve as universal *source* and *sink* nodes in D. They represent the start and the end of every duty. Source and sink are linked with *sign-on* and *sign-off arcs* $\delta^+(0) = \{(0,1), (0,2), \dots, (0, m+1)\}$ and $\delta^-(m+1) = \{(0, m+1), (1, m+1), \dots, (m, m+1)\}$, respectively. Note that $\delta^-(0) = \emptyset = \delta^+(m+1)$ because D is acyclic.

Duties correspond to directed $(0, m+1)$-paths in this duty network, but not every path is a duty. Legality and cost of a duty are determined by the rules of so-called *duty types*. Typical duty types are straight, split, and part time duties. A special duty type are the *trippers*, duties that contain exactly one duty element. A precise model of a duty type will be given in Sect. 3.

The *duty scheduling problem* (DSP) is to find a minimum cost set of duties (each duty valid for at least one given duty type) that covers all tasks exactly once.

2.3 Set Partitioning Model

Let $\hat{J} = \{1, \dots, \hat{n}\}$ denote the set of all feasible duties, let $\hat{\xi}_j$, $j \in \hat{J}$, be a 0/1 decision variable for the inclusion/exclusion of duty j, let $\hat{\omega} \in \mathbb{R}^{\hat{n}}_{+}$ be a vector of *nonnegative* duty costs, and let finally $\hat{\Phi} \in \{0,1\}^{m \times \hat{n}}$ be the task-duty incidence matrix. The duty scheduling problem can be formulated as a *set partitioning problem*

$$\text{(SPP)} \quad \min \ \hat{\omega}^{\mathrm{T}}\hat{\xi}, \quad \hat{\Phi}\hat{\xi} = \mathbb{1}, \quad \hat{\xi} \in \{0,1\}^{\hat{n}}. \tag{2}$$

The constraints $\hat{\Phi}\hat{\xi} = \mathbb{1}$ are the *task partitioning constraints*; they stipulate that each task is covered by exactly one duty. We associate with SPP the following LPs:

$$\text{(MLP)} \quad \min \ \hat{\omega}^{\mathrm{T}}\hat{\xi}, \quad \hat{\Phi}\hat{\xi} = \mathbb{1}, \quad \hat{\ell} \le \hat{\xi} \tag{3}$$

$$\text{(RMLP)} \quad \min \ \omega^{\mathrm{T}}\xi, \quad \Phi\xi = \mathbb{1}, \quad \ell \le \xi \tag{4}$$

$$\text{(RDLP)} \quad \max \ \pi^{\mathrm{T}}\eta + \omega^{\mathrm{T}}\ell, \quad \pi^{\mathrm{T}}\Phi \le \omega^{\mathrm{T}}. \tag{5}$$

We further associate with any vector $\pi \in \mathbb{R}^m$ the *pricing problem*

$$\text{(PRICE)} \quad \exists \ j \in \hat{J}: \quad \overline{\hat{\omega}}_j = \hat{\omega}_j - \pi^{\mathrm{T}}\hat{\Phi}_{\cdot j} < 0. \tag{6}$$

MLP is an *LP relaxation* of (2), called the *master LP*. Its special form allows fixings of variables by a 0/1 vector of lower bounds $\hat{\ell} \in \{0,1\}^{\hat{n}}$. A *restricted* *primal master LP* (RMLP) and its dual (RDLP) are obtained from MLP by restriction to some subset of variables $J = \{1, \ldots, n\} \subseteq \{1, \ldots, \hat{n}\}$. We shall use the following notation and conventions:

- $\hat{\cdot}$ is supposed to indicate the "closure" $\hat{J} = \{\widehat{1, \ldots, n}\} = \{1, \ldots, \hat{n}\}$.
- $\hat{\Phi}\hat{\ell} \leq \mathbb{1}$, $\Phi\ell \leq \mathbb{1}$, i.e., we always assume pairwise *compatible* fixings.
- $\hat{\eta} = \mathbb{1} - \hat{\Phi}\hat{\ell}$ and $\eta = \mathbb{1} - \Phi\ell$ denote the incidence vector of the *residuum* of rows not covered by fixed variables.
- An MLP does never contain a zero column $\Phi_{\cdot j} = 0$.
- Any MLP contains a tripper duty of (high) cost $L \in \mathbb{R}_+$ for each task i, i.e.,
$$\forall i \in M: \quad \exists j \in \hat{J}: \quad \hat{\omega}_j = L \quad \text{and} \quad \hat{\Phi}_{\cdot j} = e_i. \tag{7}$$
- $\hat{\nu} = \nu(\hat{\omega}, \hat{\Phi}, \hat{\ell})$ and $\nu(\omega, \Phi, \ell)$ denote the optima of MLP and RMLP (or RDLP), respectively. Note that assumption (7) guarantees finiteness and well-definition.
- $\overline{\hat{\omega}}_j = \hat{\omega}_j - \pi^{\mathrm{T}}\hat{\Phi}_{\cdot j}$ is the *reduced cost* associated with duty j (and π).

2.4 Pseudocode

We list in this subsection the data types of a `C` type pseudocode that we use to describe algorithms:

- `typedef double pos;`
  Holds a nonnegative real number.
- `typedef struct` $\{$ `int` $|A|$; `int *` x $\}$ `path;`
  Holds a vector $x \in \{0,1\}^A$.
- `typedef struct` $\{$`int` m; `double *` π; $\}$ `dual;`
  Holds a vector $\pi \in \mathbb{R}^m$ (and later $\pi \in \mathbb{R}^R$).
- `typedef struct` $\{$`int` n; `pos *` ξ; $\}$ `primal;`
  Holds a vector $\xi \in \mathbb{R}^n_+$.
- `typedef struct` $\{$`int` m; `int` n; `int *` Φ`_cnt`; `int *` Φ`_ind`; $\}$ `matrix;`
  Holds a 0/1 matrix $\Phi \in \mathbb{R}^{m \times n}$ in column major format.
- `typedef struct` $\{$ `pos` γ; `dual` ϕ; $\}$ `col;`
  Holds objective γ and task incidence vector ϕ of a duty.
- `typedef struct` $\{$ `primal` ω; `matrix` Φ; `primal` ℓ; `dual` π; $\}$ `rdlp;`
  Holds objective ω, task incidence matrix Φ, lower bounds ℓ, and dual multipliers π of an RDLP.

2.5 Column Generation Method

Figure 1 describes a column generation algorithm for the solution of the MLP (3). This algorithm, denoted by `cgen`, is approximative with quality control parameters $\varepsilon > 0$ and $\delta > 0$. Three subroutines, `aug`, `price`, and `test`, are called in a loop. `aug` is an approximate RDLP solver, `price` is an approximate pricing routine, and `test` is an approximation control predicate. We stipulate the following properties for these subroutines:

```
rdlp cgen( rdlp {ω, Φ, ℓ, π}, pos ε, pos δ )
{
    int k ← 0;
    rdlp {ω_k, Φ_k, ℓ_k, π_k} ← {ω, Φ, ℓ, π};
    col {γ_k, φ_k};
    do {
        k ← k + 1;
        π_k ← aug({ω_{k−1}, Φ_{k−1}, ℓ, π_{k−1}});
        {γ_k, φ_k} ← price({ω_{k−1}, Φ_{k−1}, ℓ, π_k}, δ);
        {ω_k, Φ_k, ℓ_k} ← {ω_{k−1}, Φ_{k−1}, ℓ_{k−1}} ∪ {γ_k, φ_k, 0};
    };
    while ({ω_k, Φ_k} ≠ {ω_{k−1}, Φ_{k−1}}
        && test({ω_k, Φ_k, ℓ_k, π_k}, ε, δ) = 0);
    return {ω_k, Φ_k, ℓ_k, π_k};
}
```

Fig. 1. `cgen` column generation algorithm

- `dual aug( rdlp` $\{\omega, \Phi, \ell, \pi\}$ `);`
  `aug` must produce for an arbitrary, but fixed input RDLP $\{\omega, \Phi, \ell, \pi\}$ an iteration sequence $(\pi_k)_{k\in\mathbb{N}_0}$ by means of the recursion

$$\pi_0 = \pi \quad \text{and} \quad \pi_k = \texttt{aug}(\{\omega, \Phi, \ell, \pi_{k-1}\}), \quad k \in \mathbb{N}. \tag{8}$$

  This sequence is required to have the following properties:

$$\text{(AUG1)} \quad \limsup_{k\to\infty} \pi_k^{\mathrm{T}} \Phi \le \omega^{\mathrm{T}} \tag{9}$$

$$\text{(AUG2)} \quad \lim_{k\to\infty} \pi_k^{\mathrm{T}} \eta + \omega^{\mathrm{T}} \ell = \nu(\omega, \Phi, \ell). \tag{10}$$

  These conditions formulate feasibility and optimality of the sequence (π_k) in the limit. Note that $\nu(\omega, \Phi, \ell) < \infty$ implies that every limit point of (π_k) is optimal.
  Conditions (9) and (10) allow `aug` implementations as subgradient algorithms, as well as spacer step derivatives of such algorithms, in addition to simplex and interior point implementations.
- `col price( rdlp` $\{\omega, \Phi, \ell, \pi\}$`; pos` δ `);`
  If the *δ-pricing problem*

$$(\delta\text{-PRICE}) \quad \exists \begin{pmatrix} \gamma \\ \phi \end{pmatrix} \in \begin{pmatrix} \hat{\omega}^{\mathrm{T}} \\ \hat{\Phi} \end{pmatrix} : \overline{\overline{\gamma}} = (1+\delta)\gamma - \pi^{\mathrm{T}}\phi < 0, \tag{11}$$

  has a solution, `price` must return such a column; if no such *δ-negative (reduced cost)* column exists, `price` can return an arbitrary column. δ-PRICE requires only the identification of duties of "significantly" negative reduced cost; the problem is hence a weakening of the "strict" pricing problem (6),
  Section 3 discusses such an approximate pricing algorithm.

- `int test( rdlp` $\{\omega, \Phi, \ell, \pi\}$; `pos` ε; `pos` δ;);
  `test` returns 1 if the following two conditions, together referred to as $\varepsilon\delta$-*optimality*, hold, and 0 otherwise (recall (1) for the definition of $\tilde{\omega}$):

$$(\delta\text{-feasibility}) \quad \pi^{\mathrm{T}}\Phi \leq \tilde{\omega} \tag{12}$$

$$(\varepsilon\text{-optimality}) \quad (1+\varepsilon)\pi^{\mathrm{T}}\eta \geq \nu(\omega, \Phi, \ell) - \omega^{\mathrm{T}}\ell \tag{13}$$

  `test` can be implemented by LP techniques.

It is not hard to see that the ε- and δ-accuracies of the subroutines carry over to the entire column generation algorithm.

Proposition 1. *Let* $\eta_k = \mathbb{1} - \Phi_k \ell_k$. *Algorithm* `cgen` *has these properties:*

i) `cgen` *terminates after a finite number of iterations* κ.
ii) $\hat{\nu} - \hat{\omega}^{\mathrm{T}}\hat{\ell} \leq (1+\varepsilon)\pi_\kappa^{\mathrm{T}}\eta_\kappa$.
iii) $\hat{\nu} - \hat{\omega}^{\mathrm{T}}\hat{\ell} \geq (1+\delta)^{-1}\pi_\kappa^{\mathrm{T}}\eta_\kappa$.

Proof. i) RDLP is a sub-LP of an LP of finite dimension, namely, the dual of MLP. Hence, line 10 can only see a finite number of genuine additions of columns. From that point on, `cgen` merely iterates `aug`. This produces a sequence of multipliers that approaches the set of optimal solutions of RDLP, see (9) and (10). When (12) and (13) are satisfied, `test` breaks the main loop and `cgen` terminates at some iteration $k = \kappa$.

ii) $(1+\varepsilon)\pi_\kappa^{\mathrm{T}}\eta_\kappa \geq \nu(\omega_\kappa, \Phi_\kappa, \ell_\kappa) - \omega_\kappa^{\mathrm{T}}\ell_\kappa \geq \nu(\hat{\omega}, \hat{\Phi}, \hat{\ell}) - \hat{\omega}^{\mathrm{T}}\hat{\ell} = \hat{\nu} - \hat{\omega}^{\mathrm{T}}\hat{\ell}$.
The first inequality follows from (13), the second holds because each column is a constraint in RDLP.

iii) $(1+\delta)(\hat{\nu} - \hat{\omega}^{\mathrm{T}}\hat{\ell}) = (1+\delta) \max\limits_{\pi^{\mathrm{T}}\hat{\Phi} \leq \hat{\omega}^{\mathrm{T}}} \pi^{\mathrm{T}}\hat{\eta} = \max\limits_{\pi^{\mathrm{T}}\hat{\Phi} \leq (1+\delta)\hat{\omega}^{\mathrm{T}}} \pi^{\mathrm{T}}\hat{\eta} \geq \pi_\kappa^{\mathrm{T}}\eta_\kappa$.
The final inequality holds because `test` guarantees $\pi_\kappa^{\mathrm{T}}\Phi_\kappa \leq (1+\delta)\omega_\kappa^{\mathrm{T}}$ for all RDLP columns at iteration κ, see (12), while `price` takes responsibility for $\pi_\kappa^{\mathrm{T}}\hat{\Phi} \leq (1+\delta)\hat{\omega}^{\mathrm{T}}$ for all remaining columns, see (11).

Proposition 1 justifies the use of approximation algorithms in the LP resolution and pricing subroutines of a column generation algorithm. Such approximate procedures can help to bypass algorithmic bottlenecks in the subproblems.

Our system DS-OPT implements a *heuristic* `cgen` derivative:

- `aug` is implemented as a dual ascent heuristic as described in Sect. 4.
- `price` heuristically limits the search space, see Sect. 3.
- `test` terminates `cgen` if the relative progress $(\pi_{k+l}^{\mathrm{T}}\eta_{k+l} - \pi_k^{\mathrm{T}}\eta_k)/\pi_k^{\mathrm{T}}\eta_k$ made in some last l iterations falls below ε.

We close this subsection with the remark that Proposition 1 shows a way to compute a *global lower bound* for the entire master LP, and hence for the duty scheduling problem itself. The algorithmic challenge in the computation of such a bound is the solution of the δ-pricing problem (11). The potential of our methods with respect to the computation of lower bounds will be the subject of future research.

2.6 Duty Scheduling Method

The approximate solution of MLP is only a first step in duty scheduling. What we really want is an "acceptable" feasible integer solution.

Figure 2 gives pseudocode for an LP *plunging heuristic* to produce such a solution. This `dsopt` procedure is the top level routine of our duty scheduling system DS-OPT.

```
 1  rdlp dsopt( rdlp {ω, Φ, ℓ, π}, pos ε, pos δ, pos τ )
 2  {
 3       int k ← 0;
 4       int j_k;
 5       rdlp {ω_k, Φ_k, ℓ_k, π_k} ← {ω, Φ, ℓ, π};
 6       dual η_k ← 𝟙 − Φ_k ℓ_k;
 7       double Θ ← −∞;
 8       while (Φ_k^T ℓ_k < 𝟙) do {
 9            k ← k + 1;
10            if (π_k^T η_k + ω_k^T ℓ_k > Θ) {
11                 {ω_k, Φ_k, ℓ_k, π_k} ← cgen({ω_{k−1}, Φ_{k−1}, ℓ_{k−1}, π_{k−1}}, ε, δ);
12                 Θ ← (1 + τ)(π_k^T η_k + ω_k^T ℓ_k);
13            };
14            j_k ← chuzc({ω_k, Φ_k, ℓ_k, π_k});
15            ℓ_k ← ℓ_{k−1} + e_{j_k};
16            η_k ← η_{k−1} − (Φ_k)_{·j_k};
17            π_k ← aug({ω_k, Φ_k, ℓ_k, π_k});
18       };
19       return {ω_k, Φ_k, ℓ_k, π_k};
20  }
```

Fig. 2. `dsopt` duty scheduling heuristic

`dsopt` calls the subroutines `cgen` and `chuzc` in a loop. `chuzc` is a routine to select a variable that is subsequently fixed to 1 in lines 15 to 16 of `dsopt`. The calls to `cgen` are controlled by a *branch-and-generate* scheme involving parameters $\tau > 0$ and Θ. These subroutines work as follows:

- `int chuzc` (`rdlp` $\{\omega, \Phi, \ell, \pi\}$);
  `chuzc` returns an ℓ-compatible column j, i.e., $\Phi(\ell + e_j) \leq \mathbb{1}$.
  An implementation will be described in Sect. 5.
- Branch-and-Generate Scheme.
  `dsopt` monitors a *trust region*

$$\pi_k^{\mathrm{T}}\eta_k + \omega_k^{\mathrm{T}}\ell_k \leq \Theta \tag{14}$$

for the RDLP objective $\pi_k^{\mathrm{T}}\eta_k + \omega_k^{\mathrm{T}}\ell_k$ at hand. As long as this value stays below Θ, RDLP is considered as an acceptable approximation of the

global MLP and `cgen` is bypassed. Such a "fast iteration" merely fixes a variable and adjusts the RDLP solution by a call to `aug` in line 17. When the objective goes above Θ, this is taken as an indication that the variable fixes have *potentially* changed the RDLP to such an extent that it does no longer approximate the (also changed) global MLP well. In this situation, `cgen` is rerun to update the RDLP and a new trust region is established up to τ percent above the updated objective value.
We have learned about such a control scheme from a talk of Marsten, who termed it *branch-and-generate* (BANG) and implemented it in the DOC (Delta Optimizer for Crews) system. BANG does, however, not seem to have been documented in the literature.

We remark that an efficient `dsopt` implementation will safeguard against the generation of incompatible columns $(\gamma_k, \phi_k^{\mathrm{T}})$, i.e., $\phi_k^{\mathrm{T}} \Phi_k \ell_k \neq 0$ in the `cgen` column generation routine. A convenient, but notationally not always elegant way is to set $\pi_i = -\infty$ for $\eta_i = 0$.

Our system DS-OPT is not designed to produce optimal solutions to duty scheduling problems. The focus is on the ability to process large scenarios in a reasonable way. In particular, we aim to exploit all degrees of freedom in the duty network and in the duty types. DS-OPT implements this strategy using approximative and, where we think it expedient, heuristic techniques.

3 Constrained Shortest Paths

This section deals with the solution of the δ-pricing problem (11). We propose an integer linear programming formulation for this problem that we tackle with Lagrangean relaxation techniques. It turns out that such a relaxation gives rise to lower bounds that can be used to speed up enumerative pricing algorithms.

We model the pricing problem in terms of acyclic constrained shortest path problems (ACSPs). The different constraints of different duty types are handled using separate ACSPs. We will discuss in the remainder of this section the treatment of a single duty type.

3.1 Integer Programming Model

Denote by $x \in \{0,1\}^A$ the incidence vector of a directed $(0, m+1)$-path in the duty scheduling network $D = (V, A)$ representing a duty, by $c \in \mathbb{R}^A$ a vector of costs associated with the arcs, and by $N \in \{-1,0,1\}^{V \times A}$ the node-arc incidence matrix of D. Let further $W \in \mathbb{R}^{R \times A}$ be a matrix that records in each entry w_{ra} the consumption a so-called *resource* r on traversal of the arc a, let b_r be a *goal* or best resource consumption for a duty, let $s_r = (s_r^+, s_r^-)$ be a pair of slack and surplus variables, respectively, that gather deviations from this goal value, let $u_r = (u_r^+, u_r^-) \in (\mathbb{R}_+ \cup \{\infty\})^2$ be upper limits for such deviations, and let $d_r = (d_r^+, d_r^-) \in \mathbb{R}_+^2$ be *nonnegative* objective penalties associated with the slack and surplus variables. (We do not discuss the treatment of bonuses.) We remark that we use this resource constraint model also to express "infeasible path constraints", see below. Let finally

$$c^{\mathrm{T}}x + d^{\mathrm{T}}s = c^{\mathrm{T}}x + d^{+\mathrm{T}}s^{+} + d^{-\mathrm{T}}s^{-} \tag{15}$$

be the *genuine cost* associated with a vector $(x, s) \in \mathbb{R}^m \times \mathbb{R}^R \times \mathbb{R}^R$, let $\pi \in \mathbb{R}^m$ be some vector of shadow prices, and consider a transformed cost

$$\bar{\tilde{c}}^{\mathrm{T}}x + \tilde{d}^{\mathrm{T}}s = (1+\delta)(c^{\mathrm{T}}x + d^{\mathrm{T}}s) - \sum_{j=1}^{m}\sum_{ij \in A} \pi_j x_{ij} \tag{16}$$

that arises from the genuine one by a scaling operation (recall (1)) and a subsequent subtraction of certain shadow prices.

We propose to model the single duty type pricing problem as an ACSP subject to *linear side constraints* and *linear objective penalties*:

$$\begin{aligned}
\text{(ACSP)} \qquad \min\ & \bar{\tilde{c}}^{\mathrm{T}}x + \tilde{d}^{\mathrm{T}}s && (17)\\
Nx &= e_{m+1} - e_0 && (18)\\
Wx + s^{+} - s^{-} &= b && (19)\\
0 \le s &\le u && (20)\\
x &\in \{0,1\}^A. && (21)
\end{aligned}$$

We list two conventions that will be used in the following discussion:

- $s(x) = (s^{+}(x), s^{-}(x)) = ([b - Wx]^{+}, [Wx - b]^{+})$.
- $M(x) = \{1 \le i \le m : x(\delta^{+}(i)) = 1\}$.

(18) are flow conservation constraints that define an $(0, m+1)$-path. (19) and (20) are called *resource constraints*. Note that our assumption $\tilde{d} \ge 0$ sets the penalties automatically to the right values $s(x)$. Let finally $(x, s(x))$ be an ACSP solution of genuine cost $\gamma = c^{\mathrm{T}}x + d^{\mathrm{T}}s(x)$, and let $\phi = \chi^{M(x)}$ be the task incidence vector of the visited duty elements. Consideration of the transformed objective (16)/(17)

$$\bar{\tilde{c}}^{\mathrm{T}}x + \tilde{d}^{\mathrm{T}}s = (1+\delta)(c^{\mathrm{T}}x + d^{\mathrm{T}}s) - \sum_{j=1}^{m}\sum_{ij \in A} \pi_j x_{ij} = (1+\delta)\gamma - \pi^{\mathrm{T}}\phi$$

shows that ACSP is equivalent to the δ-pricing problem (11) for a duty type that is given by the constraints (19) and (20) and the genuine objective (15).

Linear resource constraints are not only versatile modeling tools. Perhaps even more important, such constraints arise naturally in duty scheduling applications. We list some important types in a simplified format that can easily be transformed into the general form (19) and (20):

- Max constraints $W_{r\cdot}x \leq b_r$
  Max constraints can be used to stipulate upper limits on the consumption of resources such as driving time, working time, duty time, shift time, number of pieces of work, number of parts of work, etc.
- Min constraints $W_{r\cdot}x \geq b_r$
  Min constraints can set lower limits on the same quantities. They can also be used to enforce absolute and, in the form $W_{r\cdot}x \geq q \cdot W_{r\cdot}x$, $0 \leq q \leq 1$, relative minimum break times in a duty.
- Opt constraints $W_{r\cdot}x + s_r^+ - s_r^- = b_r$ with penalties $d_r^+ s_r^+ + d_r^- s_r^-$
  Opt constraints and the associated objective penalties can model linear overtime bonuses and the like costs.
- Infeasible path constraints $\chi_P^{\mathrm{T}} x \leq |P| - 1$
  Some difficulties come up in the treatment of "nonlinear" rules that govern, e.g., break positions in a duty. Such constraints can be fitted within a linear framework using *infeasible path constraints* (IPCs) that rule out individual infeasible paths. We call a set of IPCs that represent a nonlinear rule an *IPC rule complex*. Such constraints must be treated implicitly. IPCs were introduced in [4] in the context of asymmetric TSPs with time windows.

Typical duty scheduling problems feature duty types with about 10 different explicit linear resources and half a dozen implicit IPC rule complexes on break positions, sign-on and sign-off issues, the construction of composite duties with several parts, and certain compensations.

The ACSP is well studied in the literature, see [46] for a survey. The problem is $\mathcal{NP}$-hard already for a single resource constraint, see [37, A2.3, ND30]. For any fixed number of resources, fully polynomial approximation schemes exist, see [53]. Pseudopolynomial algorithms have been developed and successfully used in practical applications, see [24] and others, including penalty treatment, see [25]. Enumerative approaches using Lagrangean lower bounding techniques have been studied in [39], [48,49], and [9]. A computational comparison of different methods can be found in [46]; this reference gives also a combinatorial algorithm of low complexity to solve the LP relaxation associated with ACSP for the case of a single resource constraint.

However, as far as we know, the model ACSP has not played a role in the context of duty scheduling. The duty scheduling literature focuses on alternative and well-known *time window formulations*, see [32], [24], [27–29].

3.2 Lagrangean Lower Bound

We follow now [39] and [9] in using a Lagrangean relaxation of the ACSP to derive a class of lower bounds that will be used to establish an efficient backtracking criterion in an enumerative ACSP algorithm; this idea has also been used in [46] to develop essentially the same algorithm.

The *Lagrangean shortest path relaxation* LSP that we use is obtained by transferring the resource constraints (19) into the objective:

$$\text{(LSP)} \quad \max_{y \in \mathbb{R}^R} P(y) + Q(y) + y^{\mathrm{T}} b \tag{22}$$

$$\text{(SP)} \quad P(y) = \min \ (\bar{\tilde{c}}^{\mathrm{T}} - y^{\mathrm{T}} W)x, \quad Nx = e_{m+1} - e_0, \quad x \geq 0 \tag{23}$$

$$\text{(BP)} \quad Q(y) = \min \ (\tilde{d}^{+} - y)^{\mathrm{T}} s^{+} + (\tilde{d}^{-} + y)^{\mathrm{T}} s^{-}, \quad 0 \leq s \leq u. \tag{24}$$

LSP decomposes into an acyclic shortest path problem SP and a simple linear program BP over a box. BP has a closed formula solution, e.g.,

$$s^* = s^*(y) = \frac{1}{2}\left(I - \operatorname{diag}\left(\operatorname{sign}(\tilde{d}^{+} - y), \operatorname{sign}(\tilde{d}^{-} + y)\right)\right) u. \tag{25}$$

An optimal solution $x^* = x^*(y)$ of SP can be obtained combinatorially using, e.g., the reaching algorithm as proposed in [1]. We are also interested in an optimal solution $h^* = h^*(y)$ of the dual

$$\text{(DP)} \quad \max \ h_{m+1} - h_0, \quad h^{\mathrm{T}} N \leq \bar{\tilde{c}}^{\mathrm{T}} - y^{\mathrm{T}} W. \tag{26}$$

Such a solution can be interpreted in terms of distance labels h_v^* that give, for each node v, the minimum distance $h_v^* - h_0^*$ from the source with respect to the objective $\bar{\tilde{c}}^{\mathrm{T}} - y^{\mathrm{T}} W$. These values underestimate, in particular, the distances along all constrained paths. Combining this bound with the solution of BP suggests to consider the following *Lagrangean distance labels*:

$$g_v^* = g_v^*(y) = h_v^*(y) - h_0^*(y) + Q(y) + y^{\mathrm{T}} b, \quad v \in V. \tag{27}$$

Lemma 1. *Let x^1 and x^2 be the incidence vector of a $(0, v)$- and a $(v, m+1)$-path in D, respectively. Suppose that $x^1 + x^2$ is feasible for ACSP. Let $y \in \mathbb{R}^R$ be any vector of Lagrangean multipliers for LSP and $g^*(y)$ be the associated Lagrangean distance labels (27). Then:*

$$g_v^*(y) + (\bar{\tilde{c}}^{\mathrm{T}} - y^{\mathrm{T}} W)x^2 \leq \bar{\tilde{c}}^{\mathrm{T}}(x^1 + x^2) + \tilde{d}^{\mathrm{T}} s(x^1 + x^2). \tag{28}$$

Lemma 1 is useful in an algorithm that constructs δ-negative paths "backwards" starting from the sink, working toward the source. (28) can be used in such an algorithm to prune the search whenever $g_v^*(y) + (\bar{\tilde{c}}^{\mathrm{T}} - y^{\mathrm{T}} W)x^2 \geq 0$.

3.3 Lagrangean Distance Computation

The quality of the distance labels $g^*(y)$ depends on the identification of suitable multipliers y. This task can be tackled by a subgradient algorithm. The work per iteration consists in solving SP and BP.

We state on this occasion a characterization of the subdifferential $\partial(y)$, which is readily available in this application, see [10, Prop. 6.1.2].

Corollary 1. *Let $y \in \mathbb{R}^R$ by a vector of Lagrangean multipliers for LSP. Let h^* be the solution of the associated shortest path problem DP. Denote by $\overline{A} = \{a \in A : \bar{\bar{c}}_a - y^{\mathrm{T}} W_{\cdot a} > h^{*\mathrm{T}} N_{\cdot a}\}$ the set of all arcs that can not be contained in an optimal solution to SP. Then:*

$$\partial(y) = \operatorname{conv}\{b - Wx - s^{*+} + s^{*-} : Nx = e_{m+1} - e_0, \quad x \geq 0, \quad x(\overline{A}) = 0\}.$$

```
dual lsp ( dual y )
{
    primal x;
    dual s⁺, s⁻;
    int ϱ;
    for (;;) {
        ϱ ← [select r ∈ R with W_{r·}x + s_r⁺ − s_r⁻ − b_r ≠ 0];
        x ← x*(y);
        s ← s*(y);
        y_ϱ ← y_ϱ + θ([stepsize control])sign(W_{r·}x + s_r⁺ − s_r⁻ − b_r);
        if [convergence] return y;
    };
}
```

Fig. 3. `lsp` Lagrangean distance computation

It turns out, however, that a *coordinate ascent method* is good enough in practice. Pseudocode for such an algorithm, which adjusts a single multiplier in each iteration, is given in Fig. 3.

3.4 Constrained Shortest Path Algorithm

Figure 4 gives pseudocode for a depth first search enumeration algorithm for the ACSP that uses the Lagrangean distance labels (27) as a backtracking criterion. The distance labels are derived from a *relaxation* of the ACSP that one obtains by simply ignoring all IPCs. Starting from the sink, the algorithm computes recursively all possible $(0, m+1)$-paths. A backtrack is done in the following cases:

- Line 10.
  The path x violates an IPC. Checking an entire duty for IPC violations is in general not a problem, because the rules are made for this purpose. Detecting inevitable violation as early as possible in a partial duty is, however, certainly difficult. We use heuristic criteria that would take a detailed description of data structures in order to be discussed.
- Line 11.
  The source is reached, an improving duty is found (line 12), or not.

```
1  col price( rdlp {ω, Φ, ℓ, π}, dual y )
2  {
3      dual g* ← g*(y);
4      return dfs ( 0, g*, (+∞, 0) );
5  }
6
7  col dfs( path x, dual g*, col (γ, φ) )
8  {
9      int v ← tail(x);
10     if (x [violates an IPC]) return (γ, φ);
11     if (v = 0) {
12         if (c̿ᵀx + d̃ᵀs(x) < γ)
13             (γ, φ) ← (c̿ᵀx + d̃ᵀs(x), χ^{M(x)});
14         return (γ, φ);
15     }
16     if ([search limit exceeded]) return (γ, φ);
17     forall (u ∈ γ⁻(v)) {
18         if (c̿ᵀx + c̿_uv + g*_u ≥ max{γ, 0}) continue;
19         (γ, φ) ← dfs(x + e_uv, g*, (γ, φ));
20     }
21     return (γ, φ);
22 }
```

Fig. 4. `price` column generator (single duty type)

- Line 16.
  Some search limit is exceeded. Such search limits can help to adjust the performance of an implementation to a particular scenario. There is, however, another use that is even more important. Our `dsopt` implementation differs from the one in Fig. 4 in generating not only one δ-negative duty, but bunches of several thousands of such duties. We use search limits as a heuristic means to produced "diverse" paths in an attempt to speed up the overall convergence of the algorithm. Diversification techniques of this type have been mentioned in the literature, e.g., in [23].
- Line 18.
  The Lagrangean lower bounding criterion shows that the path x can not be extended into a δ-negative reduced cost duty.

We remark that we also use the Lagrangean distance labels for preprocessing purposes, see [9] for details.

4 Coordinate Ascent

We study in this section the $\varepsilon\delta$-optimal solution (recall (13) and (12)) of RDLP. Such a solution can be obtained using sophisticated LP codes. Large

RDLPs beyond 1,000 rows are, however, often hard to solve leading to LP performance problems as frequently reported in the literature, see, [2, p. 241/242], see also [45], [43], [13], and [33].

The search for alternatives focuses on subgradient algorithms and on dual ascent methods that are both based on Lagrangean relaxations. Subgradient algorithms have been used in [16], [15], and [34]. Dual ascent methods have been studied, among others, in [5], [8], [36], and [54]. See also [35, p. 9] for a survey on dual ascent applications in general.

4.1 Coordinate Ascent for RDLPs

We recall in this subsection the basics of coordinate ascent in an RDLP context. Starting point is the following Lagrangean relaxation of the RDLP:

$$\text{(LRDLP)} \quad \max_{\pi \in \mathbb{R}^m} \min \ \Lambda(\pi, \xi), \quad \ell \leq \xi \leq \mathbb{1} \tag{29}$$

where

$$\Lambda(\pi, \xi) = \omega^{\mathrm{T}}\xi - \pi^{\mathrm{T}}(\Phi\xi - \mathbb{1}) = \overline{\omega}^{\mathrm{T}}\xi + \pi^{\mathrm{T}}\mathbb{1}. \tag{30}$$

For fixed π, LRDLP has the closed formula solution(s)

$$\xi_j^* = \xi_j^*(\pi) = \begin{cases} 1, & \text{if } \overline{\omega}_j < 0 \\ \ell_j \leq \xi_j^* \leq 1, & \text{if } \overline{\omega}_j = 0 \\ \ell_j, & \text{if } \overline{\omega}_j > 0. \end{cases} \tag{31}$$

For given π, the *line search problem* along the i-th coordinate is

$$\begin{aligned} \text{(LS)} \quad & \max_{\alpha \in \mathbb{R}} \min \ \Lambda(\pi + \alpha \cdot e_j, \xi), \quad \ell \leq \xi \leq \mathbb{1} \\ &= \max_{\alpha \in \mathbb{R}} \ \pi^{\mathrm{T}}\mathbb{1} + \alpha + \min \ (\overline{\omega}^{\mathrm{T}} - \alpha \cdot \Phi_{i\cdot})\xi, \quad \ell \leq \xi \leq \mathbb{1}. \end{aligned} \tag{32}$$

LS has also a closed formula solution

$$\alpha^* \in \begin{cases} [\min\limits_{j \in \Phi_{i\cdot}} \overline{\omega}_j, \min\limits_{j \in \Phi_{i\cdot}}{}_2\, \overline{\omega}_j], & \text{if } \eta_i = 1 \\ (-\infty, \min\limits_{j \in \Phi_{i\cdot}} \overline{\omega}_j], & \text{if } \eta_i = 0. \end{cases} \tag{33}$$

α^* can be determined easily by a sweep over the nonzeros of the i-th row of the matrix Φ.

The basic coordinate ascent method simply iterates such line searches along the coordinates. It is well known that this method is not globally convergent, because it can get stuck at a nonoptimal point from which it is impossible to ascend along any coordinate axis, see, e.g., [10, § 6.3.3].

4.2 Boxstep

A problem that we have observed in coordinate ascent computations is that some of the multipliers tend to take unreasonably large values in early iterations that are hard to readjust later on. This phenomenon has been documented before in a simplex context in [33]. We use *boxstep*, a framework due to [44], to counterbalance this effect.

Boxstep is a method to optimize a concave and upper semicontinuous function $\nu : D \to \mathbb{R}$ on a compact domain $D \subseteq \mathbb{R}^m$

$$\text{(GLOBAL)} \quad \max_{\pi \in D} \ \nu(\pi). \tag{34}$$

The method associates with each point $\pi_0 \in D$ a *local problem*

$$\text{(LOCAL)} \quad \max \ \nu(\pi), \quad \pi \in D, \quad ||\pi - \pi_0||_\infty \leq \beta \tag{35}$$

and iterates the recursion

$$\pi_{k+1} = \text{argmax} \ \nu(\pi), \quad \pi \in D, \quad ||\pi - \pi_k||_\infty \leq \beta, \tag{36}$$

until $\nu(\pi_{k+1}) \leq \nu(\pi_k) + \varepsilon$. $\beta > 0$ is a parameter that controls the size of the domain in the local problem, and $\varepsilon > 0$ is a termination parameter. It has been proven in [44] that boxstep terminates in a finite number of iterations κ with a solution π_κ such that $\nu(\pi_\kappa) \geq \max \ \nu(D) - 2\varepsilon\Delta/\beta$, where $\Delta = \operatorname{diam} D = \max_{x,y \in D} ||x - y||_2$ denotes the diameter of D. This result carries over to our situation as follows.

Corollary 2. *Suppose RDLP contains the "tripper" duties* $\binom{L}{e_i} \in \binom{\omega^{\mathrm{T}}}{\Phi}$, $i = 1, \ldots, m$, *of cost* $L \in \mathbb{R}_+$. *Then:*

i) RDLP has an optimal solution $\pi^* \in [-(m-1)L, L]^m$.
ii) Boxstep finds, for any starting point $\pi_0 \in [-(m-1)L, L]^m$, *in finite time an RDLP solution* π_κ *such that* $\pi_\kappa^{\mathrm{T}}\eta \geq \nu(\omega, \Phi, \ell) - 2\varepsilon m^{3/2} L/\beta$.

Proof. i) Note that $\pi^{\mathrm{T}} I \leq L\mathbb{1}^{\mathrm{T}}$ because of the trippers. The further restriction $\pi \geq -(m-1)L\mathbb{1}$ does not change the optimum:

$$\begin{aligned}
&\max \ \pi^{\mathrm{T}}\mathbb{1} + \lambda^{\mathrm{T}}\ell, \quad \pi^{\mathrm{T}}\Phi + \lambda^{\mathrm{T}} = \omega^{\mathrm{T}}, \quad -\pi^{\mathrm{T}} \leq (m-1)L\mathbb{1}^{\mathrm{T}}, \quad \lambda \geq 0 \\
=&\min \ \omega^{\mathrm{T}}\xi + (m-1)L\mathbb{1}^{\mathrm{T}}\mu, \quad \Phi\xi - \mu = \mathbb{1}, \quad \xi \geq \ell, \quad \mu \geq 0 \\
=&\min \ \omega^{\mathrm{T}}\xi, \quad \Phi\xi = \mathbb{1}, \quad \xi \geq \ell \\
=&\min \ \omega^{\mathrm{T}}\xi, \quad \Phi\xi = \mathbb{1}, \quad \ell \leq \xi \leq \mathbb{1}.
\end{aligned}$$

The second equation follows by replacing overcovers with trippers.

ii) $\operatorname{diam}[-(m-1)L, L]^m = \sqrt{m \cdot m^2 L^2} = m^{3/2} L$.

4.3 Coordinate Ascent Algorithm

Figure 5 gives pseudocode for a coordinate ascent heuristic that combines a coordinate search algorithm with a boxstep steplength control in line 9. The solution formula for the line search problem in line 8 contains a control parameter $0 < \vartheta < 1$ that is used for performance adjustments.

```
dual aug( rdlp {ω, Φ, ℓ, π} )
{
     int i ← 0, k ← 0;
     double α;
     for ( ;; ) {
          for (i = 1; 1 ≤ m; i++) {
               if η_i = 0 continue;
               α ← ϑ min_{j∈Φ_i·} ϖ_j + (1 − ϑ) min2_{j∈Φ_i·} ϖ_j ;
               α ← [α]_{−β}^{+β} ;
               π_i ← π_i + α;
          };
          if ([convergence]) break;
     };
     return π;
}
```

Fig. 5. aug coordinate ascent algorithm

5 Variable Fixing

We discuss in this section the heuristic fixing of duties in the branch-and-generate phase of our algorithm. The fixings are derived in two stages from the Lagrangean relaxation LRDLP associated with the current restricted dual master LP. The first stage selects a set of *candidates* using scoring techniques. The second stage selects a variable from this candidate set by Lagrangean probing. Figure 6 gives the pseudocode for such a variable selection scheme.

To explain the routine, recall the Lagrangean relaxation

$$\text{(LRDLP)} \quad \max_{\pi\in\mathbb{R}^m} \min \ \omega^{\mathrm{T}}\xi - \pi^{\mathrm{T}}(\Phi\xi - \mathbb{1}), \quad \ell \le \xi \le \mathbb{1}. \tag{37}$$

Let $\pi \in \mathbb{R}^m$ be some vector of multipliers and $\xi^* = \xi^*(\pi)$ (cf. (31)) be the associated primal solution. We want to identify in this situation a variable that is likely to be contained in an "acceptable" feasible solution for SPP.

5.1 Stage 1: Scoring

The literature suggests the computation of a *score* value function

$$\sigma_j = \sigma_j(\omega_j, \pi, \Phi_{\cdot j}, \eta), \quad j = 1, \ldots, n \tag{38}$$

for each variable as a heuristic measure for the attractiveness of fixing this variable. A variety of rules have been studied, see [5], [8], [36], [16], and [15]. Among the more popular scores is the "average reduced cost per uncovered row"

$$\sigma_j = \overline{\omega}_j / \Phi_{\cdot j}^{\mathrm{T}}\eta, \tag{39}$$

```
int chuzc( rdlp {\omega, \Phi, \ell, \pi} )
{
    int j;
    primal \sigma, J, \theta;
    for (j = 1; j \le n; j++)
        \sigma_j \leftarrow \Phi_{\cdot j}^T \Phi \ell = 0 ? \overline{\omega}_j / \Phi_{\cdot j}^T \eta : +\infty;
    J \leftarrow {argmin_1 \sigma, ..., argmin_{20} \sigma};
    forall (j \in J) {
        dual \lambda \leftarrow aug({\omega, \Phi, \ell + e_j, \pi});
        \theta_j \leftarrow \lambda^T(\eta - \Phi_{\cdot j}) + \omega^T(\ell + e_j);
    }
    j \leftarrow argmin_{j \in J} \theta;
    return j;
}
```

Fig. 6. `chuzc` duty fixing

which is a major component of the score that we use. Selecting a set J of some 20 compatible candidates of smallest score concludes stage 1 of our fixing heuristic.

5.2 Stage 2: Lagrangean Probing

Stage 2 tries to improve the scoring information by tentatively fixing variables and exploring the consequences in an LRDLP reoptimization. This technique is called *probing* and is discussed, e.g., in [47] and, in a strong branching context, in [3]. We compute for each candidate variable $j \in J$ with our `aug` routine an *up-penalty*

$$\theta_j = \underset{\pi \in \mathbb{R}^m}{\mathtt{aug}} \ \omega^{\mathrm{T}}\xi + \pi^{\mathrm{T}}(\Phi\xi - \mathbb{1}), \quad \ell + e_j \le \xi \le \mathbb{1}, \qquad j \in J, \tag{40}$$

and fix the one with the smallest penalty.

6 Computation

We report in this section computational results obtained with our duty scheduling system `dsopt`. All tests have been performed within the MICROBUS II system on commonplace PCs.

Table 6 lists characteristics and results for three European scenarios. prob1 and prob2 are urban bus crew scheduling problems from German operations, prob3 is a subway driver scheduling problem from a European metropolis.

The top of the table gives some information about the characteristics of these problems. The rows "# vehicle blocks" ("#" means "number of")

Table 1. Solving duty scheduling problems with `dsopt`

	prob1	prob2	prob3
# vehicle blocks	63	56	59
incl. spares	yes	no	no
# duty elements	727	1065	1968
# tasks (nodes)	7037	24284	51488
# links	93725	261971	277681
duty type categories	straigths	straigths	straights
	splits	splits	splits
		shorts	shorts
opt working time	7:42/7:42	8/8/5	5:00–5:45

manual DS in MICROBUS II			
status	100 %	99 %	100 %
# duties	88	73	122
∅ working time	7:27	7:03	n.a.

DS-OPT			
status	100 %	100 %	100 %
# duties	85	63	111
∅ working time	7:06	7:50	5:41
running time	0:45[1]	1:20[a]	5:30[b]
required MB	150	230	230

[a] Pentium III 650 MHz, 578 MB memory, Linux 2.2.14, gcc-2.95.2
[b] Pentium III 500 MHz, 1 GB memory, Windows NT, VC++ 6.0

and "including spares" are self-explanatory and show that we deal here with relatively, albeit not extremely large problems on multiple lines, i.e., we allow changeovers and do not plan on a line-by-line basis. Cutting the vehicle blocks from line "# vehicle blocks" at the relief points, we obtain the "# duty elements" as listed in the next line. The large number in prob3 results from this company's willingness to relieve basically anytime at any subway station. The duty scheduling network is set up from this data in two steps. First, nodes for possible extension elements (extension elements take care, e.g., of sign-on and sign-off activities) are added. (We enumerate all possible extensions and add a node for every single one, see [14, in German] for details.) This results in a large "# nodes", but it encapsulates the treatment of extensions in the network data. It also turns out that such an *a priori enumeration* has no negative effect on the running time. The network is completed by adding the given "# links" for locally feasible transitions among duty and extension elements. We attach importance to the inclusion of *all* possible links in this step, because each link is a potentially valuable degree of freedom. The next three lines give information on "duty type categories". Each listed category represents several duty types. The last line reports, as one example of the duty type parameters, the desired average working time.

The second part of the table reports statistics on the best solutions that have been obtained by the companies. The third part lists the DS-OPT solutions. The numbers show that the optimized solutions contain sometimes significantly less duties, while the scheduling quality in terms of rule compliance is at least comparable ("status" is the percentage of duty elements that have been scheduled into duties).

Time and memory requirements for these runs are listed in the fourth part of the table. It is apparent that, also from this point of view, optimization systems such as DS-OPT definitely qualify as production tools for operational planning in public transit.

References

1. Ahuja R.K., Magnanti T.L., Orlin J.B. (1993) Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs, New Jersey
2. Andersson E., Housos E., Kohl N., Wedelin D. (1998) Crew Pairing Optimization. In [57], Chap. 8
3. Applegate D., Bixby R.E., Chvátal V., Cook W. (1995) Finding Cuts in the TSP. Technical Report 95-05, DIMACS, Rutgers University, New Jersey
4. Ascheuer N., Fischetti M., Grötschel M. (1997) A Polyhedral Study of the Asymmetric Travelling Salesman Problem with Time Windows. Preprint SC 97-11, Konrad-Zuse-Zentrum, Berlin (available at `http://www.zib.de/ZIBbib/Publications/`)
5. Balas E., Ho A. (1980) Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization: A Computational Study. Math. Prog. 12:37–60
6. Ball M.O., Magnanti T.L., Monma C.L., Nemhauser G.L. (Eds.) (1995) Network Routing, Vol. 8 of Handbooks in Operations Research and Management Science. Elsevier, Amsterdam

7. Barnhart C., Johnson E.L., Nemhauser G.L., Savelsbergh M.W.P., Vance P.H. (1994) Branch-and-Price: Column Generation for Solving Huge Integer Programs. In [11], pp. 186–207
8. Beasley J. (1987) An Algorithm for Set Covering Problem. Europ. J. on OR 31:85–93
9. Beasley J., Christofides N. (1989) An Algorithm for the Resource Constrained Shortest Path Problem. Networks 19:379–394
10. Bertsekas D.P. (1995) Nonlinear Programming. Athena, Belmont, Massachusetts
11. Birge J.R., Murty K.G. (Eds.) (1994) Mathematical Programming: State of the Art 1994. University of Michigan.
12. Bixby R.E., Gregory J.W., Lustig I.J., Marsten R.E., Shanno D.F. (1992) Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods. Op. Res. 40:885–897
13. Borndörfer R. (1998) Aspects of Set Packing, Partitioning, and Covering. Ph.D. Thesis, Technical University Berlin (available at `http://www.zib.de/ZIBbib/Publications/`). Shaker, Aachen.
14. Borndörfer R., Löbel A., Strubbe U., Völker M. (1999) Zielorientierte Dienstplanoptimierung. In Heureka '99: Optimierung in Verkehr und Transport. Forschungsgesellschaft für Straßen- und Verkehrswesen, Köln, pp. 171–194 (available as Preprint SC 98-41 at URL `http://www.zib.de/ZIBbib/Publications/`)
15. Caprara A., Fischetti M., Toth P. (1996) A Heuristic Algorithm for the Set Covering Problem. In Cunningham W.H., McCormick S.T., Queyranne M. (Eds.) Integer Programming and Combinatorial Optimization, Proc. of the 5th Int. IPCO Conf., Vancouver, British Columbia, Canada, pp. 72–84
16. Ceria S., Nobili P., Sassano A. (1995) A Lagrangian-Based Heuristic for Large-Scale Set Covering Problems. Working Paper, University of Roma La Sapienza
17. Crainic T.G., Laporte G. (Eds.) (1998) Fleet Management and Logistics. Kluwer, Dordrecht
18. Daduna J.R., Branco I., Paixão J.M.P. (Eds.) (1995) Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems, Springer, Berlin
19. Daduna J.R., Wren A. (Eds.) (1988) Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems, Springer, Berlin
20. Dell'Amico M., Maffioli F., Martello S. (Eds.) (1997) Annotated Bibliographies in Combinatorial Optimization. Wiley, Chichester
21. Desaulniers G., Desrosiers J., Ioachim I., Solomon M.M., Soumis F., Villeneuve D. (1998) A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems. In [17], Chap. 3, pp 57–93
22. Desaulniers G., Desrosiers J., Lasry A. (1999) Crew Pairing for a Regional Carrier. In [56], pp. 19–42
23. Desaulniers G., Desrosiers J., Solomon M.M. (1999) Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems. Technical Report G-99-36, Les Cahiers du GERAD, Montreal
24. Desrochers M. (1986) A New Algorithm for the Shortest Path Problem with Resource Constraints. Technical Report 421A, Centre de Recherche sur les Transports, University of Montréal

25. Desrochers M., Gilbert J., Sauvé M., Soumis F. (1992) Crew-Opt: Subproblem Modeling in a Column Generation Approach to Urban Crew Scheduling. In [26], pp. 395–406
26. Desrochers M., Rousseau J.-M. (Eds.) (1992) Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems, Springer, Berlin
27. Desrochers M., Soumis F.C. (1988a) A Generalized Permanent Labelling Algorithm for the Shortest Path Problem with Time Windows. Infor 26:191–212
28. Desrochers M., Soumis F.C. (1988b) A Reoptimization Algorithm for the Shortest Path Problem with Time Windows. Europ. J. on OR 35:242–254
29. Desrochers M., Soumis F.C. (1989) A Column Generation Approach to the Urban Transit Crew Scheduling Problem. Transp. Sci. 23:1–13
30. Desrosiers J., Dumas Y., Solomon M.M., Soumis F. (1995) Time Constrained Routing and Scheduling. In [6], Chap. 2, pp. 35–139
31. Desrosiers J., Rousseau J.-M. (1995) Results Obtained with Crew-Opt: A Column Generation Method for Transit Crew Scheduling. In [18], pp. 349–358
32. Desrosiers J., Soumis F., Desrochers M. (1984) Routing with Time Windows by Column Generation. Networks 14:545–565
33. du Merle O., Villeneuve D., Desrosiers J., Hansen P. (1999) Stabilized Column Generation. Disc. Math. 194:229–237
34. Fischetti M., Kroon L. (1999) Scheduling Train Drivers and Guards: The Dutch "Noord-Oost" Case. Manuscript
35. Fisher M.L. (1981) The Lagrangian Relaxation Method for Solving Integer Programming Problems. Mgmt. Sci. 27:1–18
36. Fisher M.L., Kedia P. (1990) Optimal Solution of Set Covering/Partitioning Problems Using Dual Heuristics. Mgmt. Sci. 39:674–688
37. Garey M.R., Johnson D.S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York
38. Grötschel M., Lovász L., Schrijver A. (1988) Geometric Algorithms and Combinatorial Optimization. Springer, Berlin
39. Handler G.Y., Zang I. (1980) A Dual Algorithm for the Constrained Shortest Path Problem. Networks 10:293–310
40. Kammler M. (2000) Optimierungsverfahren für Dienstpläne im ÖV. Nahverkehrspraxis 7/8:22
41. Leuthardt H. (1998) Kostenstrukturen von Stadt-, Überland- und Reisebussen. Der Nahverkehr 6:19–23
42. Leuthardt H. (2000) Betriebskosten von Stadtbahnen. Der Nahverkehr 10:14–17
43. Löbel A. (1997) Experiments with a Dantzig-Wolfe Decomposition for Multiple-Depot Vehicle Scheduling Problems. Preprint SC 97-16, Konrad-Zuse-Zentrum, Berlin (available at URL `http://www.zib.de/ZIBbib/Publications/`)
44. Marsten R.E., Hogan W., Blankenship J. (1975) The Boxstep Method for Large-Scale Optimization. Op. Res. 23:389–405
45. Marsten R.E., Shepardson F. (1981) Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications. Networks 11:165–177
46. Mehlhorn K., Ziegelmann M. (2000) Resource constrained shortest paths. In Proc. 8th European Symposium on Algorithms (ESA2000), Lecture Notes in Computer Science, Springer, Berlin, pp. 326–337
47. Nemhauser G., Savelsbergh M., Sigismondi G. (1994) MINTO, a Mixed INTeger Optimizer. Op. Res. Letters 15:47–58

48. Ribeiro C.C., Minoux M. (1985) A Heuristic Approach to Hard Constrained Shortest Path Problems. Disc. Applied Math. 10:125–137
49. Ribeiro C.C., Minoux M. (1986) Solving Hard Constrained Shortest Path Problems by Lagrangean Relaxation and Branch-and-Bound Algorithms. Math. of OR 53:303–316
50. Rousseau J.-M., Wren A. (1995) Bus Driver Scheduling – an Overview. In [18], pp. 173–187
51. Sanders P., Takkula T., Wedelin D. (1999) High Performance Integer Optimization for Crew Scheduling. In Proc. of the HPCS '99, Amsterdam, Lecture Notes in Cmputer Science, Springer, Berlin
52. Soumis F. (1997) Decomposition and Column Generation. In [20], Chap. 8, pp. 115–126
53. Warburton A. (1987) Approximation of Pareto Optima in Multiple-Objective Shortest Path Problems. Op. Res. 35:70–79
54. Wedelin D. (1995a) An Algorithm for Large Scale 0-1 Integer Programming with Application to Airline Crew Scheduling. Annals of OR 57:283–301
55. Wedelin D. (1995b) The Design of a 0-1 Integer Optimizer and its Application in the Carmen System. Europ. J. on OR 87:722–730
56. Wilson N. (Ed.) (1999) Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems, Springer, Berlin
57. Yu G. (1998) Operations Research in the Airline Industry. Int. Series in OR and Mgmt. Sci., Kluwer, Dordrecht