Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER ANDREAS LÖBEL STEFFEN WEIDER

A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit

This research has been supported by the German ministery for resarch and education (BMBF), grant no. 03-GRM2B4.
Responsibility for the content of this article is with the authors.

ZIB-Report 04-14 (May 2004)

A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit*

Ralf Borndörfer Andreas Löbel Steffen Weider**

May 4, 2004

Abstract

This article proposes a Lagrangean relaxation approach to solve integrated duty and vehicle scheduling problems arising in public transport. The approach is based on the proximal bundle method for the solution of concave decomposable functions, which is adapted for the approximate evaluation of the vehicle and duty scheduling components. The primal and dual information generated by the bundle method is used to guide a branch-and-bound type algorithm.

Computational results for large-scale real-world integrated vehicle and duty scheduling problems with up to 1,500 timetabled trips are reported. Compared with the results of a classical sequential approach and with reference solutions, integrated scheduling offers remarkable potentials in savings and drivers' satisfaction.

1 Introduction

The process of *operational planning* in public transit is traditionally organized in successive steps of timetabling, vehicle scheduling, duty scheduling, duty rostering, and crew assignment. These tasks are well investigated in the optimization and operations research literature. An enormous progress has been made in both the theoretical analysis of these problems and in the computational ability to solve them. For an overview see the proceedings of the last four CASPT conferences ([26], [3], [7], and [5]).

It is well known that the *integrated treatment* of planning steps discloses additional degrees of freedom that can lead to further efficiency gains. The first and probably best known approach in this direction is the so-called *sensitivity analysis*, a method on the interface between timetabling and vehicle scheduling that uses slight shiftings of trips in the timetable to improve the vehicle schedule. The method has been used with remarkable success in HOT and HASTUS, see [4] and [16].

*This research has been supported by the German ministery for resarch and education (BMBF), grant no. 03-GRM2B4. Responsibility for the content of this article is with the authors.

**Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, Email [surname]@zib.de

Vehicle and duty scheduling, the topic of this article, is another area where integration is important. The need is largest in *regional scenarios*, which often have few relief points for drivers, such that long vehicle rotations can either not be covered with legal duties at all or only at very high cost. In such scenarios the powerful optimization tools of sequential scheduling are useless. Rather, the vehicle and the duty scheduling steps must be synchronized to produce acceptable results, i.e., an *integrated vehicle and duty scheduling* method is indispensible. Urban scenarios do, of course, offer efficiency potentials as well.

The current *planning systems* provide only limited support for integrated vehicle and duty scheduling. There are frameworks for manual integrated scheduling that allow to work on vehicles and duties simultaneously, rule out infeasibilities, make suggestions for concatenations, etc. Without integrated optimization tools, however, the planner must still build vehicle schedules by hand, anticipating the effects on duty scheduling by skill and experience.

The *literature* on integrated vehicle and duty scheduling is also comparably scant. The first article on the *integrated vehicle and duty scheduling problem* (ISP) that we are aware of was published in 1983 by Ball, Bodin, and Dial [1]. They describe an ISP at the Baltimore Metropolitan Transit Authority and develop a mathematical model for it. However, they propose to solve this model by decomposing it into its vehicle and duty scheduling parts, i.e., the model is integrated, but the solution method is sequential.

For the next two decades, the predominant approach to the ISP was to include duty scheduling considerations into a vehicle scheduling method or vice versa. The first approach is, e.g., presented by [24] and [6], who propose two-step methods that first includes some duty scheduling constraints in a vehicle scheduling procedure and afterwards solve the duty scheduling problem in a second step. Examples of the opposite approach are the articles [25], [8], and [23]. They concentrate on duty scheduling and take the vehicle scheduling constraints and costs heuristically into account. A survey of integrated approaches until 1997 can be found in [15].

The complete integration of vehicle and crew scheduling was first investigated in a series of publications by Freling and coauthors ([9, 10, 11, 13, 12]). They propose a combined vehicle and duty scheduling model and attack it by integer programming methods. Computational results on several problems from the Rotterdam public transit company RET with up to 300 timetabled trips, and from Connexxion, the largest bus company in the Netherlands, with up to 653 timetabled trips are reported. A branch-and-price approach to ISP instances involving a single type of vehicles was also described by [14] and tested on artificial data.

We propose in this article an integrated vehicle and duty scheduling method similar to that of Freling et al. Our main contribution is the use of bundle techniques for the solution of the Lagrangean relaxations that come up there. The advantages of the bundle method are its high quality bounds and automatically generated primal information that can both be used to guide a branch-and-bound type algorithm. We apply this method to real-world instances from several German carriers with up to 1,500 timetabled trips. As far as we know, these are the largest and most complex instances that have been tackled in the literature using an integrated scheduling approach. Our optimization module IS-OPT has been developped in a joint research project with IVU Traffic Technologies AG (IVU), Mentz Datenverarbeitung GmbH (mdv), and the Regensburger Verkehrsbetriebe (RVB). It is incorporated in IVU's commercial scheduling system MICROBUS 2.

The article is organized as follows. Section 2 gives a formal description of the ISP and states an integer programming model that provides the basis of our approach. Section 3 describes our scheduling method. We discuss the Lagrangean relaxation that arises from a relaxation of the coupling constraints for the vehicle and the duty scheduling parts of the model, the solution of this relaxation by the proximal bundle method, in particular, the treatment of inexact evaluations of the vehicle and duty scheduling component functions, and the use of primal and dual information generated by the bundle method to guide a branch-and-bound algorithm. Section 4 reports computational results for large-scale real-world data. In particular, we apply our integrated scheduling method to half-regional half-urban instances for the German city of Regensburg with up to 1,500 timetabled trips.

2 Integrated Vehicle and Duty Scheduling

The *integrated vehicle and duty scheduling problem* (ISP) contains a vehicle and a duty scheduling part. We describe these individual parts first and conclude with the integrated scheduling problem. The exposition assumes that the reader is familar with the terminology of vehicle and duty scheduling; suitable references are [22] for vehicle scheduling and [2] for duty scheduling.

The vehicle scheduling part of the ISP is based on an acyclic directed multigraph $G = (\mathcal{T} \cup \{s, t\}, \mathcal{D})$. The nodes of G are the set $\mathcal{T}$ of *timetabled trips* plus two additional artificial nodes s and t, which represent the beginning and the end of a vehicle rotation, respectively; s is the source of G and t the sink. The arcs $\mathcal{D}$ of G are called *deadheads*, the special deadheads that emanate from the source s are the *pull-out trips*, those entering the sink t are the *pull-in trips*. Associated with each deadhead a is a *depot* $g_a \in \mathcal{G}$ from some set $\mathcal{G}$ of *depots* (i.e., vehicle types), that indicates a valid vehicle type, and a *cost* $d_a \in \mathbb{Q}$. There may be parallel arcs in G with different depots and costs. We denote by $\mathcal{D}_g := \{a \in \mathcal{D} : g_a = g\}$ the set of deadheads that can be covered by a vehicle of type $g \in \mathcal{G}$, by $\delta_g^+(v) := \delta^+(v) \cap \mathcal{D}_g$ the outcut of node v, restricted to arcs in $\mathcal{D}_g$, and by $\delta_g^-(v) := \delta^-(v) \cap \mathcal{D}_g$ the incut of node v, restricted to arcs in $\mathcal{D}_g$.

A *vehicle rotation* or *block* of type $g \in \mathcal{G}$ is an st-path in G that uses only deadheads of type g, i.e., an st-path p such that $p \subseteq \mathcal{D}_g$ for some depot $g \in \mathcal{G}$. A vehicle schedule is a set of blocks such that each timetabled trip is contained in one and only one block. The *vehicle scheduling problem* (VSP) is to find a vehicle schedule of minimal cost. It can be stated as the following integer program:

$$
\begin{array}{llll}
(\textsc{Vsp}) & \min\ d^\mathsf{T} y & & \\
\text{(i)} & y(\delta_g^+(v)) - y(\delta_g^-(v)) & = 0 & \forall v \in \mathcal{T}, g \in \mathcal{G} \\
\text{(ii)} & y(\delta^+(v)) & = 1 & \forall v \in \mathcal{T} \\
\text{(iii)} & y(\delta^-(v)) & = 1 & \forall v \in \mathcal{T} \\
\text{(iv)} & y & \in \{0,1\}^\mathcal{D} &
\end{array}
$$

The duty scheduling part of the ISP also involves an acyclic digraph $D = (\mathcal{R} \cup \{s,t\}, \mathcal{L})$. The nodes of D consist of a set of *tasks* $\mathcal{R}$ plus two artificial nodes s and t, which mark the beginning and the end of a *part of work* of a duty; again s is the source of D and t the sink. A task r can correspond either to a timetabled trip $v_r \in \mathcal{T}$ or to a deadhead trip $a_r \in \mathcal{D}$; there may also be additional tasks independent of the vehicle schedule that model sign-on and sign-off times and similar activities of drivers.

Let $\mathcal{R}_\mathcal{T}$ and $\mathcal{R}_\mathcal{D}$ be the sets of tasks that correspond to a timetabled trip and a deadhead trip, respectively. We assume that there is at least one task associated with every timetabled trip and every deadhead trip; these tasks correspond to units of driving work on such a trip. Several tasks for one trip indicate that this trip is subdivided by relief opportunities to exchange a driver into several units of driving work. The arcs $\mathcal{L}$ of D are called *links*; they correspond to feasible concatenations of tasks in a potential duty. A *part of work* of a duty is an *st*-path p in D that corresponds to the certain legality rules and has a certain cost c_p, again determined by certain rules. A *duty* is a concatenation of one or more (usually one or two) compatible parts of work.

Denote by $\mathcal{S}$ the set of all such duties, and by c_p, $p \in \mathcal{S}$, their costs. Let further $\mathcal{S}_r := \{p \in \mathcal{S} : r \in p\}$ be the set of all duties that contain some task $r \in \mathcal{R}$. Given a vehicle schedule y, a *compatible duty schedule* is a collection of duties such that each task that corresponds to either a timetabled trip or a deadhead trip from the vehicle schedule is contained in exactly one duty, while the tasks corresponding to deadhead trips that are not contained in the vehicle schedule are not contained in any duty. The *duty scheduling problem* associated with a vehicle schedule y is to find a compatible duty schedule of minimum cost. The DSP can be stated as the following integer program:

$$
\begin{array}{llll}
(\textsc{Dsp}_\mathrm{y}) & \min\ c^\mathsf{T} x & & \\
\text{(i)} & x(\mathcal{S}_r) & = 1 & \forall r \in \mathcal{R}_\mathcal{T} \\
\text{(ii)} & x(\mathcal{S}_r) & = y_{a_r} & \forall a_r \in \mathcal{D} \\
\text{(iii)} & x & \in \{0,1\}^\mathcal{S} &
\end{array}
$$

The *integrated vehicle and duty scheduling problem* is to simultaneously construct a vehicle schedule and a compatible duty schedule of minimum overall cost. Introducing suitable constraint matrices and vectors, the ISP reads:

$$\begin{array}{llrcrl}
(\text{ISP}) & \min & d^{\mathsf{T}}y & + & c^{\mathsf{T}}x & \\
(\text{i}) & & Ny & & & = b \\
(\text{ii}) & & & & Ax & = 1 \\
(\text{iii}) & & My & - & Bx & = 0 \\
(\text{iv}) & & y & & & \in \{0,1\}^{\mathcal{D}} \\
(\text{v}) & & & & x & \in \{0,1\}^{\mathcal{S}}
\end{array}$$

In this model, the *multiflow constraints* (ISP) (i) correspond to the vehicle scheduling constraints (VSP) (i)–(iii); they generate a feasible vehicle schedule. The *(timetabled) trip partitioning constraints* (ISP) (ii) are exactly the duty scheduling constraints (DSP_y) (i); they make sure that each timetabled trip is covered by exactly one duty. Finally, the *coupling constraints* (ISP) (iii) correspond to the duty scheduling constraints (DSP_y) (ii)–(iii); they guarantee that the vehicle and duty schedules x and y are synchronized on the deadhead trips, i.e., a deadhead trip is either assigned to both a vehicle and a duty or to none. We remark that a practical version includes several types of additional constraints such as depot capacities, and duty scheduling base constraints (e.g. duty type capacities, average paid/working times), which we omit in this article. The inclusion of such constraints in our scheduling method is, however, straightforward.

The integrated scheduling model (ISP) consists of a multicommodity flow model for vehicle scheduling and a set partitioning model for duty scheduling on timetabled trips. These two models are joined by a set of coupling constraints for the deadhead trips, one for each task on a deadhead trip. The model (ISP) is the same as that used by Freling and coauthors, see [9].

3 A Bundle Method

Our general solution strategy for the ISP is based on a Lagrangean relaxation of the coupling constraints (ISP) (iii). This decomposes the problem into a vehicle scheduling subproblem, a duty scheduling subproblem, and a Lagrangean master problem. All three of these problems are large scale, but of quite different nature. Efficient methods ([22]) are available to solve vehicle scheduling problems of the sizes that come up in an integrated approach with a very good quality or even to optimality. Duty scheduling is, in fact, the hardest part. We are not aware of methods that can produce high quality lower bounds for large-scale real-world instances. However, duty scheduling problems can be tackled in a practically satisfactory way using column generation algorithms, see, e.g., [2]. We will use such an algorithm to “solve” the duty scheduling subproblem. In the Lagrangean master, multipliers for several tens of thousands of coupling constraints have to be determined. Here, the complexity of the vehicle and the duty scheduling subproblems demands a method that converges quickly and that can be adapted to inexact evaluation of the subproblems. The *proximal bundle method* [21] has these properties; it further produces primal information that can be used in a superordinate branch-and-bound algorithm to guide the branching decisions. Moreover, the large dimension of the Lagrangean multiplier space, a potential computational obstacle, can be collapsed by a simple dualization.

This section discusses our Lagrangean relaxation/column generation approach to the Isp using the proximal bundle method. In a first phase, the procedure aims at the computation of an "estimation" of a global lower bound for the Isp *and* at the computation of a set of duties that is likely to contain the major parts of a good duty schedule. This procedure constitutes the core of our integrated vehicle and duty scheduling method. In a second phase, the bundle core is called repeatedly in a branch-and-bound type procedure to produce integer solutions.

3.1 Lagrangean Relaxation

We consider in this subsection a restriction (Isp_I) of the Isp to some subset of duties $I \subseteq \mathcal{S}$ that have been generated explicitly (in some way):

$$\begin{array}{llcccl}
(\text{Isp}_\text{I}) & \min & d^\mathsf{T} y & + & c_I^\mathsf{T} x^I & \\
(\text{i}) & & Ny & & & = b \\
(\text{ii}) & & & & A_I x^I & = 1 \\
(\text{iii}) & & My & - & B_I x^I & = 0 \\
(\text{iv}) & & y & & & \in \{0,1\}^\mathcal{D} \\
(\text{v}) & & & & x^I & \in \{0,1\}^I
\end{array}$$

A Lagrangean relaxation with respect to the coupling constraints (Isp_I) (iii) and a relaxation of the integrality constraints (iv) and (v) results in the Lagrangean dual

$$(\text{L}_I) \quad \max_\lambda \left[\min_{\substack{Ny=d,\\ y\in[0,1]^\mathcal{D}}} (d^\mathsf{T} - \lambda^\mathsf{T} M)y + \min_{\substack{A_I x^I = \mathbb{1},\\ x^I\in[0,1]^I}} (c_I^\mathsf{T} + \lambda^\mathsf{T} B_I)x^I \right].$$

Define functions and associated arguments by

$$\begin{aligned}
f_V \quad &: \mathbb{R}^{\mathcal{R}_\mathcal{D}} \to \mathbb{R}, \quad \lambda \mapsto \min(d^\mathsf{T} - \lambda^\mathsf{T} M)y; \quad Ny = d; \quad y \in [0,1]^\mathcal{D} \\
f_D^I \quad &: \mathbb{R}^{\mathcal{R}_\mathcal{D}} \to \mathbb{R}, \quad \lambda \mapsto \min(c^\mathsf{T} + \lambda^\mathsf{T} B_I)x^I; \quad A_I x^I = \mathbb{1}; \quad x^I \in [0,1]^I \\
f^I \quad &:= f_V + f_D^I,
\end{aligned}$$

and

$$\begin{aligned}
y(\lambda) \quad &:= \operatorname{argmin}_{y\in[0,1]^\mathcal{D}} f_V(\lambda) \\
x^I(\lambda) \quad &:= \operatorname{argmin}_{x^I\in[0,1]^I} f_D^I(\lambda),
\end{aligned}$$

breaking ties arbitrarily. With this notation, (L_I) becomes

$$(\text{L}_I) \quad \max_\lambda f^I(\lambda) = \max_\lambda \left[f_V(\lambda) + f_D^I(\lambda) \right].$$

The functions f_V and f_D^I are concave and piecewise linear. Their sum f^I is therefore a decomposable, concave, and piecewise linear function; f^I is, in particular, nonsmooth. This is precisely the setting for the proximal bundle method.

3.2 The Proximal Bundle Method

The *proximal bundle method* (PBM) is a subgradient-type procedure for concave functions. It can be adapted to handle decomposable, nonsmooth functions in a particularly efficient way.

We recall the method in this section as far as we need for our exposition. An in-depth treatment can be found in the articles [20, 21].

When applied to (L_I), the PBM produces two sequences of iterates $\lambda_i, \mu_i \in \mathbb{R}^{\mathcal{R}_\mathcal{D}}$, $i = 0, 1, \ldots$. The points μ_i are called *stability centers*; they converge to a solution of (L_I). The points λ_i are trial points; calculations at the trial points result either in a shift of the stability center, or in some improved approximation of f^I.

More precisely, the PBM computes at each iterate λ_i linear approximations

$$
\begin{aligned}
\bar{f}_V(\lambda; \lambda_i) &:= f_V(\lambda_i) + g_V(\lambda_i)^\mathsf{T}(\lambda - \lambda_i) \\
\bar{f}_D^I(\lambda; \lambda_i) &:= f_D^I(\lambda_i) + g_D^I(\lambda_i)^\mathsf{T}(\lambda - \lambda_i) \\
\bar{f}^I(\lambda; \lambda_i) &:= \bar{f}_V(\lambda; \lambda_i) + \bar{f}_D^I(\lambda; \lambda_i)
\end{aligned}
$$

of the functions f_V, f_D^I, and f^I by determining the function values $f_V(\lambda_i)$, $f_D^I(\lambda_i)$ and the subgradients $g_V(\lambda_i)$ and $g_D^I(\lambda)$; by definition, these approximations overestimate the functions f_V and f_D^I, i.e., $\bar{f}_V(\lambda; \lambda_i) \geq f_V(\lambda)$ and $\bar{f}_D^I(\lambda; \lambda_i) \geq f_D^I(\lambda)$ for all λ. Note that $\bar{f}_V$ and $\bar{f}_D^I$ are polyhedral, such the subgradients can be derived from the arguments $y(\lambda_i)$ and $x^I(\lambda_i)$ associated with the multiplier λ_i as

$$
\begin{aligned}
g_V(\lambda_i) &:= - & My(\lambda_i) \\
g_D^I(\lambda_i) &:= & B_I x^I(\lambda_i) \\
g^I(\lambda_i) &:= - & My(\lambda_i) + B_I x^I.
\end{aligned}
$$

This linearization information is collected in so-called *bundles*

$$
\begin{aligned}
J_{V,i} &:= \{(\lambda_j, f_V(\lambda_j), g_V(\lambda_j) : j = 0, \ldots, i\} \\
J_{D,i}^I &:= \{(\lambda_j, f_D^I(\lambda_j), g_D^I(\lambda_j) : j = 0, \ldots, i\}.
\end{aligned}
$$

For convenience of exposition, we will use notations such as $\lambda_j \in J_{V,i}$, $g_V(\lambda_j) \in J_{V,i}$, etc. to express that the referenced item is contained in some appropriate tuple in the bundle. The PBM uses the bundles to build piecewise linear approximations

$$\begin{aligned}
\hat{f}_{V,i}(\lambda) &:= \min_{\lambda_j \in J_{V,i}} \bar{f}_V(\lambda;\lambda_j) \\
\hat{f}^I_{D,i}(\lambda) &:= \min_{\lambda_j \in J^I_{D,i}} \bar{f}^I_D(\lambda;\lambda_j) \\
\hat{f}^I_i &:= \hat{f}_{V,i} + \hat{f}^I_{D,i}
\end{aligned}$$

of f_V, f^I_D, and f^I. Adding a quadratic term to this model that penalizes large deviations from the current stability center μ_i, the next trial point λ_{i+1} is calculated by solving the quadratic programming problem

$$(\mathrm{QP}^I_i) \qquad \lambda_{i+1} := \mathrm{argmax}_\lambda \, \hat{f}^I_i(\lambda) - \tfrac{u}{2} \left\|\mu_i - \lambda\right\|^2 .$$

Here, u is a positive weight that can be adjusted to increase accuracy or convergence speed. If the approximated function value $\hat{f}^I_i(\lambda_{i+1})$ at the new iterate λ_{i+1} is sufficiently close to the function value $f^I(\mu_i)$, the PBM stops; μ_i is the approximate solution. Otherwise a test is performed whether the predicted increase $\hat{f}^I_i(\lambda_{i+1}) - f^I(\mu_i)$ leads to sufficient real increase $f^I(\lambda_{i+1}) - f^I(\mu_i)$; in this case, the model is judged accurate and the stability center is moved to $\mu_{i+1} := \lambda_{i+1}$. The bundles are updated by adding the information computed in the current iteration, and, possibly, by dropping some old information. Then the next iteration starts, see Algorithm 3.2 for a listing (the vectors $\tilde{g}_{V,i}$ and $\tilde{g}_{D,i}$ will be defined and explained in a second).

Require: Starting point $\lambda_0 \in \mathbb{R}^n$, weights $u_0, m > 0$, optimality tolerance $\epsilon \geq 0$.

1: Initilization: $i \leftarrow 0$, $J_{V,i} \leftarrow \{\lambda_i\}$, $J_{D,i} \leftarrow \{\lambda_i\}$, and $\mu_i = \lambda_i$.
2: Direction Finding: Compute λ_{i+1}, $\tilde{g}_{V,i}$, $\tilde{g}_{D,i}$ by solving problem (QP^I_i).
3: Function evaluation: Compute $f_V(\lambda_{i+1})$, $g_V(\lambda_{i+1})$, $f^I_D(\lambda_{i+1})$, $g^I_D(\lambda_{i+1})$.
4: Stopping Criterion: If $\hat{f}^I_i(\lambda_{i+1}) - f^I(\mu_i) < \epsilon(1 + \left|f^I(\mu_i)\right|)$ output μ_i, terminate.
5: Bundle Update:
Select $J_{V,i+1} \subseteq J_{V,i} \cup \{\big(\lambda_{i+1}, f_V(\lambda_{i+1}), g_V(\lambda_{i+1})\big), \big(\lambda_{i+1}, \hat{f}_{V,i}(\lambda_{i+1}), \tilde{g}_{V,i}\big)\}$,
select $J^I_{D,i+1} \subseteq J^I_{D,i} \cup \{\big(\lambda_{i+1}, f^I_D(\lambda_{i+1}), g^I_D(\lambda_{i+1})\big), \big(\lambda_{i+1}, \hat{f}_{D,i}(\lambda_{i+1}), \tilde{g}_{D,i}\big)\}$.
6: Ascent Test: $\mu_{i+1} \leftarrow f^I(\lambda_{i+1}) - f^I(\mu_i) > m(\hat{f}^I_i(\lambda_{i+1}) - f^I(\mu_i))\ ?\ \lambda_{i+1} : \mu_i$.
7: Weight Update: Set u_{i+1}.
8: $i \leftarrow i + 1$, goto step 2.

Algorithm 1: Generic Proximal Bundle Method (PBM).

Besides function and subgradient calculations, the main work in the PBM is the solution of the quadratic problem QP^I_i. This problem can also be stated as

$$\begin{array}{llllll}
(\mathrm{QP}^{\mathrm{I}}_{\mathrm{i}}) & \max & v_V + & v^I_D & -\frac{u}{2}\left\|\mu_i - \lambda\right\|^2 & \\
(\mathrm{i}) & & v_v & & -\bar{f}_V(\lambda;\lambda_j) \leq 0 & \forall \lambda_j \in J_{V,i} \\
(\mathrm{ii}) & & & v^I_D & -\bar{f}^I_D(\lambda;\lambda_j) \leq 0 & \forall \lambda_j \in J^I_{D,i}.
\end{array}$$

A dualization results in the equivalent formulation

$$
\begin{aligned}
(\mathrm{D{\scriptstyle QP}_i^I}) \qquad \operatorname{argmax} \quad & \sum_{\lambda_j \in J_{V,i}} \alpha_{V,j} \bar{f}_V(\mu_i;\lambda) + \sum_{\lambda_j \in J^I_{D,i}} \alpha^I_{D,j} \bar{f}^I_D(\mu_i;\lambda) \\
& - \tfrac{1}{2u} \left\| \sum_{\lambda_j \in J_{V,i}} \alpha_{V,j} g_V(\lambda) + \sum_{\lambda_j \in J^I_{D,i}} \alpha^I_{D,j} g^I_D(\lambda) \right\|^2 \\
& \sum_{\lambda_j \in J_{V,i}} \alpha_{V,j} \quad = 1 \\
& \sum_{\lambda_j \in J^I_{D,i}} \alpha^I_{D,j} \quad = 1 \\
& \quad \alpha_V, \alpha^I_D \quad \geq 0.
\end{aligned}
$$

Here, $\alpha_V \in [0,1]^{J_{V,i}}$ and $\alpha^I_D \in [0,1]^{J^I_{D,i}}$ are the dual variables associated with the constraints ($\mathrm{Q{\scriptstyle P}}^I_i$) (i) and (ii), respectively. Given a solution (α_V, α^I_D) of $\mathrm{D{\scriptstyle QP}}^I_i$, the vectors

$$
\begin{aligned}
\tilde{g}_{V,i} &:= \textstyle\sum_{\lambda_j \in J_{V,i}} \alpha_{V,j} g_V(\lambda_j) \\
\tilde{g}^I_{D,i} &:= \textstyle\sum_{\lambda_j \in J^I_{D,i}} \alpha^I_{D,j} g^I_D(\lambda_j) \\
\tilde{g}^I_i &:= \tilde{g}_{V,i} + \tilde{g}^I_{D,i}
\end{aligned}
$$

are convex combinations of subgradients; they are called *aggregated subgradients* of the functions f_V, f^I_D, and f^I, respectively. It can be shown that they are, actually, subgradients of the respective functions at the point λ_{i+1} and, moreover, that this point can be calculated by means of the formula

$$
\lambda_{i+1} = \mu + \frac{1}{u} \left[\sum_{\lambda_j \in J_{V,i}} \alpha_{V,j} g_V(\lambda_j) + \sum_{\lambda_j \in J^I_{D,i}} \alpha^I_{D,j} g^I_D(\lambda_j) \right].
$$

Note that ($\mathrm{D{\scriptstyle QP}}^I_i$) is again a quadratic program, the dimension of which is equal to the size of the bundles, while its codimension is only two. In our integrated scheduling method, we solve ($\mathrm{D{\scriptstyle QP}}^I_i$) using a specialized version of the *spectral bundle method* of [17], a variant of the proximal bundle method that can take advantage of this special structure.

The PBM (without stopping) is known to have the following properties:

- The series (μ_i) converges to an optimal solution of L_I, i.e., an optimal dual solution of the LP-relaxation of ($\mathrm{I{\scriptstyle SP}}_I$).
- The series (y_i, x^I_i) defined as

  $$
  (y_i, x^I_i) := \left(\sum_{\lambda_j \in J_{V,i}} \alpha_{V,i} y(\lambda_j), \sum_{\lambda_j \in J^I_{D,i}} \alpha^I_{D,i} x^I(\lambda_j) \right)
  $$

  converges to an optimal primal solution of the LP-relaxation of ($\mathrm{I{\scriptstyle SP}}_I$).

- Convergence is preserved if the bundles contain at least two subgradients, namely, the last subgradient and the aggregated subgradient, see step 5 of Algorithm 3.2.

The bundle size controls the convergence speed of the PBM. If large bundles are used, less iterations are needed, however, problem (QP_i^I) becomes more difficult. We limit the bundle size in our implementation to 5, a value that we found a good compromise between speed and accuracy.

3.3 Adaptations of the Bundle Method

Two obstacles prevent the straightforward application of the proximal bundle method to the ISP. First, the component problem for duty scheduling is $\mathcal{NP}$-hard, even in its LP-relaxation; the vehicle scheduling LP is computationally at least not easy. We can therefore not expect that we can compute the function values $f_V(\lambda_i)$ and $f_D^I(\lambda_i)$ and the associated subgradients $g_V(\lambda_i)$ and $g_D^I(\lambda_i)$ exactly. The algorithms [22] and [2] that we use provide in general only approximate solutions. Second, the column generation algorithm process that is carried out for the duty scheduling problem must be synchronized with the bundle method.

The literature gives two versions of approximate versions of the bundle method that can deal with inexact evaluations of the component functions. Kiwiel [21] stated a version of the PBM that produces an ϵ-approximate solution, given that ϵ-subgradients can be computed, for a some arbitrary, but fixed value of ϵ. Hintermüller [18] gave an improved version for which it is not necessary to know the actual value of ϵ; his method produces solutions that are as good as the supplied ϵ-subgradients. They converge, in particular, to the optimum if the linear approximation converges to the original function.

We could use these approaches in principle in our setting, but at a high computational cost and with only limited benefit. In fact, our vehicle scheduling algorithm produces not only a primal solution, but also a lower bound and an adequate subgradient from a certain single-depot relaxation of the vehicle scheduling problem. However, the information that can be derived from the subgradients associated with this single-depot relaxation was not very helpful in our computational experiments. Concerning the duty scheduling part, we are also able to compute a lower bound and adequate subgradients for the duty scheduling component function f_D^I for any fixed column set using exact LP-techniques. However, this is a lot of effort for a bound that is not globally valid. We remark that one can, at least in principle, also compute a lower bound for the entire duty scheduling function f_D, see [2]. Such procedures are, however, extremely time consuming and do not yield high quality bounds for large-scale problems. Therefore we use a different, much faster approach to approximate the component functions themselves by piecewise linear functions. We show below how this can be done rigorously for the vehicle scheduling part; in the duty scheduling part, the procedure is heuristic, and we simply update our approximation whenever we notice an error.

Vehicle Scheduling Function f_V. Denote by $f_V^L : \mathbb{R}^{\mathcal{D}} \mapsto \mathbb{R}$ the approximation to the value of the vehicle scheduling component function $f_V(\lambda)$ as given by some vehicle scheduling algorithm, and by $y^L(\lambda) \in [0,1]^{\mathcal{D}}$ the associated argument. We have $f_V^L(\lambda) \geq f_V(\lambda)$, but f_V^L is in general not concave. However, we can use f_V^L to create a concave approximation $\hat{f}_{V,i}^L \geq f_V$ using linearizations at the points $\lambda_j \in J_{V,i}$, namely, by setting

$$
\begin{aligned}
g_{V,i}^L &:= -My^L(\lambda_i) \\
\bar{f}_V^L(\lambda;\lambda_i) &:= f_V^L(\lambda_i) + {g_{V,i}^L}^\mathsf{T}(\lambda - \lambda_i) \\
\hat{f}_{V,i}^L(\lambda) &:= \min_{\lambda_j \in J_{V,i}} \bar{f}_V^L(\lambda;\lambda_j).
\end{aligned}
$$

We use this approximation in the PBM Algorithm 3.2 by replacing f_V by $\hat{f}_{V,i}^L$. The bundle update (step 5) is implemented as

$$
J_{V,i+1} \subset \begin{cases} J_{V,i} \cup \left\{ \left(\lambda_{i+1}, f_V^L(\lambda_{i+1}), g_{V,i+1}^L\right) \right\} & \\ \qquad \cup \left\{ \left(\lambda_{i+1}, \hat{f}_{V,i}^L(\lambda_{i+1}), \tilde{g}_{V,i}^L\right) \right\}, & \text{if } f_V^L(\lambda_{i+1}) < \hat{f}_{V,i}^L(\lambda_{i+1}), \\ J_{V,i}, & \text{else.} \end{cases} \tag{1}
$$

Since the function $\hat{f}_{V,i+1}^L$ depends on $J_{V,i}$, we must also recalculate its value $\hat{f}_{V,i+1}^L(\mu_i)$ at the stability center in the stopping criterion and the ascent test (steps 4 and 6) of the PBM at each iteration.

Duty Scheduling Function f_D^I. The idea is similar as in the vehicle scheduling case. Denote by I_i the duty set that has been computed up to and in iteration i, by $f_D^{L,I_i} : \mathbb{R}^{I_i} \mapsto \mathbb{R}$ an approximate value of the duty scheduling component function $f_D^{I_i}(\lambda)$ computed by some algorithm, and by $x^{L,I_i}(\lambda) \in [0,1]^{I_i}$ the associated argument. In our integrated scheduling method, we compute this approximate primal LP solution again with a bundle method.

Again, we have $f_D^{L,I_i}(\lambda) \geq f_D^{I_i}(\lambda)$, and f_D^{L,I_i} is in general not concave. Similar, but this time heuristically, we use f_D^{L,I_i} to create a concave approximation $\hat{f}_{D,i}^{L,I_i}$ of $f_D^{I_i}$ using linearizations at the points $\lambda_j \in J_{D,i}^{I_i}$, namely,

$$
\begin{aligned}
g_{D,i}^{L,I_i} &:= B_{I_i} x^{L,I_i}(\lambda_i) \\
\bar{f}_D^{L,I_i}(\lambda;\lambda_i) &:= f_D^{L,I_i}(\lambda_i) + {g_{D,i}^{L,I_i}}^\mathsf{T}(\lambda - \lambda_i) \\
\hat{f}_{D,i}^{L,I_i}(\lambda) &:= \min_{\lambda_j \in J_{D,i}^{I_i}} \bar{f}_D^{L,I_i}(\lambda;\lambda_j).
\end{aligned}
$$

Since each linearization is computed with respect to a subset of duties I_j, it is in general not true that $\bar{f}_D^{L,I_j} \geq f_D^{I_i}$ if $I_i \neq I_j$. It can (and does) therefore happen that we notice that the current iterate is cut off by some previously computed linearization, i.e.,

$$
f_D^{L,I_i}(\lambda_{i+1}) > \bar{f}_D^{L,I_j}(\lambda_{i+1};\lambda_j)
$$

for some $j \leq i$. In this case, we have detected an error made in a previous iteration and simply remove the faulty elements from the bundle and also from the approximation. The duty scheduling bundle update in step 5 of Algorithm 3.2 is implemented as

$$J_{D,i+i}^{I_{i+1}} \subset \begin{cases} \{(\lambda_j, z_j, h_j\} \in J_{D,i}^{I_i} : f_D^{L,I_{i+1}}(\lambda_{i+1}) \le z_j + h_j^\mathsf{T}(\lambda_{i+1} - \lambda_j)\} \\ \cup\big\{\big(\lambda_{i+1}, f_D^{L,I_{i+1}}(\lambda_{i+1}), g_D^{L,I_{i+1}}(\lambda_{i+1})\big)\big\} \\ \cup\big\{\big(\lambda_{i+1}, \hat{f}_D^{L,I_{i+1}}(\lambda_{i+1}), \tilde{g}_D^{L,I_{i+1}}(\lambda_{i+1})\big)\big\}, & \text{if } f_D^{L,I_i}(\lambda_{i+1}) \\ & < \hat{f}_{D,i}^{L,I_i}(\lambda_{i+1}), \\ J_{D,i}^{I_i}, & \text{else.} \end{cases} \tag{2}$$

This approximation must also be recomputed at the stability center in every iteration.

Combined Function f^I. The combined approximate functions are

$$\begin{aligned} f^{L,I_i} &:= f_V^L + f_D^{L,I_i} \\ \hat{f}_i^{L,I_i} &:= \hat{f}_{V,i}^L + \hat{f}_{D,i}^{L,I_i}. \end{aligned}$$

Require: Starting point $\lambda_0 \in \mathbb{R}^n$, duty set I_0, weights $u_0, m > 0$, optimality tolerance $\epsilon \ge 0$.

1: Initialization: $i \leftarrow 0$, $J_{V,i} \leftarrow \{\lambda_i\}$, $J_{D,i} \leftarrow \{\lambda_i\}$, and $\mu_i = \lambda_i$.
2: Direction Finding: Compute λ_{i+1}, $\tilde{g}_{V,i}^L$, $\tilde{g}_{D,i}^{L,I_i}$ by solving problem ($\mathrm{QP}_i^{I_i}$).
3: Function evaluation: Compute $f_V^L(\lambda_{i+1})$, $g_V^L(\lambda_{i+1})$, I_i, $f_D^{L,I_i}(\lambda_{i+1})$, $g_D^{L,I_i}(\lambda_{i+1})$.
4: Stopping Criterion: If $\hat{f}_i^{L,I_i}(\lambda_{i+1}) - f^{L,I_i}(\mu_i) < \epsilon(1 + |f^{L,I_i}(\mu_i)|)$ output μ_i, terminate.
5: Bundle Update: Select $J_{V,i+1}$, $J_{D,i+1}^{I_{i+1}}$ as stated in 1, 2.
6: Ascent Test: $\mu_{i+1} \leftarrow f^{L,I_i}(\lambda_{i+1}) - f^{L,I_i}(\mu_i) > m(\hat{f}_i^{L,I_i}(\lambda_{i+1}) - f^{L,I_i}(\mu_i))\ ?\ \lambda_{i+1} : \mu_i$.
7: Weight Update: Set u_{i+1}.
8: $i \leftarrow i + 1$, goto step 2.

Algorithm 2: Inexact Proximal Bundle Method (PBM) with Column Generation.

Column generation. This is the most time consuming part of our algorithm, and we therefore enter this phase only if significant progress can be expected. Our strategy is basically to recompute the duty set when the stability center changes; we call such an iteration a *serious step*, all other iterations are called *null steps*.

The reasoning behind this strategy is as follows. The quadratic penalty term in the quadratic program $\mathrm{QP}_i^{I_i}$ ensures that the next trial value for the dual multipliers λ_{i+1} stays in the vicinity of the current stability center. When the multipliers change only little, one has reason to believe that the number and the potential effect of improving duties is also small. We therefore hope that the current duty set I_i, which has been updated when the stability center was set, does still provide a good representation of the duty space also for the new multipliers λ_{i+1}. In practice, we reduce the number of column generation phases even further by requiring a certain minimum increase ε in the objective function at the new stability center; the larger ε, the less column generation phases will occur.

Algorithm 3.3 gives a listing of our bundle algorithm using inexact evaluations of the component functions and column generation in the duty scheduling component.

3.4 Backtracking Procedure

The inexact proximal bundle method that we have described in this section is embedded in a backtracking procedure that aims at the generation of integer solutions. This procedure makes use of the primal information produced by the bundle method, namely, the sequence $\Big(y_i(\lambda_i), x_i^{I_i}(\lambda_i)\Big)$. As in an LP-approach, fractional values can be interpreted as probabilites for the inclusion/exclusion of a deadhead trip or duty in an optimal integer solution.

Our computational experiments revealed that it is adavantageous to fix the deadhead trips first, until the vehicle scheduling part of the problem is decided. The remaining duty scheduling problem can then be solved with the duty scheduling module of the algorithm as described in [2]. Our strategy for fixing the deadhead variables is to fix the deadheads in the order of largest y-values. Our algorithm also examines the consequences of such fixings and, if the increase in the objective function is too large, also reverses decisions. The details on how many variables to fix at a time, up to which threshold, etc. have been determined experimentally; in general, the algorithm fixes more boldy in the beginning and more carefully towards the end.

Figure 1 shows a typical runtime chart of our algorithm IS-OPT. The x-axis measures time in seconds, the y-axis gives statistics in two different scales, namely, for the right scale, the number of duties generated (#columns), the number of deadheads fixed to one (#fixed deadheads), and the residuum of the coupling constraints (more precisely: the norm is the square of the euclidean norm of $\tilde{g}_i^{I_i}$), as well as, for the left scale, the vehicle, duty, and the integrated scheduling objective values. Here the duty scheduling value is the lower bound of the restricted DSP calculated by the PBM, and integrated scheduling objective value is simply the sum of the VSP and the DSP value.

In the first phase of the algorithm until point A a starting set of columns was generated with Lagrangean multiplicators λ all at zero. In principle the DSP objective value should be strictly decreasing here, while the number of columns should grow. However, we calculated in this initial phase only rough lower bounds for the restricted DSP, which may be more or less accurate. Additionally we deleted columns with large reduced cost if the total number of columns exceeded 450,000. Between points A and B, a series of null steps was performed, which resulted in a decreased norm and an increased ISP-value. Be-

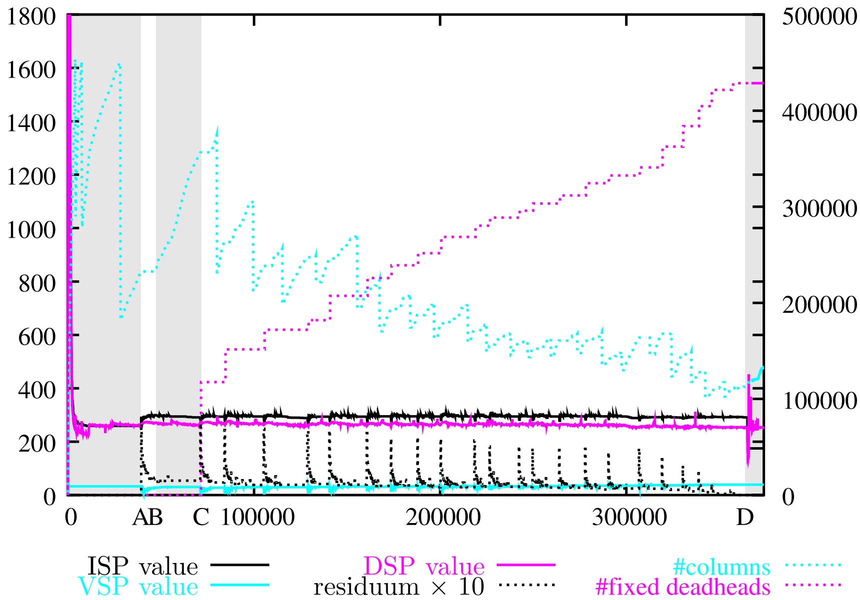

Figure 1: IS-OPT Runtime Chart.

tween points B and C, column generation phases alternated with PBM-steps, until an aggregated subgradient of small norm and thus also a "good" primal approximation of the LP-relaxation of ISP was calculated. Since the column generation process did not find enough improving columns at this point, we used the computed information to fix deadheads until (at point D) the vehicle scheduling part of the problem was completely decided. At that point, the duty scheduling component of the algorithm concluded by computing a feasible duty scheduling.

Serious steps of the PBM are marked by peaks of the norm statistic. This effect is due to the shift of the stability center in combination with the possible inclusion of additional columns in I_i. In fact, the new stability center may lie in a region where the model $\hat{f}^{L,I_i}$ of the previous iteration i is less accurate; also, new columns in I_i change the function f^{L,I_i}, what also worsens the model.

In our computational tests the algorithm rarely had to reverse a fixing decision for a deadhead and backtrack. In all our instances, the ISP objective value is very stable with respect to careful fixings of deadheads, see also Figure 1. In fact, the gap between our estimated lower bound, i.e., the objective value prior to the first fixings, and the final objective value was never larger then 5% and only 1-2% on the average. We do, however, not know the size of the gap between the estimated lower bound and the real minimum of (ISP); the mentioned behaviour is therefore only a weak indicator for the quality of the final solution found by IS-OPT.

4 Computational Results

We report in this section on the results of computational studies with our integrated vehicle and duty scheduling optimizer IS-OPT for several medium- and large-scale real-world scenarios as well as for benchmark scenarios from the literature. Our code IS-OPT is implemented in C and has been compiled using `gcc` version 3.3.3 with switches `-O4`. All computations were made single-threaded on a Dell Precision 650 PC with 4 GB of main memory and a dual Intel Xeon 3.0 GHz CPU running SuSE Linux 9.0. The computation times in the following tables are in hours:minutes.

We compare our integrated scheduling method *is* with two sequential approaches. The first one, denoted by *v+d*, is a classical sequential vehicles-first duties-second approach, i.e., *v+d* first solves the vehicle scheduling part of the problem using our optimizer VS-OPT ([22]), fixes the deadheads chosen by the vehicle schedule, and solves the resulting duty scheduling problem in a second step using our optimizer DS-OPT ([2]). The second method *d+v* uses kind of the contrary approach. A simplified integrated scheduling problem is set up that identifies drivers and vehicles, i.e., vehicle changes outside of the depot are fobidden. This "poor man's integrated scheduling model" is solved using the duty scheduling algorithm DS-OPT. The vehicle rotations resulting from this duty schedule are concatentated into daily blocks using the vehicle scheduling algorithm VS-OPT in a second step.

4.1 RVB Instances

The Regensburger Verkehrsbetriebe GmbH (RVB) is a medium sized public transportation company in Germany. We consider two instances that contain the entire RVB operation for a sunday and for a workday. The structure of the RVB data is half-regional and half-urban with only four relief points. In fact, the network of the RVB is star-shaped with nearly all lines meeting in a small area around the main railway station. Only there or at the nearby single garage the drivers can change busses and begin or end duties. The RVB uses only one type of vehicles on sundays, and three types on workdays, i.e., the sunday scenario is fleet homogenous, while the workday scenario is a multi-depot problem. The vehicle types can only be used on trips on certain sets of (non-disjoint) lines. The sunday scenario involves three different types of early, mid, and late duties, each with four different types of break rules, namely, block breaks of 1×30, 2×20, and 3×15 minutes plus 1/6-quotient breaks. The workday scenario contains in addition a type of split duties, again with the mentioned break rules per part of work. Table 4.1 reports further statistics on the number of timetabled trips, tasks, and deadhead trips. The sunday scenario is medium-sized, while the workday scenario is, as far as we known, the largest and most complex instance that has been attacked with integrated schedudling techniques.

Table 4.1 gives computational results for the sunday scenario. The column 'reference' lists statistics for the solution that RVB planners had generated by

	sunday	workday
vehicle types	1	3
timetabled trips	794	1414
tasks on tt	1248	3666
deadhead trips	47523	57646
duty types	3	4
break rules	4	4

Table 1: Statistics on the RVB Instances.

	reference	*v+d* 1	*v+d* 2	*is* 1	*is* 2
time on vehicles	518:33	472:12	472:12	501:42	512:55
paid time	545:25	562:58	565:28	518:03	531:31
paid break time	112:36	131:40	85:41	74:17	64:27
number of duties	82	83	74(1)	76	66
number of vehicles	36	32	32	32	35
average duty duration	6:39	6:48	7:38	6:40	8:03
computation time	—	0:33	5:13	35:44	37:26

Table 2: Results for the RVB Sunday Scenario.

hand. The next four columns give the results of two sequential *v+d*-optmizations and two integrated *is*-optimizations; we do not report results for the method *d+v*, because we could not produce a feasible solution for this scenario with this method. In the optimization runs "*v+d* 2" and "*is* 2", emphasis was placed on the minimization of the number of duties, while runs "*v+d* 1" and "*is* 1" tried to reproduce the average duty time of the reference solution.

As expected the sequential methods reduce the number of vehicles and the time on vehicle rotations since these are the primary optimization objectives. Also they produce quite reasonable results in terms of duty scheduling. "*v+d* 1" suffers from a slight increase in duties and paid time, "*v+d* 2" yields substantial savings in duties; however the price for this reduction is a raised average paid time. Even better are the results of the integrated opimizations. "*is* 1" is perfect with respect to any statistic and produces *large* savings. These stem from the use of short duties involving less than 4:30 hours of driving time, which don't need a break; this potential improvement of the sunday schedule is one of the most significant results of this optimization project for the RVB. Even more interesting is solution "*is* 2". This solution trades three vehicles and an increased average for another 10 duties; as longer duties must have breaks, the paid time (breaks are paid here) increases as well. Solution "*is* 2" revived a discussion at the RVB whether drivers prefer to have less, but longer duties on weekends or whether they want to stay with more, but short duties.

Table 4.1 lists the results of the workday optimizations. Method *d+v* could again not produce a feasible solution and is therefore omitted from the table. The objective in this scenario is far from obvious; it is given as a complicated

	reference	$v+d$	is
time on vehicles	1037:18	960:29	1004:27
paid time	1103:48	1032:20	1040:11
granted break time	211:53	109:11	105:23
number of duties	140	137	137
number of vehicles	91	80	82
number of pieces of work	217	290	217
number of split duties	29	39	36
average duty duration	7:56	8:03	7:55
obj. value	—	302.32	291.16
computation time	—	8:02	125:55

Table 3: Results for he RVB Workday Scenario.

mix of fixed and variable vehicle costs, fixed costs and paid time for duties, and various penalties for several pieces of work, split duties, etc., that can compensate each other such that one cannot really compare the solutions by means of single statistics. Doing it nevertheless, we see that both optimizaton approaches clearly improve the reference solution substantially. The outcome is close. In fact, $v+d$ has less paid time than *is*; in the end, however, *is* is better in terms of the composite objective function.

4.2 RKH Instances

The Regionalverkehrsbetrieb Kurhessen (RKH) is a regional carrier in the middle of Germany. They provided data for the subnetworks of Marburg and Fulda which is not (yet) in industrial use; some deadheads are missing, while for some others travel times have only been estimated by means of distance calculations. In our opinion the data still captures to a large degree the structure of a regional carrier and we therefore deem it worthwhile to report the results of the conceptual study that we did with it.

Figure 2 shows the spatial structure of the line network of Fulda, which is one part of the RKH service area. The black arcs denote the timetabled trips (drawn straight from the line's start to the end), the gray arcs indicate the potential deadhead trips. It can be seen that the trip network is hub-and-spoke-like, connecting several cities and villages among themselves and with the rural regions around them. While the deadhead network is almost complete, there are only few relief opportunities for drivers to leave or enter a vehicle.

Table 4.2 gives further statistics on the RKH instances. They are simliar to the RVB sunday scenario in terms of timetabled trips and tasks, but contain much more deadhead trips. The scenarios involve three duty types, two types of split duties that differ in the maximum duty length and one type of continous duties. Each duty type can have 1×30, 2×20, or 3×15 minutes block breaks or 1/6-quotient breaks.

Table 4.2 reports the results of our optimizations. We do not report results

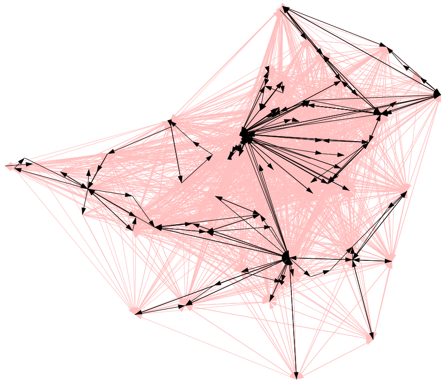

Figure 2: The graph of scenario *Fulda*

	Marburg	Fulda
depots	3	1
vehicle types	5	1
timetabled trips	634	413
tasks on TT	1022	705
deadhead trips	142,668	67,287

Table 4: RKH Instances for the Cities of Marburg and Fulda

for the method *v+d* as we were not able to produce a feasible solution for either scenario with this method. Method *d+v* yields useful results, but it is not able to cover all tasks/trips of the Fulda-scenario with duties and vehicles; in fact, *d+v* left 3 tasks and 6 timetabled trips uncovered (numbers in parentheses). These deficiencies are resolved in the *is*-solutions, which also look better in terms of numbers of vehicles.

4.3 ECOPT Instances

Finally, we compare IS-OPT with the approach of Freling et. al. on the randomly generated benchmark data proposed in their article [19]. This data consists of two sets of instances involving two and four depots, respectively. Each set contains 10 instances of 80, 100, 160, 200, 320, and 400 trips, see again [19] for a detailed description. The duty scheduling rules associated with these examples are relatively simple. Duties are allowed to have at most one break, which must be outside of a vehicle, i.e., each break also begins a new piece of work. The only other rule is that each piece of work must be of certain minimum and maximum length. It is shown in [19] that in this situation one can solve the duty generation subproblem in polynomial time, i.e., exact column generation is possible.

Tables 4.3 and 4.3 report average solution values for each of the 10 instances

	Marburg		Fulda	
	d+v	*is*	*d+v*	*is*
time on vehicles	772:02	642:41	365:41	387:37
paid time	620:27	606:30	390:08	374:53
granted break time	120:51	103:27	88:13	57:44
number of duties	73	70	41(3)	41
number of vehicles	62	50	45(6)	37
average duty duration	10:35	10:18	10:59	11:18
computation time	5:29	17:18	1:42	7:05

Table 5: Solutions on Marburg and Fulda

trips	080	10 0	160	200	320	400
vehicles	9.4	11.2	15.0	18.6	27.0	33.3
duties	21.2	25.1	33.9	40.6	57.7	69.8
total	30.6	36.3	48.9	59.2	84.7	103.1
reference	29.8	35.6	48.3	59.1	86.8	106.1
time	00:05	00:08	00:17	00:31	01:58	03:19

Table 6: Results for ECOPT-Instances with 2 Depots Variant A

of each problem class for the problem variant A; similar results for variant B have been omitted. All computations were done with the same set of parameters, which was optimized for speed. Row *reference* gives the sum of the numbers of vehicles and duties as published in [19]; for the problems with 4 depots and 320 and 400 trips, no reference is given due to excessive computation time.

It can be seen that our algorithm IS-OPT performs worse than that in [19] for the small instances, but produces better results with increasing problem size and complexity; it can also solve the largest problem instances. We remark that IS-OPT can also produce slighly better solutions for the small instances than those reported in [19] by changing the optimality parameter ϵ in Algorithm 3.3 and by raising the threshold for deadhead fixes. This leads, of course, to longer computation times.

trips	080	100	160	200	320	400
vehicles	9.2	11.2	15.0	18.5	26.7	33.1
duties	20.4	24.5	32.7	40.5	56.1	68.9
total	29.6	35.7	47.7	59.0	82.8	102.0
reference	29.6	36.2	49.5	60.4	—	—
time	00:13	00:21	00:44	01:46	05:28	12:00

Table 7: Results for ECOPT-Instances with 4 Depots Variant A

5 Conclusions

We have shown that it is possible to tackle large-scale, complex, real-world integrated vehicle and duty scheduling problems using a novel "bundle" algorithm for integrated vehicle and duty scheduling. The solutions produced by such an integrated approach can be decidedly better *in several respects at once* than the results of various types of sequential planning.

References

[1] M. O. Ball, L. Bodin, and R. Dial, *A matching based heuristic for scheduling mass transit crews and vehicles*, Transportation Science **17** (1983), 4–31.

[2] R. Borndörfer, M. Grötschel, and A. Löbel, *Duty scheduling in public transit*, MATHEMATICS — Key Technology for the Future (W. Jäger and H.-J. Krebs, eds.), Springer Verlag, Berlin, 2003, pp. 653–674 (English).

[3] J. R. Daduna, I. Branco, and J. M. P. Paixão (eds.), *Computer-aided transit scheduling*, Lecture Notes in Economics and Mathematical Systems, Springer Verlag, 1995.

[4] J. R. Daduna and M. Völker, *Fahrzeugumlaufbildung im öpnv mit unscharfen abfahrtszeiten*, Der Nahverkehr **11** (1997), 39–43.

[5] J. R. Daduna and A. Wren (eds.), *Computer-aided transit scheduling*, Lecture Notes in Economics and Mathematical Systems, Springer Verlag, 1988.

[6] K. Darby-Dowman, R. L. Lewis J. K. Jachnik, and G. Mitra, *Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience.*, Computer-Aided Transit Scheduling (Joachim R. Daduna and Anthony Wren, eds.), Springer Verlag, Berlin, 1988, pp. 226–239.

[7] M. Desrochers and J.-M. Rousseau (eds.), *Computer-aided transit scheduling*, Lecture Notes in Economics and Mathematical Systems, Springer Verlag, 1992.

[8] J. C. Falkner and D. M. Ryan, *Express: Set partitioning for bus crew scheduling in Christchurch*, Computer-Aided Transit Scheduling (M. Desrochers and J. M. Rousseau, eds.), Springer Verlag, Berlin, 1992, pp. 359–378.

[9] R. Freling, *Models and techniques for integrating vehicle and crew scheduling*, Ph.D. thesis, Erasmus University Rotterdam, Amsterdam, 1997.

[10] R. Freling, D. Huisman, and A. P. M. Wagelmans, *Models and algorithms for integration of vehicle and crew scheduling*, Tech. Report EI2000-10/A, Econometric Institute, Erasmus University Rotterdam, 2000.

[11] R. Freling, D. Huisman, and A. P. M. Wagelmans, *Applying an integrated approach to vehicle and crew scheduling in practice*, Computer-Aided Scheduling of Public Transport (Vo and J. R. Daduna, eds.), Lecture Notes in Economics and Mathematical Systems, no. 505, Springer Verlag, Berlin, 2001, pp. 73–90.

[12] ———, *Models and algorithms for integration of vehicle and crew scheduling*, Journal of Scheduling **6** (2003), 63–85.

[13] R. Freling, A. P. M. Wagelmans, and J. M. P. Paixao, *Models and algorithms for single-depot vehicle scheduling*, Transportation Science **35** (2001), 165–180.

[14] C. Friberg and K. Haase, *An exact algorithm for the vehicle and crew scheduling problem*, Computer-Aided Transit Scheduling (N. H. M. Wilson, ed.), Springer Verlag, Berlin, 1997, pp. 63–80.

[15] A. Gaffi and M. Nonato, *An integrated approach to extra-urban crew and vehicle scheduling*, Computer-Aided Transit Scheduling (N. H. M. Wilson, ed.), Springer Verlag, Berlin, 1997, pp. 103–128.

[16] Jürgen Hanisch, *Die Regionalverkehr Köln GmbH und HASTUS*, `http://www.giro.ca/Deutsch/Publications/publications.htm`, 1999.

[17] C. Helmberg, *Semidefinite programming for combinatorial optimization*, Tech. report, Zuse Institute Berlin, October 2000, also habilitation thesis.

[18] M. Hintermüller, *A proximal bundle method based on approximate subgradients*, Computational Optimization and Applications (2001), no. 20, 245–266.

[19] D. Huisman, R. Freling, and A. P. M. Wagelmans, *Multiple-depot integrated vehicle and crew scheduling*, Tech. report, Econometric Institute, Erasmus University Rotterdam, 2003.

[20] K. C. Kiwiel, *Proximal bundle methods*, Mathematical Programming **46** (1990), no. 123, 105–122.

[21] ———, *Approximation in proximal bundle methods and decomposition of convex programs*, Journal of Optimization Theory and applications **84** (1995), no. 3, 529–548.

[22] A. Löbel, *Optimal vehicle scheduling in public transit*, Ph.D. thesis, TU Berlin, 1997.

[23] I. Patrikalakis and D. Xerocostas, *A new decomposition scheme of the urban public transport scheduling problem*, Computer-Aided Transit Scheduling (M. Desrochers and J. M. Rousseau, eds.), Springer Verlag, Berlin, 1992, pp. 407–425.

[24] D. Scott, *A large scale linear programming approach to the public transport scheduling and costing problem*, Computer Scheduling of Public Transport 2 (J.-M. Rousseau, ed.), Elsevier, 1985.

[25] E. Tosini and C. Vercellis, *An interactive system for extra-urban vehicle and crew scheduling problems*, Computer-Aided Transit Scheduling (J. R. Daduna and A. Wren, eds.), Springer Verlag, Berlin, 1988, pp. 41–53.

[26] N. H. M. Wilson (ed.), *Computer-aided transit scheduling*, Lecture Notes in Economics and Mathematical Systems, Springer Verlag, 1999.