Integration of Vehicle and Duty Scheduling in Public Transport

vorgelegt von
Dipl.-Math.-oec. Steffen Weider
aus Berlin

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Berichter:	Prof. Dr. Dr. h.c. Martin Grötschel
	Prof. Dr. Jörg Rambau
Vorsitzender:	Prof. Dr. Volker Mehrmann

Tag der wissenschaftlichen Aussprache: 31.5.2007

Berlin 2007
D 83

Zusammenfassung

Diese Arbeit beschreibt den Algorithmus IS-OPT, welcher der erste Algorithmus ist, der integrierte Umlauf- und Dienstplanungsprobleme für mittelgroße Verkehrsunternehmen löst und dabei alle betrieblichen Einzelheiten berücksichtigt. Seine Lösungen können daher direkt in den täglichen Betrieb übernommen werden.

Im ersten Kapitel werden mathematische Modelle für verschiedenen Probleme aus dem Planungsprozess von Nahverkehrsunternehmen beschrieben. Es werden Ansätze zur Integration der einzelnen Probleme untersucht. In diesem Kapitel werden außerdem das Umlauf- und das Dienstplanungsproblem eingeführt. Kapitel 2 motiviert, warum Integration von Umlauf- und Dienstplanung hilfreich ist oder in welchen Fällen sie sogar unabdingbar ist; es gibt einen Überblick über die vorhanden Literatur zur integrierten Umlauf- und Dienstplanung und umreißt unseren Algorithmus IS-OPT.

Die weiteren Kapitel behandeln die in IS-OPT verwendeten Methoden: In Kapitel 3 beschreiben wir, wie Spaltenerzeugung für lineare Programme mit Lagrange-Relaxierung und Subgradienten-Verfahren kombiniert werden kann. In Kapitel 4 wird unsere Variante der proximalen Bündelmethode beschrieben. Diese wird benutzt um näherungsweise primale und duale Lösungen von lineare Programmen zu berechnen. Unsere Variante ist angepasst, um auch mit ungenauer Funktionsauswertung und ϵ-Subgradienten arbeiten zu können. Wir zeigen die Konvergenz dieser Variante unter bestimmten Annahmen. Kapitel 5 behandelt das Erzeugen von Diensten für das Dienstplanungsproblem. Dieses Problem ist als ein Kürzeste-Wege-Problem mit nichtlinearen Nebenbedingungen und fast linearer Zielfunktion modelliert. Wir lösen es, indem zuerst Schranken für die reduzierten Kosten von Diensten, die bestimmte Knoten benutzen, berechnet werden. Diese Schranken werden benutzt, um in einem Tiefensuchalgorithmus den Suchbaum klein zu halten. Im Kapitel 6 präsentieren wir die neu entwickelte Heuristik "Rapid Branching", die eine ganzzahlige Lösung des integrierten Problems berechnet. Rapid Branching kann als eine spezielle Branch-and-Bound-Heuristik gesehen werden, welche die Bündelmethode benutzt. In den Knoten des Suchbaums können mehrere Variablen auf einmal fixiert werden, die mit Hilfe einer Perturbationsheuristik ausgewählt werden.

In Kapitel 7 schließlich zeigen wir, daß wir mit IS-OPT auch große Probleminstanzen aus der Praxis lösen können und dabei bis zu 5% der Fahrzeug- und Dienstkosten sparen können.

Abstract

This thesis describes the algorithm IS-OPT that integrates scheduling of vehicles and duties in public bus transit. IS-OPT is the first algorithm which solves integrated vehicle and duty scheduling problems arising in medium sized carriers such that its solutions can be used in daily operations without further adaptions.

This thesis is structured as follows: The first chapter highlights mathematical models of the planning process of public transit companies and examines their potential for integrating them with other planning steps. It also introduces descriptions of the vehicle and the duty scheduling problem. Chapter 2 motivates why it can be useful to integrate vehicle and duty scheduling, explains approaches of the literature, and gives an outline of our algorithm IS-OPT.

The following chapters go into the details of the most important techniques and methods of IS-OPT: In Chapter 3 we describe how we use Lagrangean relaxation in a column generation framework. Next, in Chapter 4, we describe a variant of the proximal bundle method (PBM) that is used to approximate linear programs occurring in the solution process. We introduce here a new variant of the PBM which is able to utilize inexact function evaluation and the use of ϵ-subgradients. We also show the convergence of this method under certain assumptions. Chapter 5 treats the generation of duties for the duty scheduling problem. This problem is modeled as a resource-constraint-shortest-path-problem with non-linear side constraints and nearly linear objective function. It is solved in a two-stage approach. At first we calculate lower bounds on the reduced costs of duties using certain nodes by a new inexact label-setting algorithm. Then we use these bounds to speed up a depth-first-search algorithm that finds feasible duties. In Chapter 6 we present the primal heuristic of IS-OPT that solves the integrated problem to integrality. We introduce a new branch-and-bound based heuristic which we call rapid branching. Rapid branching uses the proximal bundle method to compute lower bounds, it introduces a heuristic node selection scheme, and it utilizes a new branching rule that fixes sets of many variables at once.

The common approach to solve the problems occurring in IS-OPT is to trade inexactness of the solutions for speed of the algorithms. This enables, as we show in Chapter 7, to solve large real world integrated problems by IS-OPT. The scheduled produced by IS-OPT save up to 5% of the vehicle and duty cost of existing schedules of regional and urban public transport companies.

Preface

Public mass transit in Germany is in a phase of change. The competition between the companies is becoming more intense due to the policy of the European Union to publicly tender public transit subsidies to reduce the number of local monopolies of public transit carriers. In fact, these tenderings already led to significant reductions of subsidies[1].

This increase in competition gives incentives to the companies to improve the efficiency of the allocation of their resources, i.e., personnel and vehicles, which together cause about 66-75% of the cost of a typical German public transport bus company (see Leuthardt [1998]). In this work we propose the algorithm IS-OPT that plans these resources together in a single step. We will show that IS-OPT increases the overall planning efficiency of real world planning problems in comparison to manual solutions and also in comparison to sequential optimization of vehicle and duty schedules.

The development of IS-OPT was challenging due to the large number of methods it utilizes: It combines methods from linear, convex, and combinatorial optimization, such as column generation, Lagrangean relaxation, bundle methods, shortest path algorithms, a network simplex method, and branch-and-bound approaches. Our contributions to these methods, in particular with the aim to accelerate them at the cost of getting only approximate solutions, and the way how we use and combine them to solve the integrated vehicle and duty scheduling problem is the content of this work:

- We will show how we replace exact LP-methods by an inexact proximal bundle method in a column generation context. To do this we had to solve the problems of obtaining a useful dual solution for the pricing problem from a Lagrangean relaxation and the handling of approximate evaluation of the Lagrangean dual with unknown approximation quality in the bundle method. We prove also the convergence of this new variant of the bundle method that is able to cope with this inexactness.

- We present a new active set approach to approximate LP-relaxations of large-scale set-partitioning-problems with the bundle method.

- We propose a two-phase algorithm to solve the pricing problem which is modeled by a resource-constraint-shortest-path-problem with additional constraints and a non-linear objective function. In the first phase,

[1] In the empirical study of Resch et al. [2006] three German public transit companies are mentioned whose subsidies were reduced by 32–44% in the last ten years.

we calculate bounds on this problem by Lagrangean relaxation and a labeling-algorithm. To calculate these bounds we develop an inexact labeling algorithm to approximate resource-constrained-shortest-path-problems. In the second phase, we use these bounds to prune the search tree of a depth-first-search approach.

- We introduce a primal heuristic to find an integral solution of the ISP. This heuristic can be seen as an inexact branch-and-bound algorithm that uses a branching rule which selects sets including many variables to be fixed all at once.

Another highlight of this work is the practical application of the algorithm IS-OPT. The transfer from an algorithm that works for some test instances to an algorithm that is in daily use by planners was not always easy: The industrial requirements for planning tools are going beyond the requirements for typical academic software. IS-OPT has to be stable using any reasonable data, it has to be flexible because no two public transit companies have the same duty rules, it has to be fast because often companies have to compute dozens of different scenarios for different days of operation in a limited amount of time. At last, it is not easy to find out about all side constraints and rules which feasible schedules have to fulfill in practice because often some of them are only known by the planners in charge and were never set out in writing. It was common that after presenting a solution, the planners gave us additional rules and side constraints that have to be considered.

However, it is very satisfying when finally the practitioners of a planning department admit to be amazed because not only our code is able to find solutions that are usable in their daily operations without adapting them manually, but that these solutions are also significantly better than existing schedules.

Acknowledgments

Most of the research for this thesis was done in a project of the program "math & industry" of the German ministry for research and education (BMBF)[2]. It was carried out at the Zuse Institute Berlin in cooperation with two companies that specialize on software for public transit, namely IVU traffic technologies AG (IVU) and Mentz Datenverarbeitung GmbH, and a carrier for public transit, Regensburger Verkehrsbetriebe. I want to thank all project partners for providing the test data, a deep knowledge about public transportation, and for many fruitful discussions.

After the successful completion of this research project I continued to work as one of the associates of Löbel, Borndörfer und Weider GbR closely together with IVU and DB Stadtverkehr GmbH to improve the algorithm IS-OPT, such that it can be used in the daily operations of regional bus operators of DB Stadtverkehr.

I want to thank my colleagues and friends Andreas Löbel and Ralf Borndörfer for sharing their great knowledge about mathematical optimization methods and implementation of large-scale optimization software. I want to thank Rainer Kuschel of the Regensburger Verkehrsbetriebe because without him the BMBF project would not have been the success it was. I also want to thank Martin Müller-Elschner, Oliver Grzegorski, Andreas Langenhan, Bernd Böhringer, and all the others at the IVU Traffic Technolgies AG who provided the link to the real world after finishing the BMBF-research-project and helped to implement IS-OPT at various public transit companies.

Further, I want to thank Martin Grötschel to give me the possibility to work at the Zuse Institute, which is a really great place for applied mathematics (and mathematicians). I want to thank Marc Pfetsch, Andreas Tuchscherer and Benjamin Hiller for proof reading, encouraging me, and giving helpful comments. At last, I thank all of my colleagues at the Zuse Institute for the good work atmosphere.

[2]grant no. 03-GRM2B4

Contents

Zusammenfassung i

Abstract ii

Preface iii

Acknowledgments v

1 Planning in Public Transit **1**

1.1 Introduction 1

1.2 Classification of Planning Steps 3

1.3 Basic Models 3

1.3.1 Flow Based Models 4

1.3.2 Path Based Models 6

1.4 Network Design 8

1.4.1 Description 8

1.4.2 Models 8

1.4.3 Applications 9

1.5 Line Planning 10

1.5.1 Description 10

1.5.2 Models 10

1.5.3 Approaches 12

1.5.4 Integration 13

1.6 Planning of Bus Stops 13

1.7 Timetabling 14
1.7.1 Description 14
1.7.2 Models 15
1.7.3 Algorithms 16
1.7.4 Integration 17
1.8 Planning of Public Tenderings 17
1.9 Vehicle Scheduling 18
1.9.1 Description 18
1.9.2 Graph Theoretic Model 19
1.9.3 Integer Programming Model 21
1.9.4 Algorithms 21
1.9.5 Algorithm VS-OPT 22
1.9.6 Integration 23
1.10 Duty scheduling 24
1.10.1 Description 24
1.10.2 Graph Theoretic Model 25
1.10.3 Integer Programming Model 27
1.10.4 Algorithms 28
1.10.5 Integration 28
1.11 Rostering 28
1.11.1 Description 29
1.11.2 Model 30
1.11.3 Algorithms 30
1.11.4 Integration 31
1.12 Conclusion 31

2 Integration of Vehicle and Duty Scheduling 33
2.1 Motivation . . . 33
2.1.1 Regional Public Transit . . . 34
2.1.2 Vehicle and Duty Schedule Efficiency . . . 35
2.1.3 Vehicle and Duty Costs . . . 36
2.2 Approaches to Integrated Scheduling . . . 37
2.2.1 Duty Scheduling with Vehicle Scheduling Constraints . . . 38
2.2.2 The Combined Approach . . . 39
2.2.3 Full Integration of Vehicle and Duty Scheduling . . . 42
2.3 Literature . . . 43
2.3.1 Ball, Bodin and Dial . . . 44
2.3.2 Vehicle Scheduling Centered Approaches . . . 44
2.3.3 Duty Scheduling Centered Approaches . . . 45
2.3.4 Fully Integrated Vehicle and Duty Scheduling . . . 47
2.4 IS-OPT . . . 50
2.4.1 Outline of our ISP-Algorithm . . . 50
2.4.2 Contributions . . . 52

3 Basic Methodology 53
3.1 Column Generation . . . 53
3.2 Lagrangean Relaxation . . . 55
3.2.1 Lagrangean Relaxation in General . . . 55
3.2.2 Linear Programming Duality . . . 56
3.2.3 Quadratic Programming Duality . . . 57
3.3 Lagrangean Relaxation for Column Generation . . . 58
3.3.1 Problem Class . . . 59
3.3.2 Restricted Problem . . . 59
3.3.3 Pricing Problem . . . 60
3.3.4 Lagrangean relaxation . . . 60
3.3.5 Reduced Cost Shifting . . . 62

4 Proximal Bundle Method **65**

4.1 Description 66

4.1.1 Idea and Properties 66

4.1.2 Subgradients, Linearizations, and Cutting Plane Models 68

4.1.3 Quadratic Subproblem 68

4.1.4 Algorithm 72

4.1.5 Weight updating 73

4.1.6 Notes On The Convergence 73

4.2 Comparison with other Subgradient Methods 74

4.3 Modifications and Extensions 75

4.3.1 Separable Functions 75

4.3.2 Primal Approximation of Linear Programs 76

4.3.3 Handling of Bounded Functions 78

4.4 Active Set Method 80

4.4.1 Description 81

4.4.2 Exact Active Sets 83

4.5 Applications 84

4.5.1 Approximating the Duty Scheduling Problem 85

4.5.2 Approximating the Problem ISP 87

4.6 Inexact Bundle Method 88

4.6.1 Literature 89

4.6.2 Vehicle Scheduling Component Function 90

4.6.3 Duty Scheduling Component Function 91

4.6.4 Combined Functions 98

4.7 Computational Results 100

4.7.1 Testbed 100

4.7.2 Results 101

5 The Generation of Duties 107

5.1 Motivation and Notation . . . 107
5.1.1 Master Problem . . . 108
5.1.2 Size of the Master Problem . . . 108
5.1.3 The Pricing Problem . . . 110
5.2 Description of Duties . . . 110
5.2.1 Duty Elements . . . 111
5.2.2 Duty types . . . 112
5.2.3 Resources . . . 114
5.2.4 Break rules . . . 115
5.2.5 Cost of a Duty . . . 115
5.3 Models for the Pricing Problem . . . 116
5.3.1 Pricing Networks . . . 117
5.3.2 Timelines . . . 118
5.3.3 IP Model . . . 121
5.3.4 Cost and Reduced Cost of Pairings and Links . . . 123
5.4 Literature . . . 124
5.5 Algorithm . . . 126
5.5.1 The Resource Constrained Shortest Path Problem . . . 126
5.5.2 Lagrangean Relaxation of all Resource Constraints . . . 127
5.5.3 Depth-First-Search . . . 130
5.6 Labeling . . . 131
5.6.1 Graph Construction . . . 132
5.6.2 Node Dominance . . . 134
5.6.3 Resource Scaling and Rounding . . . 136
5.6.4 Cost scaling . . . 139
5.7 Computational Results . . . 139

5.7.1 Testbed 140
5.7.2 RCSP 141
5.7.3 Lagrangean Relaxation 143
5.7.4 Results of the Enumeration Algorithm 146
5.8 Lower Bounds for the Duty Scheduling Problem 147
5.8.1 RCSP-lower-bound 147
5.8.2 LP lower bounds 148
5.9 Conclusion 149

6 Rapid Branching **151**
6.1 Overview 152
6.2 Branch-and-Bound 152
6.3 Perturbation Branching Rule 155
6.3.1 Motivation 155
6.3.2 Determining the Main Branch 155
6.3.3 Calculation of the Perturbed Objective Function 157
6.3.4 Calculation of the Other Branches 158
6.3.5 Comparison with Other Branching Rules 159
6.4 Node Selection 160
6.5 Lower Bounding 161
6.6 Upper Bounding 161
6.7 MIP-heuristics in the Literature 162
6.7.1 Simplex Based Heuristics 162
6.7.2 OCTANE 163
6.7.3 Set Covering and Set Partitioning Heuristics 163
6.7.4 Branch-and-Bound Based Heuristics 163
6.7.5 Rounding Heuristics 165
6.8 Computational Results 166

6.8.1 Testbed 166
6.8.2 Observations 168
6.8.3 Conclusion 169

7 Computational Results **171**
7.1 Running Time 171
7.2 Computation Times per Phase 173
7.3 Algorithms 174
7.4 RVB Instances 174
7.5 RKH Instances 176
7.6 Subcontractor Planning for an Regional Carrier 178
7.7 ECOPT Instances 180
7.8 Conclusion 182

Glossary **185**

Symbols **187**

Bibliography **189**

Chapter 1

Planning in Public Transit

In this chapter we give an overview of the planning process in public transit. We will describe the planning problems associated with the operation of a network of bus lines and present state of the art mathematical models and algorithms to solve them. We will show the general trend that the used planning scenarios become larger either by looking at larger planning units or by looking at more planning steps at once. Both approaches lead to potential better solutions by more degrees of freedom in the planning problems.

Some of the problems occurring in the planning process of bus traffic are similar to planning problems in other modes of public transportation, such as subways, trams, trains, or even airlines. We will comment on the details of the similarities and differences in the respective sections. Bussieck et al. [1997] describe the use of discrete optimization in the planning process of public rail transports. We concentrate here on recent models and algorithms. An overview about planning in public transport until 1994 can be found in the article of Odoni et al. [1994]. Borndörfer et al. [2006] highlight the increasing use of OR methods for planning problems in public transport and describe exemplarily some applications.

We use in this thesis the definitions and notations of Schrijver [2003] for graphs, linear programs (LPs), and mixed integer programs (MIPs). A short overview of the used symbols can also be found in the annex.

1.1 Introduction

In the last years, the budgets of the federal government, the states, and the municipalities in Germany were very tight. Therefore the federal government

has cut the so called "Regionalisierungsmittel", that is, subsidies for regional public transport in Germany, from €7.1 billion to €6.6 per year[1]. Also other subsidies were or will be reduced: Berlin has cut its subsidies to its public transport company BVG, by €100 millions from 2005 to 2006[2] and further reductions will likely follow. A study of Resch et al. [2006] reports reductions of subsidies of 32–44% of three German public transit companies[3]. Also more and more tenderings for the subsidies for public transit are put out instead of giving them directly to the local companies. Thus, public mass transit companies in Germany are under the pressure to reduce their costs. This can be accomplished by either discontinuing unprofitable lines, by lowering the wages of the personnel, or by increasing the efficiency of the schedules. All of these measures have been taken in the past. In the following we will examine where new mathematical methods may help to further improve the efficiency without disadvantages for the drivers or the passengers. We assume that even if computer based planning systems are already in use, new mathematical approaches are able to solve larger planning problems at once and help to integrate subsequent planning steps.

Besides the financial goals of the planning process also the general benefit of public transport, summarized as public welfare, is of concern. The following often opposing objectives are common for strategic and operational planning in public transport:

- increasing the attractiveness of public transport,
- reducing operation cost for a given service level,
- increasing the transport capacity for a given budget,
- reducing medium term capital investment (e.g., by reducing the number of buses, number of stops, or number of depots).

The improvement of the results of the planning process with respect to these objectives is not the only advantage of using optimization methods. The possibility of calculating alternative scenarios in short time is also of great interest for public transport companies because it backs up the decision process with reliable information.

[1] Haushaltsbegleitgesetz of Berlin, 2006

[2] business report BVG 2005

[3] The remaining subsidies are still about 9–21% of the total costs of the companies.

1.2 Classification of Planning Steps

Usually the planning process in public transport is divided into *strategic* and *operational* planning. Sometimes also tactical planning is mentioned as an intermediate step. Strategic planning consists of problems dealing with long term decisions about the infrastructure and the level of service, whereas operational planning handles the problems which occur in the operation of a given service.

We will concentrate in the sections below on the following planning steps:

1. Strategic planning:
   - network design,
   - line network planning,
   - time table planning.
2. Operational planning:
   - vehicle scheduling,
   - duty scheduling,
   - crew rostering.

We remark that the collection of passenger data as an input to strategic planning problems is a different problem on its own, which we do not discuss here. Often mentioned in this context are operational problems, such as the dispatching of vehicles or the recovery of planned schedules after delays or breakdowns. These problems need specialized algorithms due to their real time requirements, which are beyond the scope of this work. A recent overview of literature about the treatment of delays in vehicle and duty scheduling and an algorithm to deal with it can be found in Huisman [2004].

Another important planning step, which is in general not conducted by the public transit companies but by local authorities, is the planning of public tenderings of public transit for certain lines or regions. We cover this topic because it can use methods of the other planning steps.

1.3 Basic Models

We introduce two basic mathematical planning models whose variants are often used in public transport planning problems or other scheduling problems. We show the connections between those variants.

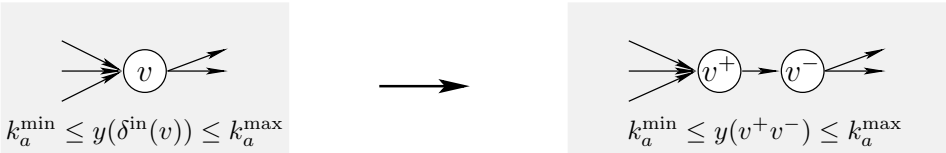


Figure 1.1: Modeling capacities on nodes

1.3.1 Flow Based Models

Many problems related to scheduling or transportation problems can be modeled as flow problems in the following general sense: Given is a directed network with a set of nodes V and a set of arcs A connecting the nodes, a source $s \in V$, and a sink $t \in V$. Additionally, we have a minimum capacity $k_a^{\min} \geq 0$ and a maximum capacity $k_a^{\max} \geq k_a^{\min}$ as well as a cost c_a per unit of flow over each arc $a \in A$. The goal is to find a minimum cost flow from s to t subject to the capacity constraints and eventually certain other constraints. This task can be formulated as the following *generalized minimum cost flow problem* (GMCF):

$$
\begin{aligned}
&\text{(GMCF)} \quad \min\ c^{\mathsf{T}} y, \\
&\qquad \text{s.t.} \\
&\text{(i)} \qquad y(\delta^{\text{in}}(v)) - y(\delta^{\text{out}}(v)) = 0, \qquad \forall v \in V \setminus \{s,t\}, \\
&\text{(ii)} \qquad k_a^{\min} \le y_a \le k_a^{\max}, \ \forall a \in A, \\
&\text{(iii)} \qquad By = b \text{ or } By \le b, \\
&\text{(iv)} \qquad y_a \geq 0, \qquad \forall a \in A, \\
&\text{(v)} \qquad y \in \mathbb{Z}^A.
\end{aligned}
$$

Here, y_a is the flow variable of arc a giving the number of units of the flow over arc a. Because the decision variables are associated to arcs, we say (GMCF) is an *arc-based model*.

The flow conservation constraints (i) ensure that at every node the in-flow is equal to the out-flow. The capacity constraints (ii) guarantee that the arc capacities are satisfied. These constraints can also be used to enforce minimum or maximum flows over a node as follows: Replace a node by two new adjacent nodes and a new arc connecting these nodes. The lower capacity on the new arc is the desired minimum flow (see Figure 1.1). Additional properties of the flow can be modeled by constraints (iii). Here either equalities or inequalities or a mixture is possible, $B \in \mathbb{R}^{\mathcal{R} \times A}$ and $b \in \mathbb{R}^{\mathcal{R}}$

are an appropriate matrix and vector, and $\mathcal{R}$ is an index set. Inequalities (iv) ensure the non-negativity of the flow. Constraints (v) model integrality. This is required, e.g., to solve the problem of assigning integral resources like vehicles or drivers to certain activities.

The minimum cost flow problem (MCF) without constraints (iii) is solvable in polynomial time by special network algorithms (see Ahuja et al. [1993]) or by a specialized version of the simplex algorithm (see, e.g., Löbel [1996]).

The multi-commodity flow problem is a specialization of (GMCF), for which constraints (iii) of (GMCF) take the shape

$$\text{(iiia)} \quad y(\delta_g^{\text{in}}(v)) - y(\delta_g^{\text{out}}(v)) = 0 \quad \forall (v,g) \in V \setminus \{s,t\} \times G.$$

Here G is the set of *commodities*, $\delta_g^{\text{in}}(v) := \delta^{\text{in}}(v) \cap A_g$, and $\delta_g^{\text{out}}(v) := \delta^{\text{out}}(v) \cap A_g$, whilst $A_g, g \in G$ are disjoint subsets of A, and $\bigcup_{g \in G} A_g = A$. The equations (iiia) partition the network into $|G|$ subnetworks which each give rise to independent flow conservation constraints. Thus, equations (iiia) render equations (i) redundant. However, equations (i) are still useful when relaxing equations (iiia).

Multi-commodity flow problems occur in the planning of telecommunication networks, where each traffic gives rise to a single commodity, or as a subproblem of the network design problem (see next section). Also the multi-depot vehicle scheduling problem (see section 1.9) can be modeled as a multi-commodity flow problem.

Another specialization of (GMCF) is the minimum cost flow problem with resource constraints. For this problem we replace constraints (iii) by

$$\text{(iiib)} \quad Ry \leq \ell_r \quad \forall r \in \mathcal{R}.$$

Here $R \in \mathbb{R}^{\mathcal{R} \times A}$ is a matrix whose entry in the r-th row and a-th column gives the *resource consumption* of a resource $r \in \mathcal{R}$ by arc a. The vector $\ell \in \mathbb{R}^{\mathcal{R}}$ gives the maximal allowed consumption of the resources. A special case of (GMCF) with constraints (iiib) is the resource constraint shortest path problem. It occurs, e.g., in the duty generation subalgorithm of our duty scheduling algorithm (see Section 5.5.1).

Another practically relevant special case of (iiib) are the following "infeasible path constraints" (iiic), which are used to remove a set of forbidden paths $\mathcal{P}$ in feasible flows:

$$\text{(iiic)} \quad \sum_{a \in P} y_a \leq |P| - 1, \quad \forall P \in \mathcal{P}.$$

This type of constraints can be used to model complicated constraints on subflows, as they occur for example in duty scheduling problems or in vehicle scheduling problems with maintenance requirements, which are described below.

The model (GMCF) with constraints (iiic) is difficult to solve in practice, if the set $\mathcal{P}$ of infeasible paths is large in comparison to the set of all possible paths. This occurs, for example, in the duty scheduling problem since here most paths in the graph are not representing a valid duty (see Schlechte [2003]).

1.3.2 Path Based Models

To overcome the difficulties to solve (GMCF) with many constraints of type (iiic) the flow model can be transformed by Dantzig-Wolfe decomposition (Dantzig & Wolfe [1960]) into a path based model. The idea of this transformation is that each *st*-flow can be decomposed into a sum of *st*-paths and cycles. Thus, the problem (GMCF) can be reformulated as a problem of finding a (cost minimal) set of paths and cycles such that the resulting flow fulfills all capacity constraints. We call the result of this transformation *generalized path covering problem* (GPCP).

$$
\begin{aligned}
&\text{(GPCP)} \quad \min\ d^{\mathsf{T}}x, \\
&\qquad\qquad \text{s. t.} \\
&\text{(i)} \qquad\qquad k_a^{\min} \leq \sum_{P \ni a} x_P \leq k_a^{\max}, \qquad \forall a \in A, \\
&\text{(ii)} \qquad\qquad\qquad x_P \geq 0, \qquad\qquad \forall P \in \mathcal{S}, \\
&\text{(iii)} \qquad\qquad\qquad x \in \mathbb{Z}^{\mathcal{S}}.
\end{aligned}
$$

Here $\mathcal{S} \subset \mathcal{P}(A)$ is the set of feasible arc sets. In our applications of this model these sets form *st*-paths. The variables $x_P, P \in \mathcal{S}$ indicate how much flow is routed over path P. Inequalities (i) guarantee, like the constraints (GMCF)(ii), the compliance with the minimum and maximum capacities $k_a^{\min}$ and $k_a^{\max}$ on every arc $a \in A$. The notation $P \ni a$ denotes all arc sets $P \in \mathcal{S}$ that include arc a. Constraints (ii) and (iii) ensure non-negativity and integrality of the variable x_P, $P \in \mathcal{S}$. The cost $d_P, P \in \mathbb{R}^{\mathcal{S}}$, denotes the cost of a certain arc set P. Often $d_P = \sum_{a \in P} c_a$, that is, the cost of an arc set P is the sum of the cost of its arcs, or $d_P = 1$ for all $P \in \mathcal{S}$ if only the number of arc sets should be minimized.

In general, the set $\mathcal{S}$ of feasible arc sets is very large. Nevertheless models of this kind can be solved by column generation approaches (see Barnhart et al. [1998] or Section 3.1). Specializations of this problem are the path packing, path partitioning, as well as the path covering problem. In these problems the arc sets form st-paths. In the path covering problem all minimum capacities are one and the maximum capacities are infinite, i.e. $k_a^{\min} = 1$ and $k_a^{\max} = \infty$ for all $a \in A$, and in the path packing problem the maximum capacities are one ($k_a^{\max} = 1$) and the minimum capacities are zero ($k_a^{\min} = 0$) for each $a \in A$. The path partitioning problem finally has $k^{\min} = \mathbb{1} = k^{\max}$.

A slightly more generalized form of these problems are the well known set covering (SCP), set packing (SSP), and set partitioning (SPP) problems. In these problems the paths in a graph are replaced by subsets P of a basic set A. These problems can be formulated as follows:

$$
\begin{aligned}
&(\mathsf{SCP}) \quad \min\ d^{\mathsf{T}}x,\\
&\text{s. t.}\\
&\text{(i)} \qquad \sum_{P \ni a} x_P \geq 1 \qquad \forall a \in A,\\
&\text{(iii)} \qquad x \in \{0,1\}^{\mathcal{S}},
\end{aligned}
$$

$$
\begin{aligned}
&(\mathsf{SSP}) \quad \min\ d^{\mathsf{T}}x,\\
&\text{s. t.}\\
&\text{(i)} \qquad \sum_{P \ni a} x_P \leq 1 \qquad \forall a \in A,\\
&\text{(iii)} \qquad x \in \{0,1\}^{\mathcal{S}},
\end{aligned}
$$

and

$$
\begin{aligned}
&(\mathsf{SPP}) \quad \min\ d^{\mathsf{T}}x,\\
&\text{s. t.}\\
&\text{(i)} \qquad \sum_{P \ni a} x_P = 1, \qquad \forall a \in A,\\
&\text{(iii)} \qquad x \in \{0,1\}^{\mathcal{S}}.
\end{aligned}
$$

A survey about those problems can be found in Borndörfer [1998].

It is problem specific whether the formulation as a GMCF or a path oriented problem such as SSP, SCP, or SPP is appropriate. In our experience the path formulations are easier to solve if the network contains many infeasible paths. It may also be necessary to formulate a problem path based if the network is very large. This occurs regularly in multi-commodity flow

problems with many commodities. If there are only a few additional constraints of type (iii) in (GMCF) the arc based formulation seems in general to give better results. A reason for this may be that for problems arising in applications the LP-relaxation is often already nearly integral.

1.4 Network Design

1.4.1 Description

The network design problem (NDP) consists of selecting routes that can be used by bus lines. The routes are meeting at the *transfer points*. At these points buses can change their route. The routes have to be selected in such a way that a given demand of traffic can be handled and that the cost for setting up the routes and the cost for using them are minimized.

Typically only the extension or modification of an existing network is considered since in most cases historically grown transportation networks exist that can not be modified easily, and because any alteration of lines involves expenses for removal or addition of bus stops, printouts of timetables, building new parking facilities, and marketing to inform the passengers about the changes.

1.4.2 Models

A framework for a slightly more general class of network design problems is presented in Kim & Barnhart [1997]. The model is based on a very similar model in Magnanti & Wong [1984]. We want to sketch here the mixed integer programming model of Kim & Barnhart [1997] and discuss its properties. Given is a network $N = (V, A)$. The nodes V represent potential end points and transfer points of bus lines which include also the origins and destinations of traffic. The set of arcs A models the physical links between these points called *routes*. The set of origin/destination pairs (OD-pairs) is denoted by P, while $O(p)$ and $D(p)$ for $p \in P$ are the origin or destination of p. Let b^p denote the demand of traffic from $O(p)$ to $D(p)$ (measured in number of passengers). The set of potential traffic modes is denoted by F. Typical traffic modes are different types of buses, such as articulated buses or double deckers, trams, or different types of trains. Let u_a^f denote the passenger capacity of one unit of traffic mode $f \in F$ on arc a. The cost $c_a^p, p \in P, a \in A$

and $d_a^f, f \in F, a \in A$ denote the variable cost of traffic depending on p and the fixed cost of providing one unit of traffic mode f with respect to arc a, respectively. Now the integer design variables y_a^f indicate the number of units of traffic mode f that uses arc a in the solution. The flow variables x_a^p denote the fraction of the traffic b^p which is routed over arc a.

$$
\begin{aligned}
&\text{(NDP)} \quad \min \textstyle\sum_{p\in P} b^p c^{p\mathsf{T}} x^p + \sum_{f\in F} d^{f\mathsf{T}} y^f \\
&\qquad \text{s.t.} \\
&\text{(i)} \qquad \textstyle\sum_{p\in P} b^p x_a^p - \sum_{f\in F} u^f y_a^f \le 0, \quad \forall a \in A \\
&\text{(ii)} \qquad x(\delta_p^{\text{in}}(u)) - x(\delta_p^{\text{out}}(u)) = 0, \quad \forall p \in P, u \in V \setminus (O(p) \cup D(p)), \\
&\text{(iii)} \qquad x(\delta_p^{\text{in}}(D(p))) = 1, \quad \forall p \in P, \\
&\text{(iv)} \qquad x(\delta_p^{\text{out}}(O(p))) = 1, \quad \forall p \in P, \\
&\text{(v)} \qquad x_a^p \ge 0, \quad \forall p \in P, \quad a \in A, \\
&\text{(vi)} \qquad y_a^f \in \mathbb{N}_0, \forall a \in A, \quad f \in F.
\end{aligned}
$$

Inequalities (i) limit the amount of flow on an arc a to its installed capacity, determined by the design variables. The flow equations (ii), (iii), and (iv) ensure that each traffic demand is routed from its origin to its destination. Constraints (v) and (vi) ensure non-negativity of all variables and integrality of the design variables. The model (NDP) can be seen as a multicommodity flow problem with the OD-pairs as commodities and additional slack variables y_a^f for maximum capacity constraints.

This model can be seen as a network flow problem with additional setup cost for using an arc for the first time.

Drawbacks of this model are that the flows for an OD-pair can not be controlled, and therefore the resulting routes of the passengers may be unacceptable in practice due to their length or the number of transfers. Also the structure of the designed network may not be reasonable for the operation of lines, because there may not exist a "good" decomposition of the solution of the NDP into lines since the lines can, e.g., become too short to be operated economically.

1.4.3 Applications

There are, as far as we know, no publications that apply mathematical models and methods to a real or at least realistic approximation of a problem of a traffic company. The reason seems to be that, on the one hand, a complete redesign or a construction of a traffic network from scratch occurs very

rarely. On the other hand, many design decisions are heavily influenced by politics, which cannot be modeled properly. But we think that a mathematical approach is nevertheless useful for extensions or modifications of existing networks or for evaluation purposes.

1.5 Line Planning

1.5.1 Description

The line planning problem (LPP) consists of designing lines and their frequencies, if a network of possible end and transfer points of lines and physical links between them and a set of OD-pairs and associated demands is given. Eventually some of the links are only usable by certain modes of traffic, such as roads that can only be used by buses or railways that can only be used by trains. The set of OD-pairs is the same as in the network design problem. The line routes and their frequencies must be constructed in such a way that they are able to satisfy the transport volume given by the OD-pairs. The objective is to minimize the operation cost as well as the inconvenience of the users. That means we try to reduce their dwell times in the system, as well as the number of transfers which are necessary to get from the origin to the destination.

In practice, the usable links are not given by a preceding NDP optimization but by used plus potential ones that are manually selected. In general, the LPP does not consider set up costs for new routes.

1.5.2 Models

The LPP itself is already a problem that consists of two subproblems. On the one hand lines and their frequency have to be selected, on the other hand it has to be determined how the passengers utilize these lines to satisfy their transportation needs.

In many articles the LPP is solved sequentially. At first the traffic volume of an OD-pair is distributed over the arcs of the network by the algorithm of Bouma & Oltrogge [1994]. This algorithm routes passengers through the network by rules which should simulate the behavior of real travelers. Then a path-based model for the line planning problem can be formulated where each potential line is a path in a network $N = (V, A)$. We use the same

network and notation for OD-pairs as for the NDP. Let now be $\mathcal{L}$ the set of potential lines, c_ℓ the cost, and $\mathcal{F}_\ell$ the set of potential frequencies for line $\ell \in \mathcal{L}$. In general, every set $\mathcal{F}_\ell, \ell \in \mathcal{L}$ also includes 0 for not using the corresponding line. Let $A \in \{0,1\}^{A \times \mathcal{L}}$ be the arc-line incidence matrix, and let finally be $b \in \mathbb{R}^A$ the demand of traffic over the arcs. Then we formulate a model for the LPP as

$$
\begin{array}{lll}
\text{(LPP)} & \min\ c^\mathsf{T} x, & \\
& \text{s. t.} & \\
\text{(i)} & \quad Ax \geq b, & \\
\text{(ii)} & \quad x_\ell \in \mathcal{F}_\ell,\ \forall \ell \in \mathcal{L}. &
\end{array}
$$

The decision variables x give the frequency for the used lines and zero if a line is not used. Variants of this of model are, e.g., used in Bussieck et al. [2003] and Claessens et al. [1998]. There also the number of carriages of a line plays a role because lines of railways are planned. In Goossens et al. [2004] a model with binary decision variables is proposed by introducing a variable for each combination of line and frequency.

Another model that integrates the determination of the passenger flows and the finding of lines with their frequencies is proposed by Borndörfer et al. [2004]. There two path-based models for the lines and the passengers are coupled. It uses additional variables $y_p \in \mathbb{R}_+$ for every path $p \in P$. Here P is the set of potential paths that passengers use to get from a start point of an OD-pair to an end-point. The variables x_ℓ, $\ell \in \mathcal{L}$ are binary variables that are one if a line is used and zero otherwise. The frequency of a line is given by a third kind of variable, namely $f_\ell > 0$, $\ell \in \mathcal{L}$. The maximum capacity of a line is given by F_ℓ. Like in (NDP) b^p denotes the traffic demand for an OD-pair p. We denote the set of all paths from the source to the sink of an OD-pair p by P^p and the set of all paths that use an arc $a \in A$ by $P(a)$. We denote the set of lines that uses an arc $a \in A$ by $L(a)$. The vector $\kappa \in \mathbb{R}_+^{\mathcal{L}}$ gives the capacities of the lines in terms of passengers. To calculate the cost of a line plan we need the inconvenience d_a of a passengers that uses an arc a, the set up cost c_ℓ and the operating cost e_ℓ of a line $\ell \in \mathcal{L}$. Now we are

able to formulate model (LPP2):

$$
\begin{aligned}
&\text{(LPP2)} \quad \min\ c^{\mathsf{T}}x + d^{\mathsf{T}}y + e^{\mathsf{T}}f,\\
&\qquad \text{s. t.}\\
&\text{(i)} \qquad\qquad y(P^p) = b^p, \quad \forall \text{OD-pairs } p,\\
&\text{(ii)} \qquad \sum_{l \in L(a)} \kappa_\ell f_\ell - y(P(a)) \geq 0, \quad \forall a \in A,\\
&\text{(iii)} \qquad\qquad f \leq Fx,\\
&\text{(iv)} \qquad x \in \{0,1\}^{\mathcal{L}}, 0 \leq f \leq F,\ y \geq 0.
\end{aligned}
$$

Constraints (i) ensure that all traffic demands are met. They are equivalent to the flow conservation constraints (ii), (iii), and (iv) of (NDP). Constraints (ii) couple the passenger flow and the provided capacity of the lines. Constraints (iii) are forcing x_ℓ to one if line ℓ has a positive frequency.

We have left out in both models the constraints for minimum frequencies for the sake of simplicity. In practice various additional requirements on lines and line plans occur which are modeled as additional constraints in the literature.

We think that (LPP2) is more appropriate for bus and especially urban traffic, because in urban traffic the routing of the passenger is highly dependent on the existing lines, since in most cases a passenger can choose between different reasonable routes to get from one location to another. (LPP) is more appropriate for railway traffic, because there, in general, the network is sparser and hierarchically structured. Therefore the travel routes of the passengers are more ore less predetermined.

1.5.3 Approaches

In Claessens et al. [1998] a variation of model (LPP) with binary decision variables and additional constraints for maximal frequencies is used. This problem is solved by first generating all possible lines, which have to be "reasonable" (e.g., at most twice as long as the shortest path between its endpoints). The resulting integer program is then solved by a branch-and-bound approach. Other publications such as Bussieck et al. [2003] treat a non-linear model that stems from adding variables for the number of carriages per line. In Schöbel & Scholl [2006] two passenger centric approaches are presented. The first minimizes the overall travel times of the passengers, the second minimizes the number of transfers of passengers, both with a certain budget to establish lines. Computational results to solve the second problem

with Dantzig-Wolfe-decomposition are provided. However, the maximum number of lines considered in these computations seem to be rather small for real world applications.

A short review of recent publications about line planning and an improved version of the algorithm of Claessens, Dijk and Zwaneveld using Branch-and-Cut can be found in Goossens et al. [2004]. Another model that does not minimize cost but maximizes the number of passengers that do not have to change a line is proposed in Bussieck et al. [1997].

An approach to solve the LP-relaxation of (LPP2) with dynamic column generation is proposed in Borndörfer et al. [2004]. With this approach lower bounds for the LPP of medium public transit carriers can be calculated.

1.5.4 Integration

The model (LPP2) is already an model that integrates the line planning problem with the passenger flows. A drawback of this model as well as of model (LPP) is that they cannot consider the number of line changes of the passengers in the objective function. It would be preferable to simultaneously minimize the cost and the number of changes. However, at the moment the models for this problem are too large to be solvable.

1.6 Planning of Bus Stops

In the line planning problem only end points of lines and transfer points are determined, the locations of intermediate stops were not taken into account. These locations influence the comfort of the passengers, and therefore also the attractiveness of the line. Furthermore, additional stops are slowing down the average trip time and are increasing the operation cost.

The literature about the determination of the number and locations of bus stops is dealing mostly with simplified networks, such as single lines, or finite or infinite grid networks. An example is Chien et al. [2003]. Here a subset of possible bus stops on a segment of a bus route is selected by complete enumeration of all possible combinations of bus stops, minimizing user and supplier cost. Another example is van Nes & Bovy [2000], where a section of a network is examined, in which a homogenous coverage by parallel lines of a certain spacing is assumed. A spacing of the lines and a distance between the bus stops is calculated which again minimizes the sum of the

cost of the supplier and the users. The non-linear models in this article were either solved analytically or numerically by standard software.

It seems to be highly unrealistic to integrate the planning of bus stops into the network design problem or the line planning problem, because only simplified networks are examined here. Additional, it is unclear how potential bus stops should be integrated into the planning network of the NDP or LPP.

1.7 Timetabling

Once the lines and the service frequencies are established, the precise arrival and departure times of the vehicles at the stations have to be determined. That is, the result of this planning step are the *timetables* of the different lines at the stations. These timetables should often be cyclic or periodic. The goal is to minimize the waiting times (in- and outside the vehicle) for transfers from one line to another. Additional usual requirements are that the departure times are distributed uniformly over a certain time period. That means, e.g., that it is not allowed that all vehicles depart in a short time span and then for a long time span nothing happens. Sometimes also the number of vehicles needed to operate the resulting timetable is part of the objective function or limited.

Since the literature on timetabling without periodicity is very sparse and applications of it are not know to us, we will discuss it here only very shortly and then describe the timetabling problem with periodicity in more detail. A model for the timetabling problem without periodicity was proposed by Ceder & Tal [1997]. They give a straightforward mixed integer program, which maximizes the number of possible transfers without waiting time. In this model it is possible to give minimal and maximal headways between two adjacent bus departures at the same route. It was not possible with a standard MIP solver to solve other than very small problems, using five transfer nodes and five routes. Therefore a kind of greedy heuristic is given, which subsequently fixes the departure times of the buses.

1.7.1 Description

Most European public transport companies are using periodic timetables, i.e., the time intervals between two departures of the same line at the same station have the same length for some period of time. The least common

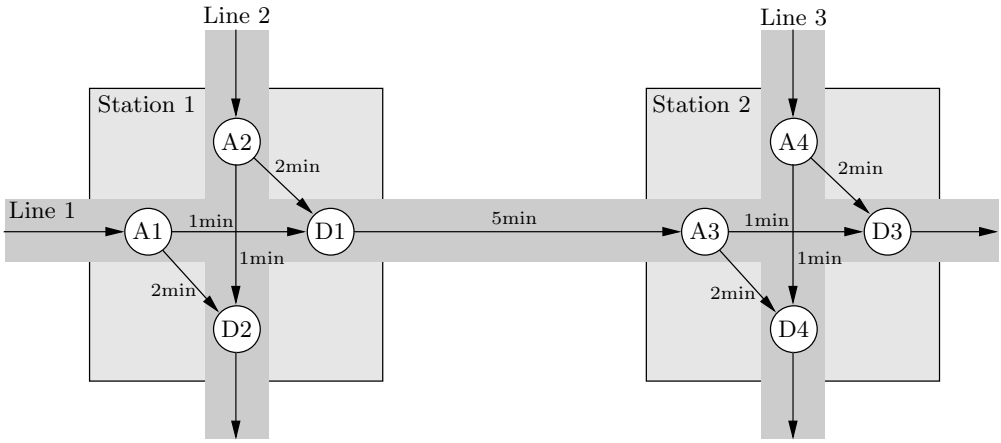


Figure 1.2: Graph for the Timetabling Model

multiple of these intervals (usually one hour) is the *planning period* T. After this planning period, all departures at time t reoccur at time $T + t$. The problem of finding these cyclic departure times minimizing the waiting time of passengers that want to change a line is called the *periodic timetabling problem*.

1.7.2 Models

Recent models for the periodic timetabling problem are based on a model for the periodic event scheduling problem (PESP) by Serafini & Ukovich [1989]. Additional to the planning period T a set of events V is given. For timetabling an event is the arrival or departure times of a directed line at a certain location. Furthermore we have a set of arcs A connecting pairs of interdependent events. For every event $v \in V$ a time π_v has to be found such that the sum of time spans between interdependent events i and j with $ij \in A$ is minimized. The time-span between two events can be expressed by $\pi_j - \pi_i - \ell_a$. The constants ℓ_a, and u_a, $a = ij \in A$ denote the minimum and maximum time-span between to events i and j. The term

$$(\pi_j - \pi_i - \ell_{ij}) \mod T$$

can then be interpreted as the avoidable waiting time between the events i and j. Finally c_{ij} gives a weight how important a link between two events i and j is. This could, e.g., be the number of passengers that attend to event i

as well as event j

$$\begin{aligned}
&\text{(PESP)} \quad \min \sum_{a=(i,j)\in A} c_a((\pi_j - \pi_i - \ell_a) \bmod T), \\
&\text{s. t.} \\
&\text{(i)} \qquad (\pi_j - \pi_i - \ell_a) \bmod T \leq u_a - \ell_a, \qquad \forall a = (i,j) \in A, \\
&\text{(ii)} \qquad \pi_v \geq 0, \qquad \forall v \in V.
\end{aligned}$$

Here ℓ_a and u_a are lower and upper bounds for the duration between the two events i and j. This model can be transformed into an mixed integer program (see Liebchen & Möhring [2002]).

Figure 1.2 illustrates the arrival and departure of different lines at two stations. Events A1 and D1 symbolize the arrival and departure of line 1 at station 1, and A3 and D3 the arrival and departure at station 2 of the same line. The other events symbolize arrivals and departures of other lines. Each time on an arc is the minimum time-span between two events. The time between event A1 and D1 can, e.g., stem from the time needed by the passengers to leave and enter the vehicle. The time on arcs between events of different lines may be the walking time from one platform to the one of the other line. The time between event D1 and A3 finally is the driving time to get from station 1 to station 2.

1.7.3 Algorithms

In Liebchen & Möhring [2004] a survey of algorithms for periodic timetabling can be found and an outlook of possible enhancements and integration with line planning and vehicle scheduling are given. A practical application of their timetabling algorithm is presented in Liebchen & Möhring [2002]. In this article the timetables of the Berlin subway were successfully optimized under consideration of the needed number of vehicles by a standard MIP-solver

An extension and application of the timetabling problem for railways can be found in Kroon & Peeters [2003]. They consider the problem of safety of trains using the same track in the PESP. These problems were solved by a commercially available timetabling system DONS (see Hooghiemstra et al. [1999]), which is based on a constraint programming solver.

An approach which considers line planning issues in the timetabling problem is described in Lindner [2000]. Here a cutting plane algorithm based on an IP-model is used.

1.7.4 Integration

The articles mentioned above show that intentions to integrate the timetabling either with the previous step of line planning or with the subsequent step of vehicle planning exist. However, at the moment this integration is either rudimental or academic. Also, in all these publications timetables are optimized for a standard hour, and peak times are ignored or considered manually in a post-processing step.

1.8 Planning of Public Tenderings

Another important topic in the planning process of public transport is the planning of public tenderings. In the European Union it is common practice that local authorities, such as governments of counties, cities, or even states, subsidy the operators of public transit which in return guarantee a defined service level. We will call these authorities *orderers*, since they order a certain level of public transport from the *operators*, i.e., the traffic companies which operate the public transit. Sometimes the orderers own the operators, what can be problematic in terms of market access for competitors.

Due to a recent proposal of an amendment to the directive 1191/69 in the version 1893/91 of the European Union, the liberalization of the market for public transit has started in the European Union. This directive proposes economic independence of public transit carriers and the elimination of distorting conditions of competition between them. The amendment of this directive, which was published in February 2002, concerns "public service requirements and the award of public service contracts in passenger transport by rail, road and inland waterway". The most important point in this proposal is that most services in public transport will be obliged to be tendered publicly by the orderers of public transit. Only subways, commuter trains, and very small traffic networks are excluded. This means that also the public transit companies which are in communal ownership have to face competition.

Even if this proposal is still under discussion in the various EU bodies, it already has the effect that an increasing amount of public transit is tendered publicly. In such a situation the question arises, in which partitions a public transit network should be tendered. Here two conflicting goals can be identified. On the one hand competition between the operators of public transport is desired, on the other hand the pieces of the network which are tendered

as a unit must be economically reasonable. The size and the shape of the partitions influences these goals. If, for example, single lines are commissioned, the operator has not much potential to build good vehicle schedules and duties. If on the other hand large regions are only operated by a single company, the replacement of the operator is difficult, if not impossible, which may lead to less competition and higher fares.

The tendering problem is described in more detail in Daduna [2001]. A practical approach to take the economical aspects of the partitioning into account would be to select different partitions of the network to be tendered which are reasonable in terms of competitivity, and then to approximate the operation cost of the partitions. The best partition of the network in terms of operation cost and competitivity could then be selected and tendered. In general, the orderers lack the detailed planning know-how of the operators. Therefore integrated optimization tools are helpful that support the orderers to reliably estimate the expenses needed to operate a certain part of the network.

1.9 Vehicle Scheduling

The next problems are operational planning problems. They can be characterized by planning operative resources of a company like vehicles and duties. In this section we give an overview about vehicle scheduling. We will describe the vehicle scheduling problem and the duty scheduling problem in more depth than the other planning steps, because these two are the basis for the main topic of this work, the integrated vehicle and duty scheduling problem.

1.9.1 Description

The goal of the vehicle scheduling problem (VSP) is to find a cost minimal assignment of vehicles to the trips stemming from the timetabling.

Input for the vehicle scheduling are the *timetabled trips* and the possible *deadhead trips* of the vehicles. The timetabled trips are the trips of vehicles that transport passengers. Deadhead trips give the possible concatenation of timetabled trips into *rotations*. The set of timetabled trips and deadhead trips together are simply called *trips*. Each trip has a start- and end-time and a start- and end-location, further we need to know the length and the driving time of each trip.

We use the term deadhead trip, or shortly deadhead, also for the concatenation of two timetabled trips without moving the vehicle. This is different to the terminology of most public transit companies, where these concatenations are called *turns*. Turns regularly occur after the end of a timetabled trip, when the vehicle is stopped for a while and then restarts the next trips with the opposite direction. We have also special deadhead trips called *pull-out trips* which model the begin of a rotation, or *pull-in trips* which model the end of a rotation.

The trips will be assigned to vehicles. The set of available vehicles is called a *fleet*. The maximum number of vehicles used can be a constraint of the VSP or be part of its result. Each vehicle has a unique *vehicle type*. Typical vehicle types are standard bus, double decker, or articulated bus. Each vehicle type has a set of characteristics which is relevant for the planning process, such as the number of seats, an average speed, minimum maintenance intervals, or maximum length of covered distance without refueling. Not all vehicle types are able to service all trips. For instance, long buses cannot go around narrow curves, double deckers may not pass low bridges, or a larger bus is preferred for trips with high passenger volume.

Each vehicle of a fleet is associated with a unique *garage* at a certain location. Each garage contains vehicles of varying types in certain quantities. We call a vehicle type/garage combination a *depot*. We may have a maximum number of vehicles of certain types per garage or in total. These numbers are called *capacities* of the depots or vehicle type capacities.

A *rotation*, sometimes also called block, is an alternating sequence of deadhead and timetabled trips that begins and ends in the same depot. Rotations can be combined to *courses*. A course is a set of rotations that can be driven by a single vehicle. We call a set of courses that covers all timetabled trips a *vehicle schedule*.

The cost of a vehicle schedule is composed by fix cost per used vehicle, cost per driven distance, and cost per time outside of a garage of a vehicle. These costs depend on the vehicle types. In the scenarios known to us, the fixed costs of the vehicles clearly dominate the other operation cost.

1.9.2 Graph Theoretic Model

The vehicle scheduling problem can be described in terms of an acyclic directed network $G_{\mathsf{VSP}} = (V_{\mathsf{VSP}} \cup \{s, t\}, A_{\mathsf{VSP}})$. The nodes V_{VSP} of G_{VSP} are arising by the set of *timetabled trips* plus two additional artificial nodes s

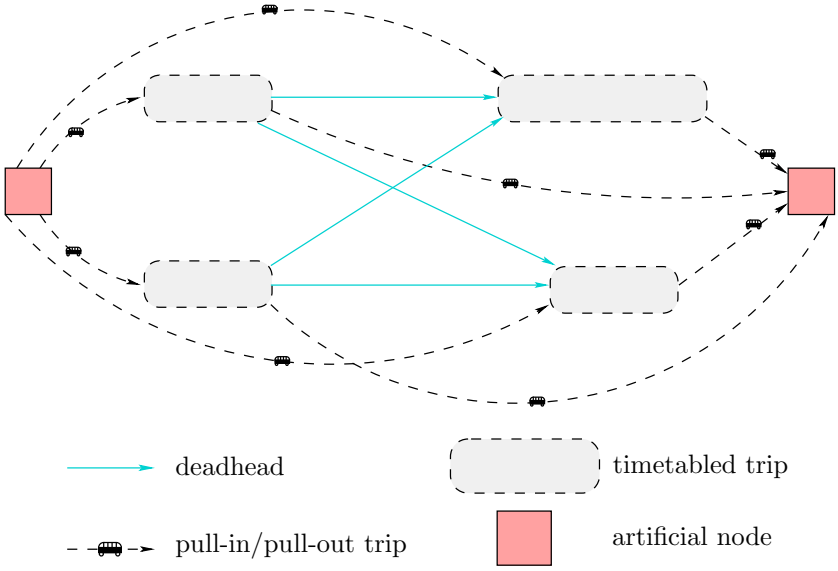


Figure 1.3: Vehicle scheduling graph

and t, which represent the beginning and the end of courses; s is the source of G_{VSP} and t the sink. The arcs A_{VSP} of G_{VSP} represent the *deadheads*, the arcs that emanate from the source s correspond to beginnings of courses, those entering the sink t to endings. An example of such a graph is shown in Figure 1.3. An alternative graph model (see, e.g., Lamatsch [1992]) introduces *timelines*. These are paths build by arcs and nodes that symbolizes the standing of a vehicle at a garage between to rotations. Timelines help to reduce the size of the planning graph under the drawback that certain restrictions on courses are not modelable.

Associated with each deadhead a is a depot $g_a \in \mathcal{G}$ from some set $\mathcal{G}$ of depots. A depot is a unique tuple of vehicle type and garage. The cost of an arc a is denoted by $d_a \in \mathbb{R}$.

There may be parallel arcs in G_{VSP} with different depots and costs. We denote by $A^g_{\mathsf{VSP}} := \{a \in A_{\mathsf{VSP}} : g_a = g\}$ the set of deadheads that can be covered by a depot $g \in \mathcal{G}$, by $\delta^{\text{in}}_g(v) := \delta^{\text{in}}(v) \cap A^g_{\mathsf{VSP}}$ the set of arcs going into node v, restricted to arcs in A^g_{VSP}, and by $\delta^{\text{out}}_g(v) := \delta^{\text{out}}(v) \cap A^g_{\mathsf{VSP}}$ the set of arcs leaving node v, restricted to arcs in A^g_{VSP}.

A course of type $g \in \mathcal{G}$ is an st-path in G_{VSP} that uses only deadheads of type g, i.e., an st-path p such that $p \subseteq A^g_{\mathsf{VSP}}$. The *vehicle scheduling problem* (VSP) is to find a vehicle schedule of minimal cost which complies with the

capacities of the depots.

1.9.3 Integer Programming Model

We formulate the VSP now in terms of integer programming:

$$
\begin{aligned}
&(\mathsf{VSP}) \quad \min\ d^{\mathsf{T}} y, \\
&\qquad \text{s. t.} \\
&\text{(i)} \qquad y(\delta_g^{\text{in}}(v)) - y(\delta_g^{\text{out}}(v)) = 0, && \forall v \in V_{\mathsf{VSP}}, g \in \mathcal{G}, \\
&\text{(ii)} \qquad y(\delta^{\text{in}}(v)) = 1, && \forall v \in V_{\mathsf{VSP}}, \\
&\text{(iii)} \qquad y(\delta^{\text{out}}(v)) = 1, && \forall v \in V_{\mathsf{VSP}}, \\
&\text{(iv)} \qquad y(\delta_g^{\text{out}}(s)) \leq k_g, && \forall g \in \mathcal{G}, \\
&\text{(v)} \qquad y \in \{0,1\}^{A_{\mathsf{VSP}}}.
\end{aligned}
$$

The variables $y_a \in \{0,1\}$, $a \in A_{\mathsf{VSP}}$ are 1 if a is used in the solution and 0 otherwise. Constraints (i) ensure that every block uses only trips of the same depot. Constraints (ii) and (iii) can either be interpreted as set partitioning constraints, because they take care that every timetabled trip is covered by exactly one arc starting or ending at it. Or they can be interpreted as flow conservation constraints combined with a minimum and maximum capacity of one per timetabled trip. One of the sets of constraints (ii) and (iii) is redundant. However, both will become important in certain relaxations of (VSP). Constraints (iv) enforce the capacities k_g on the number of vehicles of a certain type at a certain garage defined by the depot g.

This ILP models a multi-commodity-flow problem (see Section 1.3) with additional capacity constraints. Each depot corresponds to one commodity.

1.9.4 Algorithms

The main problem of solving the VSP for bus traffic is the large number of possible deadheads. In practice, many of the deadheads consist of a pull-in trip, a stopover at a parking facility or a depot (called *standing time*), and a pull-out trip. Such deadheads are called *long arcs* or *pull-in-pull-out trips* in the literature. The number of these long arcs can in the worst case be $n(n+1)/2$, if n is the number of timetabled trips, since every timetabled trips can be connected to every other subsequent timetabled trip. The largest instance in the literature solved to optimality with respect to the number of

vehicles up to now has 28,000 timetabled trips and millions of deadheads (Löbel [1997b]).

There are two methods proposed in the literature to cope with the large number of arcs: In Lamatsch [1992] and in the recent publications Kliewer et al. [2004] a time-line model is proposed, which is based on the single depot model in Desrosiers et al. [1982]. In the time-line model a long arc is represented by a path which consists of the pull-in-trip, a path on a time-line modeling the standing time, and the pull-out trip. This model has the drawback that it is, e.g., not possible to restrict the maximal length of long arcs or it is also not possible to vary the cost for long arcs that should be preferred or avoided. The second approach is to generate the long arcs dynamically in the process of solving the multi commodity flow problem, as proposed in Löbel [1997b]. This approach can be seen as a column generation (here also called arc generation) approach.

Solution methods for large real-world instances of the vehicle scheduling problem are either based on Lagrangian relaxation heuristics (see Löbel [1997b]) or by heuristic preprocessing and solving the resulting problem by standard MIP solvers as in Kliewer et al. [2004]. In the more theoretical oriented articles of Fischetti et al. [2001] and Hadjar et al. [2001] the polyhedral structure of the flow and a path based model of the VSP are examined. An extensive literature survey of the VSP until 1997 can be found in Löbel [1997a].

1.9.5 Algorithm VS-OPT

We use the algorithm of Löbel [1997b] called VS-OPT to solve the VSP occurring as a subproblem of the ISP. It works in two stages: At first we relax all conditions of (VSP) concerning depots. I.e., we solve (VSP) without conditions (i) and (iv). We call this relaxation one-depot-relaxation because the remaining problem is again a vehicle scheduling problem, but all trips are belonging to the same (artificial) depot. This relaxation yields a min-cost-flow-problem and can be solved in polynomial time. In our case we use the network simplex min-cost-flow solver MCF described in Löbel [1996][4]. The solution of the one-depot-relaxation gives us a lower bound on the cost of the multi-depot problem.

In the second stage an assignment of timetabled trips to depots is found by a tabu search. In each iteration of this tabu search an assignment of the

[4]freely available at the URL `http://www.zib.de/Optimization/Software/Mcf/` (for academic purposes)

depots G to the timetabled trips V_{VSP} is found. Thus, we get a partition of V_{VSP} by the sets $V^g_{\mathsf{VSP}} := \{v \in V_{\mathsf{VSP}} \mid g_v = g\}$, for all $g \in G$. This defines for every depot $g \in G$ a subnetwork induced by V^g_{VSP} which defines a single depot problem that can again be solved by a min-cost-flow solver. If every subproblem has a feasible solution, all these solutions together form a solution of the original VSP. This solution gives us an upper bound on (VSP). In general we are able to find a solution which uses the same number of vehicles as calculated in the relaxation. Also the upper bound of (VSP) found in the tabu search has in most cases a gap below 5% to the lower bound given by the one-depot-relaxation. At this, we have to mention that the dominating factor in our objective function is the number of vehicles. I.e., the pullout-trips have significant higher cost than other arcs (about 20 times higher).

The main problem in solving the different min-cost-flow problems occurring in VS-OPT is the huge number of deadheads and, thus, the number of arcs in the planning graph G_{VSP}. The main portion of these arcs arises by arcs which represents the connection of two trips by a pull-in-trip of a vehicle, a waiting time at a depot, and a subsequent pull-out-trip. We call these kind of arcs *pull-in-pull-out trip*. These arcs are generated dynamically if needed in VS-OPT in a kind of column generation. This allows a maximum of flexibility to model the pull-in-pull-out trips. It is, e.g., possible to forbid line changes for certain lines of specific depots by simply not generating the respective trips or, alternatively, to make them more expensive than deadheads that do not imply line changes.

For more details of the algorithm we refer the reader to Löbel [1997b].

The largest instance of (VSP), that we know of, arises in scheduling all buses of the public transit carrier BVG in Berlin. It has about 50 commodities, 28,000 nodes, and millions of arcs. This instance was solved by the algorithm VS-OPT described in Löbel [1997a] to optimality. Since the BVG is one of the largest carriers in the world we assume that most of the vehicle scheduling problems (without consideration of further constraints) from real world applications can be solved by VS-OPT. VS-OPT is used as a subalgorithm of our integrated duty and vehicle scheduling solver.

1.9.6 Integration

If several days of operation should be planned at once, additional constraints to the VSP arise. Then tank stops or maintenance checks of the vehicles have

to be considered after covering a certain distance. Sometimes also a uniform distribution of the traveled distance per vehicle is required. In Gintner et al. [2004], among other aspects of vehicle scheduling, maintenance checks are examined. Models and algorithms for these types of problems can also be found in airline scheduling, see Yu [1997].

The planning of subcontractors, i.e., the decision which timetabled trips should be operated by vehicles and drivers of the company itself and which should be outsourced to subcontractors, demands restrictions on the sum of the driven distance of all vehicle schedules of a certain depot. This is, e.g., needed to model subcontractors with a minimal guaranteed mileage.

The vehicle scheduling problem can be partially integrated with the timetabling problem by allowing to shift timetabled trips. This is known as *trip shifting* or *sensitivity analysis*. Trip shifting gives additional degrees of freedom and therefore in general allows better vehicle schedules. Only few publications about trip shifting are known to us: These are Völker & Schütze [1995], Desaulniers et al. [1998], and Kliewer & Mellouli [2002]. Models and computations for it can also be found in the master's thesis Bönisch [2006]. Trip shifting algorithms are integrated in commercial planning tools for public transit such as Microbus[5] of the IVU AG or Hastus[6] of Giro Inc. However, trip shifting neglects waiting times of passengers or periodic timetables.

Computational studies (Borndörfer [2006]) show that full integration of (aperiodic) timetabling and vehicle scheduling for single lines is possible.

1.10 Duty scheduling

After the determination of the vehicle schedules, drivers have to be assigned to the vehicles. This is done in the duty scheduling planning step and will be described here.

1.10.1 Description

The duty scheduling problem (DSP) consists of finding *duties* that cover a set of mandatory *tasks* as efficient as possible. This set of tasks stems in general from the timetabled and deadhead trips used in the vehicle schedules

[5] see `http://www.ivu.de`
[6] see `http://www.giro.ca`

and eventually also from other activities of drivers, such as being on call or doing administrative work.

A duty is a set of tasks that has to comply to certain rules. In our model each mandatory task has to be covered exactly once, in other models it is also allowed to cover tasks more than once (this can be interpreted as a second driver, who uses a bus as a passenger). Besides the mandatory tasks there are activities which depend on the structure of a duty such as the so called *sign on* and *sign off times*. They are, e.g., necessary to prepare a vehicle before or after the change of a driver. Also obligatory driver breaks have to be planned. In general, the breaks can be taken in the vehicle when it is parking or at certain locations with facilities where the drivers can rest. The rules to which a duty has to comply are, for instance, restrictions on the duration of a duty or on the position and duration of breaks. A set of basic rules is explained in Section 5.2, in practice variations and specializations of these rules are used.

Global constraints on subsets of the duties may also exist, such as a maximum number of used split duties or minimum and maximum average paid times over certain duties in a solution. Such constraints are called *base-constraints*. Sometimes deviations from these minima and maxima are allowed within certain limits, however, these deviations are penalized and increase the overall cost of the duty schedule.

The cost of a single duty depends mainly on the paid time. Additionally we may have a fix cost per duty or penalties for the deviation of paid times or duty durations from certain targets per duty. A detailed description of the cost of a duty can be found in Section 5.2.5.

1.10.2 Graph Theoretic Model

The basic structure of the duty scheduling problem is modeled by an acyclic network $G_{\mathsf{DSP}} = (V_{\mathsf{DSP}} \cup \{s, t\}, A_{\mathsf{DSP}})$. The nodes of G_{DSP} consist of a set of nodes V_{DSP} representing tasks that can be performed by drivers and two additional nodes s and t, which mark the beginning and the end of duties; again, like in the vehicle scheduling case s is the source of G_{DSP} and t the sink. A task v is in general either part of one timetabled trip in the set V_{VSP}, or of one deadhead trip in A_{VSP}, or it may be a *supplementary task* independent of the vehicle schedule. The supplementary tasks model, for example, sign-on and sign-off times, walking times of a driver, or break times outside of a vehicle. Sometimes tasks also arise by activities that are not

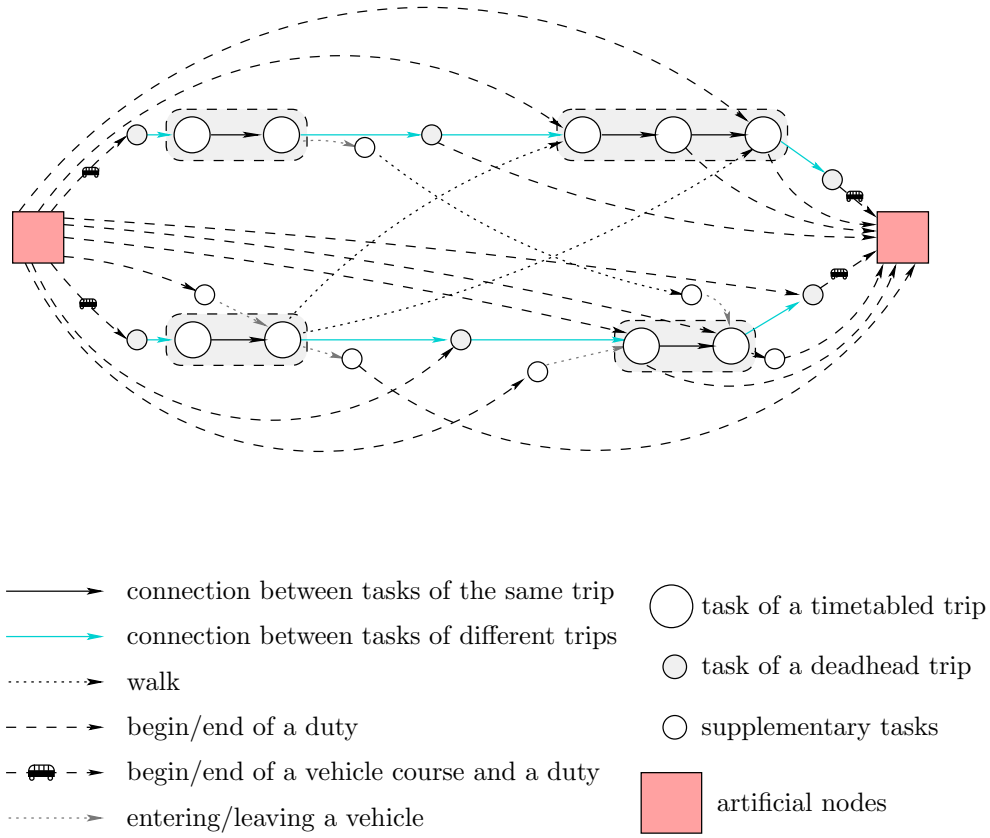


Figure 1.4: Duty scheduling network

related to driving, this occurs only rarely and we leave out this case in our model for the sake of simplicity. We assume that there is at least one task associated with every timetabled trip and every deadhead trip; these tasks correspond to units of work that have to be performed by a single driver without interruption. Several tasks for the same trip indicate that this trip has relief opportunities where a driver can be replaced. We denote the tasks corresponding to a (timetabled or deadhead) trip t by $V_{\mathsf{DSP}}(t) \subset V_{\mathsf{DSP}}$. Often the relief opportunities are bound to certain locations such as depots or hubs of a network. Locations with relief opportunities are often called *relief points*.

The arcs A_{DSP} of G_{DSP} are called *links*; they correspond to possible concatenations of tasks in duties or to beginnings or endings of duties. An example of a duty scheduling graph is given in Figure 1.4. Every *duty* p is an st-path in G_{DSP}, but not every st-path is also a duty, because each duty has to obey various – sometimes very complex – rules.

More details about duties and duty generation can be found in Chapter 5.

1.10.3 Integer Programming Model

Our approach to the duty scheduling problem (DSP) is a path based model (see Section 1.3.2) and can be seen as a set-partitioning problem with additional base-constraints over the set of mandatory tasks: We want to find a set of duties, which implies a cost minimal disjoint partition of all mandatory tasks.

We model DSP as the following integer program denoted by (DSP): Let D be the set of all duties. Let $\bar{V}_{\mathsf{DSP}}$ be the set of mandatory tasks. A duty $p \in D$ is characterized by the set of mandatory tasks it uses, denoted by $V(p) \subset V_{\mathsf{DSP}}$. The cost of a duty p is denoted by c_p. Let $A \in \{0,1\}^{\bar{V}_{\mathsf{DSP}} \times D}$ be the task-duty incidence matrix. That is, A_{tp} is 1 if task $t \in V(p)$ and zero otherwise. Let $\mathcal{B}$ be the set of base-constraints and $R \in \mathbb{R}^{\mathcal{B} \times D}$ is the coefficient matrix of the base-constraints:

$$
\begin{array}{lll}
(\mathsf{DSP}) & \min c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z, & \\
& \text{s.t.} & \\
\text{(i)} & Ax & = \mathbb{1}, \\
\text{(ii)} & Rx - z & \leq r, \\
\text{(iii)} & x & \in \{0,1\}^D, \\
\text{(iv)} & & z \geq 0.
\end{array}
$$

The variable x_p, $p \in D$, is one if duty p is used in the duty schedule and zero otherwise. The slack variables z_b, $b \in \mathcal{B}$, give the deviation of base constraint b from its target r_b upwards. The tasks in $\bar{V}_{\mathsf{DSP}}$ may include deadheads used by a fixed vehicle schedule. The set of feasible duties D uses only deadheads used by this vehicle schedule.

(DSP) (or its set covering variant) is the model which is used in most publications about duty scheduling. Duty scheduling problems of airlines and railways are often formulated as set covering problems. This allows to consider passenger rides of drivers implicitly as overcoverings of certain tasks and therefore keeps the model compact. In duty scheduling for public transit this is often not wanted, because passenger rides are only allowed for certain connections or not at all because bus trips are vulnerable to delays.

In the literature also flow oriented models of the DSP can be found in, e.g., Friberg & Haase [1997]. In our applications the number of constraints

on feasible duties and the difficulties to formulate them as linear constraints makes it inappropriate to use such kind of model.

1.10.4 Algorithms

Basically, in the recent literature the DSP is solved by column generation approaches and various set-partitioning- or set-covering-heuristics. The approaches differ in the methods to solve the LPs that occur in the column generation, in the models of the pricing problems and their solution approaches, and in the used heuristics to find integral solutions.

To our knowledge, the first approach to solve duty scheduling problems in public transit with a set-partitioning/column generation-approach was Desrochers & Soumis [1989]. Further examples of such duty scheduling algorithms are Desrochers et al. [1992] for bus transit (integrated in the Hastus Crew-Opt System), Andersson et al. [1998] for airline cew scheduling (integrated in the Carmen System), and Kroon & Fischetti [2000] for railway transit (integrated in the Turni System). An extensive survey about duty scheduling can be found in Ernst et al. [2004].

Our algorithm DS-OPT to solve DSP, which is based on the work of Borndörfer et al. [2003], relies heavily on the MIP (DSP): it generates (DSP) (or at least an approximation of it), solves its LP-relaxation by the bundle method, and uses the information of the LP-relaxation in a heuristic to finally find a solution of (DSP). DS-OPT is integrated in the Microbus Planning System for public transit of the IVU AG and also an important subroutine of our integrated vehicle and duty scheduling approach.

1.10.5 Integration

Integration of vehicle and duty scheduling is already discussed in some publications; real world applications of it have only recently emerged. A literature survey and an overview of our algorithm IS-OPT that solves the integrated duty and vehicle scheduling problem can be found in Section 2. In the subsequent chapters important parts of IS-OPT will be described and computational results achieved by it will be reported.

1.11 Rostering

1.11.1 Description

After the completion of the duty schedule a set of anonymous duties for a fixed planning horizon is determined. These duties have to be assigned to individual drivers. This step is called *rostering* and a specific assignment is called *roster*. A roster has to satisfy various rules stipulated by laws and agreements with trade unions. In the European Union the most important rules deal with minimum rest periods between shifts, maximum shift times, maximum weekly driving times, and minimum weekly rest periods. Additional conditions on the rosters may arise by requests of drivers, e.g., an assignment to a certain duty or a specific type of duty at a certain date.

In some publications the rostering is split into a rostering and a crew assignment step. Then the rostering consists of generating anonymous sequences of duties that have to be assigned in a second step to individual drivers.

Drivers can not be handled uniformly with respect to the rostering problem because they may have a different history concerning rest periods, due to their different assignments in previous planning periods. E.g., a driver which had a late duty at the last day of the previous planning period may not begin the new planning period with an early duty because this violates the requirement of a minimum rest period between two duties. Moreover, there are in general groups of drivers with different labor contracts, e.g., stipulating different weekly working times. Also the local knowledge or the qualifications to drive certain vehicles may differ.

The objective of rostering step depends on the working time model of the public transit company. If drivers have fixed working time, an important objective of the rostering is to find a cost minimal roster, with respect to overtime or paid time. In companies with flexible working time, it suffices to produce rosters with certain average working times for groups of drivers because here deviations of the contractual working time can be compensated in later periods. Another goal is to maximize the satisfaction of the drivers regarding wishes about the kind of duties and the time of the rest periods. Sometimes also only the problem to find a feasible solution of the rostering problem is considered.

In Germany and in Switzerland it is common practice to create a roster by sorting duties into a fixed pattern of duty types, which is called *rota*. An example of a rota can be seen in Figure 1.5. Such a rota may be stipulated by agreements with trade unions or the workers council of the company. If

	Mon	Tue	Wen	Fri	Thu	Sat	Sun
1	late	late	day	day	early	early	
2		late	late	day	day	early	early
3			late	late	day	day	early
4	early			late	late	day	day
5	early	early			late	late	day
6	day	early	early			late	late
7	day	day	early	early			late
8	late	day	day	early	early		

Figure 1.5: Example of a rota with six workdays followed by two days off

a rota is used, the duty scheduling planning step must produce the duties of different duty types in the right ratio for the rotas.

1.11.2 Model

The model of the rostering problem is very similar to the duty scheduling problem. The duties correspond to the tasks of the DSP. Then we can again create a planning network and formulate rules for feasible paths in this network that corresponds to roster. These roster are corresponding to the columns of a duty-roster-coincidence-matrix of a set-partitioning problem. Additional constraints may arise by qualifications of drivers needed to perform certain rosters. So maybe there is only a limited number of drivers that is allowed to drive a certain vehicle or the drivers only have knowledge of certain routes and are not allowed to driver other ones. Also predetermined rotas can be considered in such models by additional constraints.

1.11.3 Algorithms

There is an extensive literature about the rostering problem in airline traffic, see Kohl & Karisch [2004] for a literature survey and the description of the model and algorithm of the Carmen Crew Rostering system. There are also some publications in rostering for train companies, e.g., Caprara et al. [1998]; Ernst et al. [2001]. An exhaustive literature survey about duty scheduling and crew rostering in general can be found in Ernst et al. [2004]. In general, column generation and LP-techniques are used. Literature about crew rostering for bus traffic or public transport in general is sparse. A recent technical report about a crew rostering system for public transport was presented at the CASPT 2000 by Emden-Weinert et al. [2001]. In this article the crew rostering problem is solved by genetic algorithms.

1.11.4 Integration

Duty rostering can be integrated with duty scheduling. This is equivalent to using longer time horizons than one day in the duty scheduling problem. Then additional rules stemming from rostering have to be considered, such as minimum rest periods between duties and sufficient days off. However, this makes the already difficult pricing problem of the duty scheduling problem considerably harder for two reasons: On the one hand additional rules increase the complexity inherently, on the other hand the support of each column becomes on the average larger with longer planning horizons.

Also the size of the planning problems and the number of potential duties increases. In contrast, the size of solvable duty scheduling problems is already limited to ones arising by large depots or medium sized public transport companies because of the exhaustion of the computer memory.

1.12 Conclusion

We have presented an overview of mathematical models and algorithms of problems that come up in the planning process in public transport. These methods are in different stages of practical application. Some, such as the network design problem, are only theoretically examined, others, such as vehicle and duty scheduling, are integrated in commercial software and are in daily use in public transport companies. Especially the operational planning steps have reached a relatively mature stage, but we think that faster computers with more memory and new methods are enabling further improvement of the support of the planning process by mathematical methods. We have pointed out, where this can be done by more detailed or extensive modeling of the planning problems, such as looking in the timetabling problem not only at certain standard planning periods, but including the transitions between peak hours and time of low traffic.

Another field of improvement is the enlargement of the solution space by integrating subsequent planning steps as in the integration of

- network design and line planning,
- line planning and passenger route selection,
- time tabling and vehicle scheduling,

- vehicle scheduling and duty scheduling, and
- duty scheduling and rostering.

The integrated vehicle and duty scheduling will be examined in detail in the remainder of this thesis.

Chapter 2

Integration of Vehicle and Duty Scheduling

This chapter discusses approaches to and advantages of integrated vehicle and duty scheduling in comparison to traditional sequential planning.

The chapter is structured as follows: Section 2.1 analyzes the importance of efficient vehicle and duty schedules. We show that integrated planning can help to improve the overall efficency of vehicle and driver schedules. Section 2.2 describes two levels of integration. These are used to analyze the literature about integrated vehicle and duty scheduling in Section 2.3. In Section 2.4 our approach to integrated vehicle and duty scheduling is sketched.

2.1 Motivation

The main motivation for integrated vehicle and duty scheduling comes from regional scenarios where traditional sequential approaches do not lead to feasible schedules. This is discussed in Section 2.1.1. In Section 2.1.2 we motivate that also in other scenarios integrated scheduling can increase the overall planning efficiency. Finally, Section 2.1.3 quantifies the cost of vehicles and personnel of typical German public transport companies to illustrate the potential benefits of an integration of vehicle and duty scheduling.

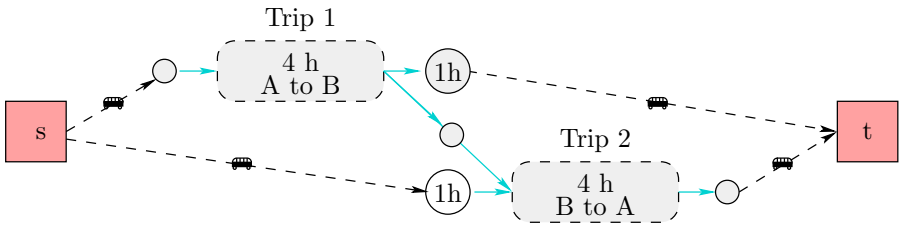


Figure 2.1: Regional Scheduling

2.1.1 Regional Public Transit

The main argument for integrated scheduling is that sequential approaches often do not work in regional public transit. This section shows the cause of this fact.

In comparison to urban public transit, regional carriers use few relief points outside the garages. Sometimes a driver may even start or end pieces of work only at garages. Additionally, trips are in general longer than trips in an urban context. Traditional vehicle-first-duty-second planning often results in vehicle rotations without possibility to grant drivers the necessary breaks or without possibility to relieve a driver before the maximum driving time is reached. Eventually, no feasible duty schedule can be constructed using this approach.

An example for such a situation is illustrated in Figure 2.1[1]. It shows a small planning graph that consists of two timetabled trips denoted by Trip 1 and Trip 2. One of these trips starts at location A and ends at location B, the other trip goes the opposite direction from B to A. The duration of the trips is four hours each. The connections from A to B and vice versa require only one hour of driving time (perhaps the timetabled trips do not take the shortest route). The other connections have a duration of zero, that is, Trip 1 begins at the depot, Trip 2 ends there, and the end of Trip 1 is at the same location and time as the beginning of Trip 2. Feasible vehicle rotations and duties consist of paths from node s to t. We assume for this example that the only condition on the feasibility of duties is that they do not exceed a duration of seven hours.

The fleet minimal vehicle schedule uses only one vehicle. It consists of the rotation that starts at the depot, performs Trip 1, then Trip 2, and goes back to the depot. However, no feasible duty exists for this rotation because

[1]The symbols used in this figure are explained in Figure 1.4.

$$\eta_v := \frac{\text{time in timetabled trips}}{\text{driving time}}$$
$$\eta_d := \frac{\text{driving time}}{\text{paid time}}$$
$$\eta_t := \frac{\text{time in timetabled trips}}{\text{paid time}}$$

Figure 2.2: Indicators of operational planning

the driver would have to work for eight hours. The only feasible solution for this scheduling problem consists of two vehicle blocks. The first one covers only Trip 1 and the needed deadheads, the second one the other trip and the appropriate deadheads. These blocks can be assigned to two duties that have a length of five hours each. Obviously this solution can not be found by a vehicle-first-duty-second approach.

2.1.2 Vehicle and Duty Schedule Efficiency

A method to measure the quality of vehicle and duty schedules besides actual costs are performance ratios. These are quotients of certain characteristics of the schedules, which are often used to compare the efficiency of different public transit companies in industry benchmarks.

The quality of the vehicle schedule can be measured by the ratio η_v between the total time of timetabled trips and the total time of vehicles out of depot. This ratio is called the *vehicle schedule efficiency*. The *duty schedule efficiency* is the ratio η_d between the driving time (which is equal to the time the vehicles are driven) and the total paid time. Finally, the *total planning efficiency* $\eta_t (= \eta_v \cdot \eta_d)$ is the product of these two ratios. Thus, η_t is the ratio between the total time on timetabled trips and the total paid time. These efficiencies are summarized in Figure 2.2.

These ratios can be used to illustrate basic dependencies between vehicle schedule efficiency, duty schedule efficiency, and the total efficiency. Figure 2.3 gives an example for a real company. Each of the lines in the figure represents possible combinations of vehicle and duty scheduling efficiencies, which lead to a total efficiency of 55%, 60%, or 65% respectively. One can see that a bad vehicle schedule efficiency can be compensated by a good duty schedule efficiency and vice versa. E.g., it is possible to build high efficiency duties by building vehicle schedules including turning times that can

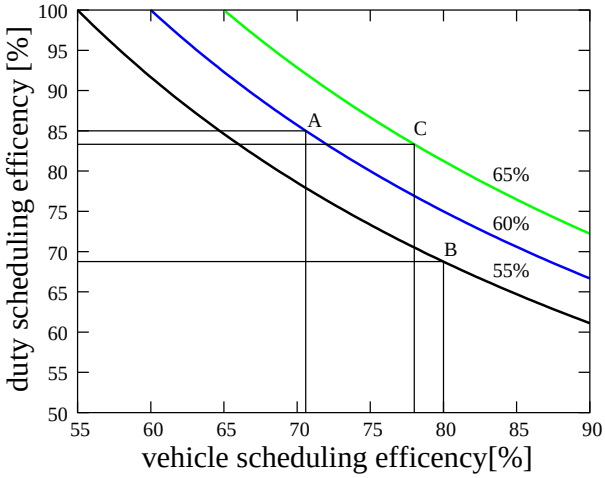


Figure 2.3: Vehicle scheduling efficiency vs. duty scheduling efficiency

be used by the drivers to take their breaks. This situation is represented by point A in the figure 2.3. On the other hand, the reduction of turning times may force the planners to schedule more breaks for the drivers at locations with rest facilities. And the additional time needed by the drivers to get to these facilities worsens the duty scheduling efficiency (point B). In general, the concentration on the improvement of only one of these ratios leads not to an optimal overall efficiency. An integrated approach, which solves the vehicle and the duty scheduling problems at once, is able to improve the total efficiency (point C), even if it produces, from individual points of view, suboptimal vehicle and duty schedules.

2.1.3 Vehicle and Duty Costs

It is in general not easy to come up with cost figures for public transit. However, in Leuthardt [1998] the cost structure for the operation of buses of German public transport carriers is analyzed. In particular, urban and regional public transit companies are compared.

In this article it is stated that urban carriers employ in the average 2.6 drivers and 1.2 other personnel (administration, maintenance) per bus, regional carriers are having per bus 2.0 drivers and 0.6 other personnel. On the average the crew costs of urban carriers amount to about 75% of the total cost and to about 70% for regional carriers. Assuming that drivers and other personnel have roughly the same average income, the crew costs (i.e.,

	driver cost	vehicle cost		sum
		fixed	fuel	
urban carriers	51%	11%	5%	67%
regional carriers	54%	15%	6%	75%

Table 2.1: Vehicle and duty costs according to Leuthardt [1998]

the income of the drivers) accounts for about 50% of the costs of a public transit company.

The fixed costs for buses (amortization and interest on invested capital) is the second largest cost factor with about 11% and 15% of the total costs per bus for urban and regional carriers, respectively. As the next item follows the fuel cost with about 5–6% of the total cost. Thus, the direct costs of vehicles are about 16–21% of the total costs. In Table 2.1 this figures are summarized.

We conclude that the overall costs of drivers and vehicles amount to about 67–75% of the total costs of an average public transit company in Germany. It is therefore crucial for the cost effectiveness of public transport companies to utilize these resources efficiently.

2.2 Approaches to the Integrated Vehicle and Duty Scheduling Problem

We have explained traditional sequential vehicle and duty scheduling in Sections 1.9 and 1.10. In practice and in the literature also other approaches are taken:

1. The traditional approach of *urban carriers* is to first schedule the vehicles and then the duties. We denote this approach by *sequential* scheduling. Sometimes some duty scheduling constraints are considered in the vehicle scheduling step to improve total efficiency or to improve the probability to find a feasible solution. An example for this approach is to allow before or after some timetabled trips only connections with enough break time for drivers to locally fulfill at least one break rule. This, of course, may lead to suboptimal schedules.

2. In *regional traffic* companies usually construct at first duties such that each piece of a duty is also a valid vehicle rotation, and then use these rotations to build vehicle schedules. The rotations are used to construct vehicle blocks by a greedy algorithm. This approach is called *combined vehicle and duty scheduling*. The drawback in comparison to integrated vehicle and duty scheduling is that it is not possible to control vehicles, i.e., the number of vehicles can not be minimized and it is not possible to set limits on numbers of vehicles of certain types or on certain depots.

3. Recently algorithms have emerged that fully integrate the problems of vehicle and duty scheduling and solve them in one step minimizing an objective function that considers vehicle as well as duty related costs.

At first we will give a generalized ILP model for duty scheduling that takes deadheads into account, then we model the combined approach, which can be solved by a slightly modified duty scheduling problem solver. In Section 2.3 we present models and algorithms for integrated optimization from the literature. Finally, in Section 2.4 we describe our algorithm IS-OPT which solves fully integrated vehicle and duty scheduling problems. Important parts of IS-OPT are analyzed in the following chapters, for example the the proximal bundle method in Chapter 4, duty generation in Chapter 5, and the primal heuristic in Chapter 6.

2.2.1 The Link between Duty Scheduling and Vehicle Scheduling

In general the DSP is dependent on a vehicle schedule. So let y^* be an assignment of values to the variables of model (VSP) of Section 1.9 satisfying all constraints of this model. That is, for all deadhead trips $a \in A_{\mathsf{VSP}}$ is $y^*_a = 1$ if deadhead trip a is used by the vehicle schedule and $y^*_a = 0$ otherwise. We will also denote a vehicle schedule by y^*. We define a *duty schedule that is compatible to* a vehicle schedule y^* as: a set of duties that covers all tasks corresponding to timetabled trips exactly once and that also covers exactly those tasks corresponding to deadhead trips that are used by the vehicle schedule y^*. Supplementary tasks only appear in this definition indirectly: They have to be included in single duties to make them feasible but do not matter in the definition of a duty schedule.

The *duty scheduling problem* with a fixed vehicle schedule y^*, denoted by DSP_{y^*} is to find a duty schedule compatible to y^* with minimum cost.

Let $D_v := \{p \in D : v \in V(p)\}$ the set of all duties that contain some task $v \in V_{\mathsf{DSP}}$ and denote by $a(v) \in A_{\mathsf{VSP}}$ the deadhead of which the task $v \in tasks$ is a part of. The problem DSP_{y^*} can be stated as the following (path-oriented) integer program:

$$
\begin{array}{lllll}
(\mathsf{DSP}_{y^*}) & \min & c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z, & & \\
& \text{s. t.} & & & \\
\text{(i)} & & x(D_v) & = 1, & \forall v \in V^T_{\mathsf{DSP}}, \\
\text{(ii)} & & x(D_v) & = y^*_{a(v)}, & \forall v \in V^D_{\mathsf{DSP}}, \\
\text{(iii)} & & Rx - & z \leq r, & \\
\text{(iv)} & & x & \in \{0,1\}^D, & \\
\text{(v)} & & & z \geq 0. &
\end{array}
$$

The variables $x \in \{0,1\}^D$ are one if the corresponding duty is in the solution and zero else. The slack variables $z \in \mathbb{R}^{\mathcal{B}}_+$ denote the deviation of a resource consumption from its target. Equations (i) und (ii) guarantee, that every task, which is part of a trip used in the vehicle schedules, is also used in the duty schedule. At this (i) considers the timetabled trips and (ii) the deadhead trips used in y^*. The constraints (iii) model the base constraints. We denote the set of base constraints by $\mathcal{B}$. The matrix R is an element of $\mathbb{R}^{\mathcal{B}\times D}$ and $r \in \mathbb{R}^{\mathcal{B}}$. D denotes the set of all feasible duties. The vector $c \in \mathbb{R}^D$ specifies the cost of the duties, and $\gamma \in \mathbb{R}^{\mathcal{B}}$ is the vector of the penalties to violate a global requirement on a duty schedule. All entries of c are positive and of p are non-negative.

If its clear which vehicle schedule is used we formulate (DSP_{y^*}) with fixed vehicle schedule y^* as a "pure" DSP and denote its integer program by (DSP) as defined in Section 1.10.

2.2.2 The Combined Approach

In the combined approach we create duties whose pieces of work are also valid rotations. That is every piece of work starts and ends at a depot and uses only a single vehicle without interruptions. These rotations are in a second step merged to a vehicle schedule.

For this approach we can use a duty scheduling algorithm if automatically every piece of work defines also a vehicle rotation. This can guaranteed if every deadhead trip consists of exactly one task, i.e., no relief points are allowed on deadheads, and the planning scenario must either not have more

than one depot and one vehicle type, or otherwise relieves of drivers are only allowed at the depots. These restrictions are due to the fact that otherwise we probably can not construct rotations from the deadheads used in a solution.

For an ILP-model of the combined approach we replace constraints (ii) of model (DSP_{y^*}) by two constraints for each timetabled trip. The new constraints ensure that a vehicle is available at the beginning of each timetabled trip and that it is moved away at the end.

$$
\begin{aligned}
(\mathsf{DSP}_{\mathrm{v}})\ \min\ & c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z \quad , \\
& \text{s. t.} \\
\text{(i)} \qquad & x(D_v) = 1,\ \forall v \in V^T_{\mathsf{DSP}}, \\
\text{(iia)} \qquad & \sum_{a \in \delta^{\mathrm{in}}(v)} x(D_a) = 1,\ \forall v \in V_{\mathsf{VSP}}, \\
\text{(iib)} \qquad & \sum_{a \in \delta^{\mathrm{out}}(v)} x(D_a) = 1,\ \forall v \in V_{\mathsf{VSP}}, \\
\text{(iii)} \qquad & Rx - z \leq r, \\
\text{(iv)} \qquad & x \in \{0,1\}^D, z \geq 0.
\end{aligned}
$$

Here the set $\delta^{\mathrm{in}}(v) \subset A_{\mathsf{VSP}}$ is the set of deadheads which end in trip v and the set $\delta^{\mathrm{in}}(v) \subset A_{\mathsf{VSP}}$ is the set of deadheads which directly follow the trip v. Equations (iia) and (iib) are equivalent to the flow conservations constraints (VSP)(ii) and (iii). They ensure that each timetabled trip has an ingoing and an outgoing deadhead. In the duty generation process we have additionally to consider, that pieces of work are only using trips with a common valid combination of depot and vehicle type. That is, a trip which has to be assigned to a vehicle of type "A" and a trip which have to be assigned to a vehicle of type "B" are not allowed in the same piece of work. This can, e.g., be satisfied by constructing duties separately for each vehicle type. The main drawback of this model is that it is not possible to include fixed cost per vehicle in the objective function. An approach to overcome this problem has been proposed by Haase et al. [2001]. They introduce additional constraints and certain points in time Θ which count the number of vehicles. Their

model (DSP_v) reads

$$
\begin{aligned}
(\mathsf{DSP}_{\mathrm{v}}^{\mathrm{z}}) \quad & \min\ c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z + \delta\xi, \\
& \text{s. t.} \\
\text{(i)} \quad & x(D_v) = 1 && \forall v \in V_{\mathsf{DSP}}^{T}, \\
\text{(iia)} \quad & \sum_{a\in\delta^{\mathrm{in}}(v)} x(D_a) = 1, && \forall v \in V_{\mathsf{VSP}}, \\
\text{(iib)} \quad & \sum_{a\in\delta^{\mathrm{out}}(v)} x(D_a) = 1, && \forall v \in V_{\mathsf{VSP}} \\
\text{(iii)} \quad & Rx - z \le r, \\
\text{(iv)} \quad & x(D_t) \le \xi, && \forall t \in \Theta, \\
\text{(v)} \quad & x \in \{0,1\}^{D}, z \ge 0, \xi \in \mathbb{N}.
\end{aligned}
$$

The variable z counts the number of vehicles and the real number δ is the fixed cost per used vehicle. The set Θ is the set of points in time, at which timetabled or deadhead trips are starting. The sets $D_t \subset D, t \in \Theta$ are the sets of duties, which contain a task, which includes driving a vehicle at time t. The computational results in Haase et al. [2001] seem to indicate that this extension makes the problem considerably harder to solve.

If there is no possibility of changing a vehicle outside of a depot each duty consists of pieces of work that are simultaneously vehicle rotations. Thus, this model also is able to solve problem instances with more than one vehicle type and depot, because it can be guaranteed that each feasible piece of work starts and ends at the same depot and only consists of trips which are valid for at least one common vehicle type. Under these circumstances for every solution to model (DSP_v) (even without constraints (iia) and (iib)) vehicle schedules can be constructed by the pieces of work of the duty schedule by, e.g., a greedy algorithm. This approach is described in Hanisch [1990]. Its advantage is that this problem can be solved with existing algorithms for the DSP and VSP.

In a general context with multiple depots or vehicle types and relief points outside depots combined scheduling can still be useful because the optimal solution of model (DSP_v) produces a lower bound for the cost of the duty schedules in an integrated optimization problem, and it is in most cases significantly better than the optimal value of model (DSP_v) without equations (iia) and (iib). We exploit this connection in our integrated approach.

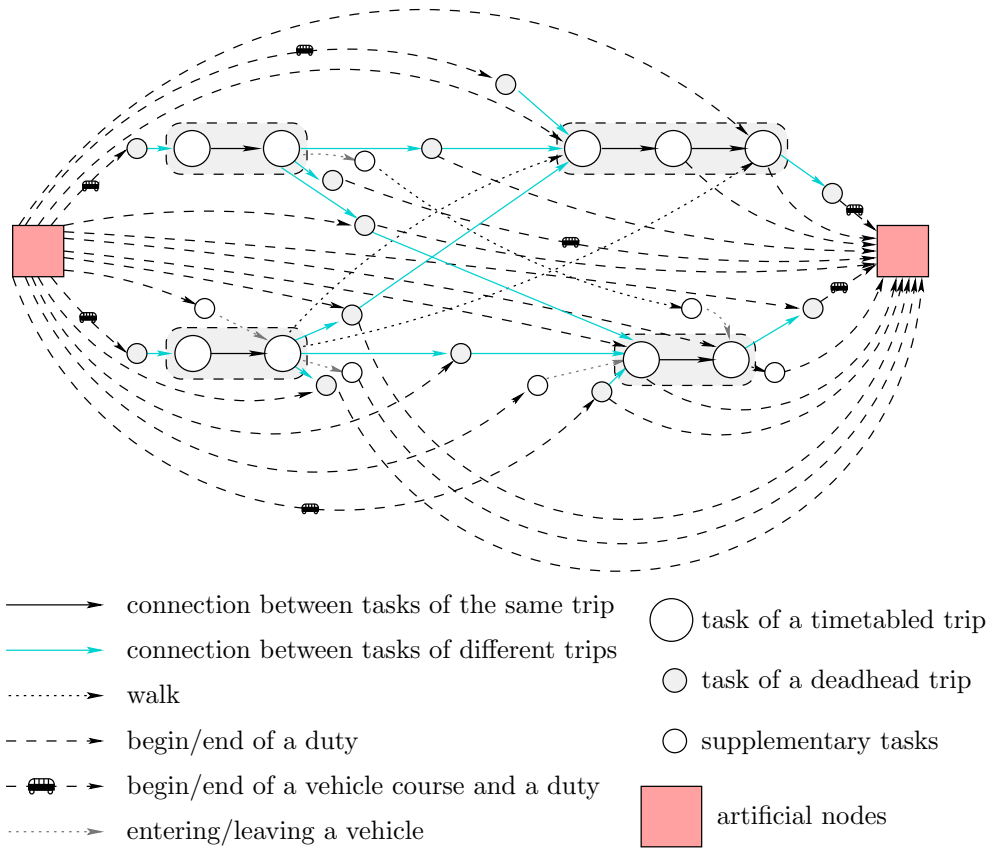


Figure 2.4: Integrated vehicle and duty scheduling graph

2.2.3 Full Integration of Vehicle and Duty Scheduling

The *integrated vehicle and duty scheduling problem* or shortly ISP is to simultaneously construct a vehicle schedule and a compatible duty schedule of minimum overall cost.

The graph model of ISP is very similar to the model of the DSP. It only additionally includes all possible deadheads and the corresponding walks of the drivers to and from these, not only the ones used by a certain vehicle schedule. An example of such a graph can be seen in Figure 2.4. These graphs can become very large for the same reasons as the vehicle scheduling graphs of Section 1.9. We will treat this topic in more depth in Chapter 5.

Introducing suitable constraint matrices and vectors, the model (ISP)

reads:

$$
\begin{aligned}
&(\mathsf{ISP}) \quad \min\ c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z + d^{\mathsf{T}}y, \\
&\qquad \text{s.t.} \\
&\text{(i)} \qquad Ax \qquad\qquad\quad = 1, \\
&\text{(ii)} \qquad Rx - \quad z \qquad\quad \leq r, \\
&\text{(iii)} \qquad\qquad\qquad Ny = b, \\
&\text{(iv)} \qquad Bx \quad - \quad My = 0, \\
&\text{(v)} \qquad x \in \{0,1\}^{D}, \quad z \geq 0, \quad y \in \{0,1\}^{A_{\mathsf{VSP}}}.
\end{aligned}
$$

In this model, the *multi-commodity-flow constraints* (ISP) (iii) correspond to the vehicle scheduling constraints (VSP) (i)–(iii); they generate a feasible vehicle schedule. The *(timetabled) trip partitioning constraints* (ISP) (i) and (ii) are exactly the duty scheduling constraints (DSP_{y^*}) (i) and (iii); they make sure that each timetabled trip is covered by exactly one duty and that the base constraints are satisfied. Finally, the *coupling constraints* (ISP) (iv) correspond to the duty scheduling constraints (DSP_y) (ii); they guarantee that the vehicle and duty schedules x and y are synchronized with respect to the deadhead trips, i.e., a task of a deadhead trip is either assigned to both a vehicle and a duty or to none. So we define $B \in \{0,1\}^{V^D_{\mathsf{DSP}} \times D}$ as follows: B_{td} is one, if task t of a deadhead is used by duty d and zero otherwise. And $M \in \{0,1\}^{V^D_{\mathsf{DSP}} \times A_{\mathsf{VSP}}}$ is the incidence-matrix of tasks on deadheads. That is, M_{td} is one if task t is used by deadhead d and zero otherwise.

The model (ISP) is a generalization of a model used by Freling and coauthors, see Freling [1997]. There it is assumed that every deadhead consists of exactly one task, and also constraints (ii) are not considered. Also minimum and maximum capacities on vehicles are an addition of our model.

2.3 Literature

The literature on integrated vehicle and duty scheduling (ISP) is comparably scant. One reason for this is that instances of ISP of practical relevant size and complexity became tractable only a few years ago.

The existing articles on the topic differ in terms of the complexity of the rules for vehicle and duty schedules, of the ILP-models (mainly flow or set partitioning/covering models), in the degrees of integration, and in the used algorithms. The following survey examines the literature with special attention to these criteria.

A survey of integrated approaches to duty and vehicle scheduling until 1997 can be found in Gaffi & Nonato [1997].

2.3.1 Ball, Bodin and Dial

The first article on the ISP that we are aware of was published in 1983 by Ball et al. [1983]. They describe an ISP at the Baltimore Metropolitan Transit Authority (MTA) and develop a mathematical model for it.

The model consists of two (directed and acyclic) networks with identical sets of vertices, one for the vehicle scheduling problem and the other one for the duty scheduling problem. Duties and vehicle schedules are paths in these graphs that have to have certain properties. The compatibility of the duty and the vehicle schedules in a solution is ensured by certain constraints. The networks are essentially the same as the ones presented in Sections 1.9 and 1.10.

The cost function is complex. It is based on the working time in a duty, but also considers minimum paid times and penalties for excessive short or long duties or duties with more than one break. Additionally the number of split duties, i.e., duties with a break of more than three hours, must be less than the number of so called straight runs, i.e., duties that are not too short and also do not have a long break.

This problems was solved by a sequence of matching and local improvement heuristics. From today's perspective this method resembles more a combined duty and vehicle scheduling approach than a fully integrated algorithm. This approach was mainly designed to be memory efficient because this was the main bottleneck in the year 1980 when the paper was submitted. Ball, Bodin, and Dial say: "It would be unreasonable to expect to store a graph with more than 10 or 30 thousand edges in the core of a modern day computer while executing a scheduling code." Today our largest solvable instance of ISP has about 2 Million edges in the planning graph.

2.3.2 Vehicle Scheduling Centered Approaches

We now present approaches that enhance vehicle scheduling algorithms to take duty scheduling into account.

Scott [1985] designed a heuristic on base of the vehicle scheduling optimizer (see Desrosiers et al. [1982]) and the duty scheduling optimizer (see

Blais & Rousseau [1982]) which were part of the planning system HASTUS at that time. His approach works as follows: At first he solves the VSP, then he tries to improve the resulting vehicle schedule such that the cost of the best duty schedule based on this vehicle schedule decreases. Here the crew cost are approximated using a certain relaxation of the DSP, called HASTUS-Macro which ignores locations and round start and end times of tasks. In the next step the vehicle schedules are adjusted to diminish the lower bound on the DSP calculated by HASTUS Macro by a kind of 2-Opt heuristic. No method to generate feasible duty schedules was mentioned. The method was tested on single lines of the transit authority in Montréal with about 54 vehicles and 57 duties at most.

Darby-Dowman et al. [1988] present another vehicle scheduling centered approach. They describe a method that calculates vehicle schedules on single routes such that "crewing considerations are taken into account". This method also determines duty schedules. No details about the algorithm or possible restrictions on vehicle and duty schedules are reported.

2.3.3 Duty Scheduling Centered Approaches

We now review approaches that concentrate on duty scheduling and take vehicle scheduling constraints and costs heuristically into account.

In Tosini & Vercellis [1988] an approach for extra-urban scenarios is described. The model and the algorithm has the following restrictions: Transfers of drivers are only possible on board of vehicles. A driver has to start and end his duty in the same depot. Every vehicle can use all deadheads and trips. Therefore this problem can be seen as a combined vehicle and duty scheduling problem of Section 2.2.2. Tosini & Vercellis [1988] generate the columns of a set-covering-model analogously to model (DSP_v) by a matching based heuristic and find a solution of it by a greedy heuristic. The resulting vehicle rotations will be covered by a vehicle schedule calculated by a minimum-cost-flow algorithm.

Falkner & Ryan [1992] examine the duty scheduling problem of Christchurch, New Zealand. This problem also involves decisions about some of the pull-in and pull-out trips of vehicles and therefore is at least partially an integrated vehicle and duty scheduling problem. The structure of the considered transportation network is radial, almost all changeovers happen at the center of the network. The only other locations for changeovers are the depots. For some vehicle blocks it has to be decided whether they end

at the central depot or at the town center. The town center is nearer to the endpoints of the vehicle blocks but has a limited parking capacity. The problem is modeled as a set partitioning problem with additional constraints. To be able to handle the size of the problem it is partitioned into groups of lines that are scheduled separately. Each of these groups is scheduled in three stages. In the first stage the middle and late duties are scheduled, secondly some early duties, and in the third stage the remaining early and split duties (called broken duties). To ensure the compatibility of the different stages linking constraints are added to the set partitioning problems. The individual stages are solved by a specialized set partitioning solver called ZIP.

Patrikalakis & Xerocostas [1992] propose an alternative decomposition of vehicle and duty scheduling. Again their approach is not fully integrated, however an integrated model similar to the one of Ball et al. [1983] is proposed. This model is solved by the following algorithm: At first trips are split up into tasks at the relief points. Then so called "duty skeletons" are generated. These are incomplete duties without deadhead trips. I.e., a skeleton only defines which timetabled trips a duty contains, but it has still to be determined how they are connected. Then schedules for morning and afternoon duties are computed separately by a set covering approach which covers all tasks by duty skeletons. Next, all deadheads which cannot be used by the duty skeletons in the duty skeleton schedule are eliminated. The remaining vehicle scheduling problem is solved by a minimum cost flow solver. At last the duty skeletons are completed such that they cover all deadheads used in the vehicle schedules. To be able to handle practical problems it was necessary to reduce the number of relief points and possible connections between tasks heuristically. Nevertheless improvements over manually produced solutions formerly used by a public transit company in Athens were reported. According to the authors this approach performs better when vehicle dependent constraints are less important.

The approach of Patrikalakis & Xerocostas [1992] has similarities to the combined approach of Section 2.2.2, but keeps more degrees of freedom in the vehicle scheduling phase by fixing fewer deadheads in the duty skeleton schedules. The drawback of the approach is that duty skeleton generation has to be more "generous" with some constraints on maximum working time between breaks and similar rules, because at this stage of the algorithm it remains unclear how much work is actually done between two tasks.

Gaffi & Nonato [1997] also present an approach for integrated duty and vehicle scheduling suitable for ex-urban public transit. The VSP and DSP are modeled as set partitioning problems. They are coupled by constraints

on the deadheads similar to our approach to the ISP.

Their algorithm constructs at first a vehicle-compatibility-graph G_V and a duty-compatibility-graph G_D modeling the possibilities for including pieces of work in the same block or duty, respectively. Both graphs have the same node sets. Every node represents pieces of work of duties and, simultaneously, vehicle rotations. In particular it is necessary that each of these nodes begins and ends in a (not necessary the same) depot. This implies that relief points outside of depots are not allowed, because there it would be possible to end or begin a piece of work without beginning or ending a vehicle rotation.

The arcs in the compatibility graphs represent the possibility to include the pieces of work represented by the adjacent nodes in the same block or, respectively, in the same duty. Not that the vehicle-compatibility graph has more arcs than the duty-compatibility graph, because of the more restrictive constraints on duties. The nodes of the graphs are calculated by solving a resource constrained shortest path problem with respect to a dual solution of a restricted LP-formulation of this problem. This is possible, because in this article the cost of a duty is linear dependent on the costs of its pieces of work .

Then many duty schedules are calculated by a greedy algorithm using a scoring function on the duties. Vehicle cost are considered in terms of Lagrangean multipliers for a MIP-model concerning the coupling constraints. A feasible vehicle schedule is computed afterwards based on the best found crew schedule.

With this approach instances of the extra-urban service of Bologna in Italy were calculated. The largest instances consisted of 257 trips, resulting in 36 vehicles and 45 duties or 200 trips, resulting in 44 vehicles and 65 duties. It was reported that the two stage approach of generating at first pieces of work and then duties caused problems, due to the large number of possible pieces of work.

2.3.4 Fully Integrated Vehicle and Duty Scheduling

The availability of computers with large memory and computing power, as well as the progress in Linear and Integer Programming made it possible to develop fully integrated vehicle and duty scheduling algorithms in the past ten years.

The complete integration of vehicle and crew scheduling was first investigated in a series of publications by Freling and coauthors (Freling [1997];

Freling et al. [2000, 2001, 2003]). 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 fully integrated ISP instances involving a single type of vehicles was also described by Friberg & Haase [1997] and tested on artificial data. We will now describe these approaches in detail.

Freling [1997] proposes an integer programming model for the integrated vehicle and duty scheduling problem. This model is used in a Lagrangian heuristic with column generation to calculate lower bounds and dual information. Feasible (integral) solutions are constructed heuristically by the information obtained while calculating the lower bound.

The model of the ISP is very similar to the model proposed in this work. It consists of a set partitioning problem to model the DSP. Its columns are generated by a dynamic programming algorithm or by enumeration on a network with levels. The VSP is constricted to a single depot. The only objective is to minimize the number of vehicles, further costs are neglected. The feasibility of duties depends on constraints that can be easily modeled as linear equations like piece duration, working time, and number of pieces. The only break rule considered is the one-block break. Lower bounds on ISP are calculated by means of Lagrangean relaxation of the coupling constraints resulting in a Lagrangean problem similar to (L-ISP) (see Section 2.4.1) and a subgradient approach.

Integral solutions are calculated for various cost functions depending on Lagrangean multipliers found by the subgradient approach. For this two heuristics are used: The first solves at first a duty scheduling problem (DSP_v), then pieces of work based on this solution are fixed, then the resulting vehicle scheduling problem is solved, at last again a DSP is solved based on the solution of the vehicle scheduling problem. The other heuristic solves the VSP with current reduced cost and then the DSP. Computational tests were performed on a Pentium 90 with 32 MB RAM. The duties had up to 3 pieces of work. The test-data had up to 296 trips (14 buses and 32 drivers), or 240 trips (38 buses and 90 drivers).

In the article Freling et al. [2000] scenarios with up to 476 trips and 2,260 arcs that use up to 9 buses and 23 drivers in their solutions were solved. These scenarios use at most two pieces of work per duty. Here and in the following publications constraints on valid duties are such that the pricing problems are solvable in polynomial time.

The article Huisman et al. [2003a] treats a multi-depot VSP subproblem. Here the feasibility of a piece only depends on its duration. The solved scenarios include up to 653 trips (67 buses and 117 drivers). Also results on random data that is publicly available are published there. In de Groot & Huisman [2004] a decomposition technique for scenarios that cannot be solved by the techniques of Huisman et al. [2003a] is presented. These technique is tested on the same instances as before.

A different approach to solve the ISP to optimality is published in Friberg & Haase [1997]. Here two set partitioning models for the VSP and the DSP are coupled by constraints on deadheads. These are solved by column generation, a revised simplex algorithm for the LP-relaxations of the restricted problems, a branch-and-bound algorithm and dynamic programming for the pricing problem of the DSP. The pricing problem here is a resource constraint shortest path problem with linear resource constraints. This approach allows to solve artificially generated instances of the ISP with up to 20 trips to optimality. Computational results for instances with up to 30 trips and 54 tasks were given. Only about 4,000 columns were generated for the largest instance. The number of duties or vehicles were not reported.

In Haase et al. [2001] a duty scheduling centered approach is given that only works for single depots vehicle scheduling problems with homogeneous fleet of vehicles. In this paper the combined model of Section 2.2.2 is extended by constraints that count the number of vehicles. The objective function of this problem is composed by a term linear in the number of used vehicles and cost per used duty. The solution approach is a column generation algorithm in a branch-and-price-and-cut framework. Computational results on problems with up to 350 trips, 700 tasks using 67 vehicles and 121 drivers in the average with an integrality gap up to 1.5% were given.

Some results of this thesis were presented at the conferences Heureka '02 (see Borndörfer et al. [2002]) and CASPT 2004 (see Borndörfer et al. [2004]).

Table 2.2 gives an overview about reported results on integrated vehicle and duty scheduling problems in the literature. The columns #garages and #trips gives an impression of the input size of the largest problem reported in the respective article. In practice the degree of difficulty to solve an ISP also depends on the number of relief points, the number of potential links between trips, the complexity of the duty rules, and the tightness of capacity constraints. The columns #vehicles and #duties give the size of the solution of the problem if reported. In the last column we briefly comment the used model.

Article	#garages	#trips	#vehicles	#duties	used model
Ball et al. [1983]	1	~1,000	–	133	sequential heuristic
Scott [1985]	1	456	54	–	only cost approximation
Tosini & Vercellis [1988]	17	300	–	–	multi-commodity-flow with side constraints
Falkner & Ryan [1992]	1	182	–	41	(DSP) with add. constraints
Patrikalakis & Xerocostas [1992]	–	111	20	45	set covering + min. cost flow
Gaffi & Nonato [1997]	28	257	44	65	(ISP) without relief points outside depots
Freling [1997]	1	296	38	90	(ISP), first general approach
Friberg & Haase [1997]	1	30	–	–	two coupled SPP, exact method
Freling et al. [2000]	1	476	9	23	(ISP)
Haase et al. [2001]	1	350	67	121	(DSP^z_v)
Huisman [2004]		653	67	117	(ISP)
Chapter 7	7	3,698	209	260	(ISP) with resource- and capacity-constraints

Table 2.2: Computations in selected publications about the ISP

2.4 IS-OPT

In this section we give a high level view of our algorithm for integrated vehicle and duty scheduling called IS-OPT. We also give a list of the improvements and extensions in comparison to previous methods to solve the ISP.

The details of the subroutines are discussed below in the appropriate chapters of this thesis.

2.4.1 Outline of our ISP-Algorithm

IS-OPT finds an integral solution for the model (ISP) of section 2.2.3. The main loop solves approximately the LP-relaxation of (ISP) by relaxing the coupling constraints (iv) of (ISP) in a Lagrangean way and solving the Lagrangean Problem (L-ISP)

$$\max_{\lambda \in \mathbb{R}^{A_{\mathsf{VSP}}}} \left[\min \ (c^{\mathsf{T}} - \lambda^{\mathsf{T}} B)x + \gamma^{\mathsf{T}} z \ + \min \ (d^{\mathsf{T}} + \lambda^{\mathsf{T}} M)y \right],$$
$$\begin{aligned} \text{s.t. } & Ax = 1, \\ & Rx - z \le r, \\ & x \in \{0,1\}^D, z \ge 0 \end{aligned} \qquad \begin{aligned} \text{s.t. } & Ny = b, \\ & y \in \{0,1\}^{A_{\mathsf{VSP}}} \end{aligned}$$

by the proximal bundle method (see Chapter 4). Lagrangean relaxation is explained in the next chapter. Problem (L-ISP) decomposes in two subprob-

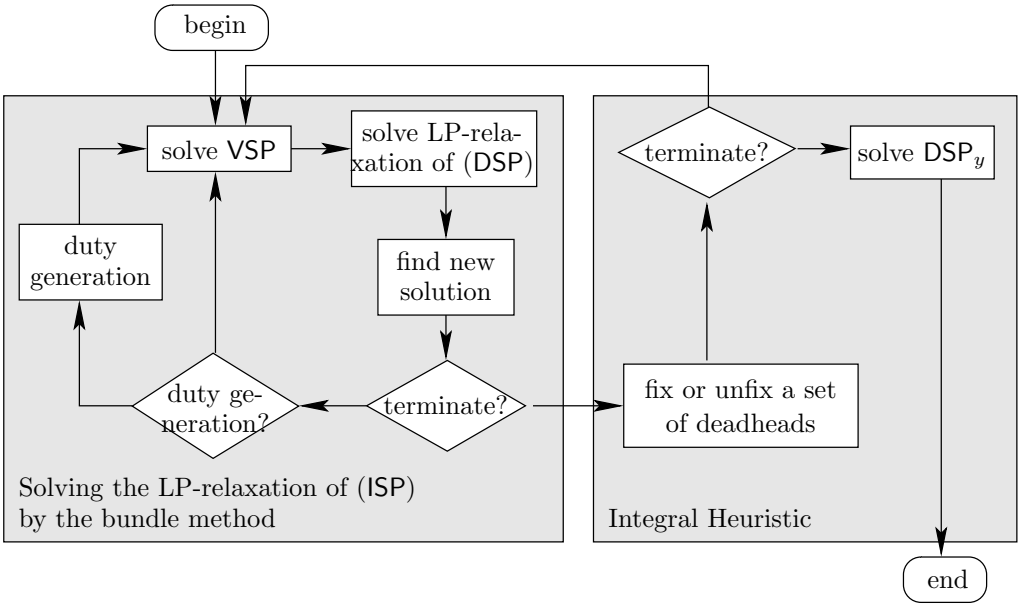


Figure 2.5: Outline of the ISP Algorithm

lems, namely the duty scheduling problem with all possible deadheads, i.e., model (DSP$_v$) of Section 2.2.2, and the vehicle scheduling problem modeled by (VSP). The subproblems are only coupled by the Lagrangean multipliers λ. Model (VSP) is solved repeatedly by VS-OPT, see Section 1.9.5, giving lower bounds on it and, in general, non-optimal integral solutions. The LP-relaxation of model (DSP$_v$) is generated by a column generation approach and approximated by the bundle method, giving lower bounds and an approximation of a primal LP-solution. The pricing problem of the column generation and solution approaches are explained in Chapter 5. How we use these inexact solutions of the subproblems of (L-ISP) in the bundle method to calculate bounds and primal information of (ISP) is explained in Chapter 4. Finally, these bounds and primal information are used as input for a branch-and-bound like integral heuristic described in Chapter 6. Computational results of IS-OPT on real world data are discussed in Chapter 7.

A simplified flow chart of IS-OPT is pictured in Figure 2.5. The left box of this figure contains a simplified outline of our column generation method to solve the LP-relaxation of (ISP). The right box depicts the primal heuristic, the procedure to solve (DSP$_y$) called *rapid branching* is explained in Chapter 6.

The subproblems and techniques needed to solve the ISP used in IS-OPT are

- Multi-Commodity Flow Problems arising by the model of VSP, approximated by Lagrangean relaxation and arc generation, see Löbel [1997b],
- Minimum Cost Flow Problems arising by single depot vehicle scheduling problems that occur as subproblems solved by the network-simplex solver MCF, see Löbel [1996],
- an inexact bundle method to solve the occurring LP-relaxations, see Chapter 4
- a generalized Set Partitioning Problem solved by column generation, see Chapter 3, the pricing problem itself is a generalized resource constrained shortest path problem and treated in Chapter 5, and
- Rapid Branching heuristic to solve the model (ISP) to integrality, see Chapter 6.

2.4.2 Contributions

Our contributions to solution techniques for integrated vehicle and duty scheduling problems concentrate on solving large real world instances in reasonable time. In general we are not able to prove the quality of our solutions because we are not able to calculate lower bounds of objective values of model (ISP). Therefore we can also not guarantee the optimality of our solutions.

In detail we

- combine column generation and subgradient based methods, that is we replace an optimal dual solution by Lagrangean multipliers in the pricing problems,
- we use inexact subgradients in a bundle method to calculate the Lagrangean multipliers and primal information,
- we solve the pricing problems inexact by resource constrained shortest path problem approximations, and
- we use a fast inexact branch and bound heuristic to solve ISP to integrality.

Summarizing, we replace known exact LP-techniques by LP-based approximation algorithms. This has the advantage that we are able to tune the accuracy and speed of the individual components of IS-OPT to get good solutions in reasonable time even of large problems.

Chapter 3

Basic Methodology

In this chapter we explain the basic methodologies column generation and Lagrangean relaxation that are used by our algorithm to solve the duty scheduling problem as well as by the algorithm to solve the integrated vehicle and duty scheduling problem. We also describe how Lagrangean relaxation can be used in column generation approaches to calculate dual variables needed for their pricing problems.

3.1 Column Generation

Column generation can be seen as a technique of solving large LPs, whose columns are given implicitly. These implicitly given columns will be generated throughout the solution process if it becomes necessary. The complete LP with all its columns is called *master problem*. An LP consisting of a explicitly given subset of columns of the master problem is called *restricted problem*.

We now look at the following master problem:

$$\begin{aligned} (\mathsf{M}) \qquad & \min\ c^{\mathsf{T}}x, \\ & \text{s.t.}\quad Ax = a, \\ & \qquad\quad x \geq 0, \end{aligned}$$

where the constraint matrix A is in $\mathbb{R}^{m\times n}$, $c, x \in \mathbb{R}^n$, and $a \in \mathbb{R}^m$. In general, column generation is used if n is much larger than m. We now look at a

subset $I \subset \{1, \ldots, n\}$ of the columns of A, which are given explicitly. That is, we consider the *restricted problem*

$$\begin{aligned} (\mathsf{R}) \qquad & \min\ {c_I}^{\mathsf{T}} x_I, \\ & \text{s.t.}\ \ A_{\cdot I} x_I = a, \\ & \qquad\quad x_I \geq 0. \end{aligned}$$

We are able to solve (M), if we are able to solve (R) and if additionally an oracle is available that solves the so called *pricing problem* (PRICE) for any $\lambda \in \mathbb{R}^m$:

$$(\mathsf{PRICE}) \qquad \min_{i=1,\ldots,n} c_i - \lambda^{\mathsf{T}} A_{\cdot i}$$

If λ is an optimal dual solution of (R) for a given $I \subset \{1, \ldots, n\}$ and the optimal value of (PRICE) is non-negative, then λ is also an optimal dual solution of (M). In this case, an optimal primal solution of (R), extended by $x_{\{1,\ldots,n\}\setminus I} = 0$, is also an optimal primal solution of (M). Otherwise we have found a column that potentially improves the objective value, so we add it to I and solve the restricted problem again. This can be iterated until optimality has been proven. The process terminates, because n is finite.

This column generation algorithm is stated as pseudo code in Algorithm 1. One can show, that if (PRICE) can be solved in polynomially time also (M)

Algorithm 1 Column Generation

Input: A starting set of columns I, a method to solve (PRICE), an oracle that gives for column i its cost c_i and the column $A_{\cdot i}$ of the coefficient matrix.

Output: An optimal primal solution x of (M).

1: **repeat**
2: Solve (R) with columns I. This results in a optimal dual solution $\lambda \in \mathbb{R}^m$ and an optimal primal solution $x_I \in \mathbb{R}^I$.
3: Solve (PRICE) with respect to λ. Let $i^* \in \{1, \ldots, n\}$ be the solution of (PRICE).
4: Add i^* to I.
5: **until** $c_{i^*} - \lambda^{\mathsf{T}} A_{\cdot i^*} \geq 0$
6: Set $x_{\{1,\ldots,n\}\setminus I} \leftarrow 0$.

can be solved in polynomially time, even if (M) contains an exponential number of columns. A proof can be found in [Grötschel et al. 1993, Chapter 6].

The article Desrosiers & Lübbecke [2005] is a good primer about column generation. A first survey about column generation has been written by

Barnhart et al. [1998]. A generalized branching rule for branch-and-bound algorithms using column generation and a recent survey about column generation can be found in Villeneuve et al. [2003], a recent book about column generation and its applications is Desaulniers et al. [2005].

3.2 Lagrangean Relaxation

Lagrangean relaxation is a tool to find upper bounds for a maximization problem. Under certain circumstances also optimal solutions can be found. We will highlight the basics of Lagrangean relaxation and results used in this work. For further details see [Hiriart-Urruty & Lemaréchal 1993b, Chapter XII] or Lemaréchal [2001].

3.2.1 Lagrangean Relaxation in General

Let us consider a combinatorial problem (P) on a set $X \neq \emptyset$:

$$\begin{aligned} \text{(P)} \quad & \sup\, c(x), \\ & \text{s.t. } a_j(x) = 0, \quad \text{for } j = 1, \ldots, m, \\ & \quad x \in X. \end{aligned}$$

(P) has an objective function $c : X \to \mathbb{R}$, and constraint functions $a_1, \ldots, a_m : X \to \mathbb{R}$. (We write also, analogously to vectors: $a := (a_1, \ldots, a_m)^\mathsf{T} : X^m \to \mathbb{R}^m$.) (P) is called the *primal problem*. We call an $x \in X$ satisfying all $a_j(x) = 0$ *feasible*. The Lagrange function or shortly *Lagrangean* L is defined by

$$L(x, \lambda) := c(x) - \lambda^\mathsf{T} a(x), \forall \lambda \in \mathbb{R}^m \text{ and } x \in X.$$

The vectors $\lambda \in \mathbb{R}^m$ are called *Lagrangean multipliers*. The suprema of the Lagrangean for fixed $\lambda \in \mathbb{R}^m$ are defining a function $\phi(\lambda) := \sup_{x \in X} L(x, \lambda)$ called Lagrangean dual function or shortly *dual function* with some interesting properties: For all $\lambda \in \mathbb{R}^m$ and all feasible x it holds $\phi(\lambda) \geq c(x)$. This property is called *weak duality*. If $\phi(\lambda)$ is significantly easier to compute than the optimum of (P) we are probably able to solve (or at least approximate) the dual problem (D), also called Lagrangean dual, defined by

$$\text{(D)} \quad \inf_{\lambda \in \mathbb{R}^m} \phi(\lambda),$$

faster than the original problem.

This gives us an upper bound on (P) that is under certain circumstances tight, and sometimes we may even get a solution of the primal problem or an approximation of it.

The function ϕ is convex and lower semicontinuous, therefore, if we are able to compute $\phi(\lambda)$ efficiently, we are also able to solve (D) efficiently. If further $\sup_{x \in X} L(x, \lambda)$ has an optimal solution x^λ for a certain λ then $g^\lambda := -a(x^\lambda)$ is a subgradient of ϕ at λ. Moreover, all subgradients of ϕ are defined by maximizers of the Lagrangean if X is compact, c is semi-continuous on X, and each a_j is continuous. More precisely:

$$\partial\phi(\lambda) = -\operatorname{conv}\{a(x) : x \text{ is a maximizer of } L \text{ at } \lambda\}. \tag{3.1}$$

We denote by $\partial f(\lambda)$ the set of subgradients of a function f at point λ. Equation (3.1) is called the *filling property* at λ.

The next interesting question is under which circumstances the primal and dual problem have the same optimum. That is, when does the following equation hold:

$$\inf_{\lambda \in \mathbb{R}^m} \phi(\lambda) = \sup_{x \in X, a(x)=0} c(x). \tag{3.2}$$

Theorem 3.1 (Hiriart-Urruty & Lemaréchal [1993b])**.** Equation (3.2) holds if the filling property (3.1) is fulfilled for all λ and X is convex, c is concave and $a : X \to \mathbb{R}^m$ is affine.

The proof can be found in Hiriart-Urruty & Lemaréchal [1993b].

Moreover, all optimal solutions of (P) are maximizers of L at an optimal solution λ^* of (D). This is the basis for recovering primal solutions from the solution of the Lagrangean dual.

How we use Lagrangean relaxation to approximate certain LPs can be found in Chapter 4. In the next two sections we will show that Lagrangean duality is closely related to LP duality and quadratic duality.

3.2.2 Linear Programming Duality

Here we show the connection between Lagrangean and LP duality.

Let the primal problem be an LP:

$$\begin{aligned} \text{(P)} \qquad & \max\ c^\mathsf{T} x, \\ & \text{s.t.}\quad Ax = b, \\ & \qquad\ \ x \geq 0. \end{aligned}$$

Then the Lagrangean dual is

$$\begin{aligned}\phi(\lambda) &= \max_{x\geq 0} \left[c^\mathsf{T}x + \lambda^\mathsf{T}(b - Ax)\right], \\ &= \max_{x\geq 0} \left[\lambda^\mathsf{T}b + (c^\mathsf{T} - \lambda^\mathsf{T}A)x\right]\end{aligned}$$

Here you can see that

$$\phi(\lambda) = \begin{cases}\infty, & \text{if } \lambda^\mathsf{T}A > c, \\ \lambda^\mathsf{T}b, & \text{otherwise.}\end{cases}$$

Thus the Lagrangean dual takes its optimum at the minimizer of

$$\begin{aligned}\text{(D)} \quad & \min\ \lambda^\mathsf{T}b, \\ & \text{s.t. } \lambda^\mathsf{T}A \leq c.\end{aligned}$$

This is exactly the LP-dual of (P).

3.2.3 Quadratic Programming Duality

Another general problem, of which a specialized version occurs in the bundle method, is the quadratic programming problem. One of its variants can be written as

$$\begin{aligned}\text{(QP)} \quad & \max\ c^\mathsf{T}x - \tfrac{1}{2}x^\mathsf{T}Qx, \\ & \text{s.t. } Ax \geq b.\end{aligned}$$

Here Q is assumed to be positive semi-definite. This ensures that the objective function is convex. The Lagrangean function for (QP) is (replacing $Ax \geq b$ by $Ax - s = b$, $s \geq 0$)

$$L(x, s, \lambda) := c^\mathsf{T}x - \frac{1}{2}x^\mathsf{T}Qx - \lambda^\mathsf{T}(Ax - s - b). \tag{3.3}$$

The Lagrangean dual is

$$\begin{aligned}\phi(\lambda) &= \max_{x,s\geq 0} c^\mathsf{T}x - \frac{1}{2}x^\mathsf{T}Qx - \lambda^\mathsf{T}(Ax - s - b), \\ &= \begin{cases}\max_x (c^\mathsf{T} - \lambda^\mathsf{T}A)x - \frac{1}{2}x^\mathsf{T}Qx + \lambda^\mathsf{T}b, & \text{for } \lambda \geq 0, \\ \infty, \text{ otherwise.}\end{cases}\end{aligned}$$

The Lagrangean function L takes its maximum for a fixed λ at a point x^λ with $Qx^\lambda = c - A^\mathsf{T}\lambda$ by the Karush-Kuhn-Tucker conditions (see, e.g.,

Nering & Tucker [1993]). So the Lagrangean dual can be written directly as (see Lemaréchal [2001]),

$$\phi(\lambda) = \lambda^{\mathsf{T}} b + \frac{1}{2} x^{\lambda\mathsf{T}} Q x^{\lambda}, \quad \text{with } Qx^{\lambda} = c - A^{\mathsf{T}}\lambda. \tag{3.4}$$

Thus, the Lagrangean dual of the quadratic problem (QP) denoted by (DQ) is equal to the quadratic dual in Dorn [1960]:

$$\begin{aligned} \text{(DQ)} \quad & \min \ \lambda^{\mathsf{T}} b + \tfrac{1}{2} x^{\mathsf{T}} Q x, \\ & \text{s.t. } A^{\mathsf{T}}\lambda + Qx = c, \\ & \lambda \geq 0. \end{aligned}$$

This stems from (3.4) simply by adding the optimality condition $A^{\mathsf{T}}\lambda + Qx = c$ as a constraint. We will use problem (DQ) in Chapter 4 to find an improving direction in the bundle method.

3.3 Lagrangean Relaxation for Column Generation

In column generation approaches there are two time consuming problems which have to be solved repeatedly: On the one hand an optimal dual solution of the restricted problem has to be found, i.e., LPs have to be solved. On the other hand we have to find new columns (or prove that none exists) depending on the solutions of the LPs by solving the pricing problems.

Using Lagrangean relaxation and subgradient methods is often faster than LP-methods, but in general, this approach only gives bounds and approximated solutions of the original problem. We explain in this section how we transform the Lagrangean multipliers of the LP-relaxation of (DSP) heuristically such that they are useful to approximately solve the pricing problem corresponding to the original master problem.

3.3.1 Problem Class

We consider in this section the ILP of the duty scheduling problem as proposed in Section 1.10.3:

$$
\begin{array}{lll}
(\mathsf{DSP}) & \min\ c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z, & \\
& \text{s. t.} & \\
\text{(i)} & Ax & = \mathbb{1}, \\
\text{(ii)} & Rx \quad -z & \leq r, \\
\text{(iii)} & x & \in \{0,1\}^n, \\
\text{(iv)} & z & \geq 0.
\end{array}
$$

$A \in \{0,1\}^{m' \times n}$ is a binary matrix, $R \in \mathbb{R}^{m'' \times n}$ is a coefficient matrix of so called resource constraints. The variables $z \in \mathbb{R}_+^{m''}$ can be seen as slack variables of the base-constraints (ii). The cost coefficients c_i, $i = 1, \ldots, n$ are larger than zero and γ_i, $i = 1, \ldots, m''$ are non-negative. Most of the duty and crew scheduling problems as well as multi depot vehicle scheduling problems can be modeled as problems of type (DSP). Also generalized assignment problems as stated in Barnhart et al. [1998] belong to this class.

Observe that the upper bounds on x in model (DSP) are redundant because A is binary and the cost coefficients of its columns are positive.

3.3.2 Restricted Problem

In general we only know a subset I of columns of A because the total number of columns is too large. In the following (DSP^I) denotes an LP-relaxation of (DSP) restricted to columns in a set $I \subset \{1, \ldots, n\}$:

$$
\begin{array}{lll}
(\mathsf{DSP}^I) & \min\ {c_I}^{\mathsf{T}}x_I + \gamma^{\mathsf{T}}z, & \\
& \text{s. t.} & \\
\text{(i)} & A_{\cdot I}x_I & = b, \\
\text{(ii)} & R_{\cdot I}x_I \quad -z & \leq r, \\
\text{(iii)} & x_I & \in \{0,1\}^I, \\
\text{(iv)} & z & \geq 0.
\end{array}
$$

3.3.3 Pricing Problem

The dual problem of the LP-relaxation of (DSP^I) is (letting out the redundant upper bounds on x):

$$\begin{aligned} (\mathsf{D\text{-}DSP}^I) \quad & \max \quad \lambda^\mathsf{T} b + \mu^\mathsf{T} r, \\ & \text{s. t.} \\ \text{(i)} \quad & A_{\cdot I}{}^\mathsf{T}\lambda + R_{\cdot I}{}^\mathsf{T}\mu \leq c_I, \\ \text{(ii)} \quad & -\gamma \leq \mu \leq 0. \end{aligned}$$

Let (λ^*, μ^*) be an optimal solution of ($\mathsf{D\text{-}DSP}^I$). If we are able to proof that no column $i \in \{1, \ldots, n\}$ exists such that the *reduced cost* of column i

$$\bar{c}_i := c_i - \lambda^{*\mathsf{T}} A_{\cdot i} - \mu^{*\mathsf{T}} R_{\cdot i}$$

is negative, then (λ^*, μ^*) is also an optimal solution of the dual of the LP-relaxation of (DSP).

To proof that no column with negative reduced cost exists, we have to solve the following pricing problem:

$$\min_{i \in \{1,\ldots,n\}} c_i - \lambda^{*\mathsf{T}} A_{\cdot i} - \mu^{*\mathsf{T}} R_{\cdot i}. \tag{3.5}$$

Observe that all columns $i \in I$ have non-negative reduced cost by inequalities (i) of ($\mathsf{D\text{-}DSP}^I$).

Column generation approaches are useful if problem (3.5) can be solved in short time. Often it is again a linear program or has at least a linear cost function. Let Y be an appropriate solution space. Then the pricing problem can be written as:

$$\min_{y \in Y} \zeta^\mathsf{T} y - \lambda^{*\mathsf{T}} a(y) - \mu^{*\mathsf{T}} r(y).$$

Here $a(y')$ and $r(y')$ gives the column of A or R, respectively, that corresponds to the solution $y' \in Y$ of the pricing problem. In the duty scheduling problem for example, the cost of a column depends often linearly on the arcs of the underlying planning graph, which are used by the corresponding duty.

3.3.4 Lagrangean relaxation

We are able to find a lower bound of the objective value of problem ($\mathsf{D\text{-}DSP}^I$) by solving a Lagrangean relaxation of (DSP^I) which stems from relaxing its

equations (i) and (ii). The Lagrangean dual problem is then:

$$\max_{\substack{\lambda \text{ free},\\ \mu \leq 0}} \left[\min_{\substack{x \in [0,1]^n,\\ z \geq 0}} {c_I}^\mathsf{T} x_I + \gamma^\mathsf{T} z + \lambda^\mathsf{T}(1 - A_{\cdot I} x_I) + \mu^\mathsf{T}(r - R_{\cdot I} x_I + z) \right].$$

It can also be written as

$$\max_{\substack{\lambda \in \mathbb{R}^{m'},\\ -\gamma \leq \mu \leq 0}} \left[\lambda^\mathsf{T} 1 + \mu^\mathsf{T} r + \min_{x \in [0,1]^n} ({c_I}^\mathsf{T} - \lambda^\mathsf{T} A_{\cdot I} - \mu^\mathsf{T} R_{\cdot I}) x_I \right].$$

We denote this problem by (L-DSP^I). The variables z can be eliminated because they are zero in every optimal solution of (L-DSP^I).

For fixed Lagrangean multipliers λ and μ the subproblem

$$\min_{x \in [0,1]^I} (c_I - \lambda^\mathsf{T} A_{\cdot I} - \mu^\mathsf{T} R_{\cdot I}) x$$

is easy to solve by setting x_i to one if $\lambda^\mathsf{T} A_{\cdot i} + \mu^\mathsf{T} R_{\cdot i} \geq c_i$ and to zero otherwise.

Unfortunately a solution (λ^L, μ^L) of (L-DSP^I) is in general not a solution of (D-DSP^I) because ${A_{\cdot i}}^\mathsf{T} \lambda^L + {R_{\cdot i}}^\mathsf{T} \mu^L \leq c_i$ is not guaranteed to hold. However, this solution can be complemented to a solution of the dual of (DSP^I) including the upper bounds on x:

$$\begin{aligned}
&(\text{D-DSP2}^I) \quad \max \quad \lambda^\mathsf{T} b + \mu^\mathsf{T} r + \nu^\mathsf{T} \mathbb{1}, \\
&\qquad\qquad \text{s. t.} \\
&\text{(i)} \qquad\qquad {A_{\cdot I}}^\mathsf{T} \lambda + {R_{\cdot I}}^\mathsf{T} \mu + \nu_I \leq c_I, \\
&\text{(ii)} \qquad\qquad -\gamma \leq \mu \leq 0, \\
&\text{(iii)} \qquad\qquad \nu \leq 0.
\end{aligned}$$

This can be done by setting ν_i^L to $c_i - {\lambda^L}^\mathsf{T} A_{\cdot i} - {\mu^L}^\mathsf{T} R_{\cdot i}$ if this term is negative and to zero otherwise. Observe, that all ν_i^L for $i \in \{0, \ldots, n\} \setminus I$ are in an optimal solution at their upper bound and therefore zero. If (λ^L, μ^L) was optimal for (L-DSP^I), $(\lambda^L, \mu^L, \nu^L)$ is optimal for (D-DSP2^I), because the objective values of (L-DSP^I) and (D-DSP^I) are equal and we know that both problems have no duality gap to (DSP) (see Theorem 3.1).

The pricing problem related to (D-DSP2^I)

$$\min_{i \in \{1,\ldots,n\}} c_i - {\lambda^*}^\mathsf{T} A_{\cdot i} - {\mu^*}^\mathsf{T} R_{\cdot i} - \nu_i \tag{3.6}$$

is therefore equivalent to

$$\min_{i\in\{1,\dots,n\}\setminus\{j\in I|\nu_i<0\}} c_i - \lambda^{*\mathsf{T}} A_{\cdot i} - \mu^{*\mathsf{T}} R_{\cdot i}. \tag{3.7}$$

Problem (3.7) is in general more difficult to solve than problem (3.5), because the exclusion of columns i with negative ν_i may destroy an eventually existent structure of the solution space. E.g., the pricing problem of the duty scheduling problem corresponding to (3.5) is a constrained shortest path problem. To formulate the pricing problem corresponding to (3.7) we would have to add constraints that prevent solutions with negative ν_i, $i \in I$.

3.3.5 Reduced Cost Shifting

We present now a heuristic to transform a solution of (D-DSP2^I) into a solution of (D-DSP^I) with hopefully only a small decrease of the function value.

Our heuristic method "shifts" the dual variables $\lambda^L \in \mathbb{R}^{m'}$ of a solution $(\lambda^L, \mu^L, \nu^L)$ of (D-DSP2^I) towards a dual solution $(\lambda^L + \delta, \mu^L)$ of (D-DSP^I). The idea of this algorithm is to find negative $\delta_j, j \in \{1, \dots, m'\}$ corresponding to active rows of columns of A with negative reduced cost until the new reduced cost of all columns are not less than zero.

Let for this $S(i)$ be the index set of non-zero entries of $A_{\cdot i}$. The reduced-cost-shifting algorithm is shown as Algorithm 2.

The steps 5–13 ensure that the Lagrangean multipliers (λ^i, μ^L) gets shifted to (λ^{i+1}, μ^L) such that the new negative reduced cost $\bar{c}_i(\lambda^{i+1}, \mu^L)$ of column i are zero. Here simply the reduced cost with respect to (λ^i, μ^L) are added to the Lagrangean multipliers related to rows covered by column i. The sorting of the columns by their initial reduced cost in step 2 is a kind of greedy strategy: We adapt at first the Lagrangean multipliers with the largest impact.

Observe, ν_i is exactly $c_i - A^\mathsf{T}\lambda - R^\mathsf{T}\mu$ if negative and zero else. Therefore, if we would be able to calculate a δ such that $A^\mathsf{T}\delta = \nu$ the optimality of $(\lambda + \delta, \mu)$ would follow by the equality of the objective values. However, our heuristic only ensures $A^\mathsf{T}\delta \leq \nu$ and thus potentially decreases the objective value. We repair this by performing a warmstart of the bundle method with the new multipliers $(\lambda^L + \delta, \mu^L, 0)$. This is iterated until ν is nearly zero also in the solution of (D-DSP^I). We are not able to give a guarantee of convergence of this procedure, however in practice it works well until all $-\nu_i$ are smaller than a certain threshold $\epsilon > 0$. Then the reoptimizing of the bundle method often leads again to negative ν_i smaller than $-\epsilon$.

Algorithm 2 Reduced Cost Shifting

Input: Lagrangian multipliers λ^L, μ^L of (L-DSP^I).

Output: A (in general non-optimal) dual solution λ, μ of (D-DSP^I).

1: Calculate the reduced cost $\bar{c}_i(\lambda^L, \mu^L) := c_i - (\lambda^L)^\mathsf{T} A_{\cdot i} - (\mu^L)^\mathsf{T} R_{\cdot i}$ for all columns $i \in I$.
2: Sort the columns in I by their reduced cost $\bar{c}_i(\lambda^L, \mu^L)$, such that $\bar{c}_1(\lambda^L, \mu^L) \leq \bar{c}_2(\lambda^L, \mu^L) \leq \cdots \leq \bar{c}_{|I|}(\lambda^L, \mu^L)$, set $\lambda^1 \leftarrow \lambda^L$. Let j be the last column with $\bar{c}_j(\lambda^L, \mu^L) < 0$.
3: **for** $i = 1$ to j **do**
4: Let $\bar{c}^i \leftarrow \bar{c}_i(\lambda^i, \mu^L)$.
5: **if** $\bar{c}^i < 0$ **then**
6: **for** all $t \in \{1, \ldots, m'\}$ **do**
7: **if** $t \in S(i)$ **then**
8: $\lambda_t^{i+1} \leftarrow \lambda_t^i + \bar{c}^i / |S(i)|$
9: **else**
10: $\lambda_t^{i+1} \leftarrow \lambda_t^i$
11: **end if**
12: **end for**
13: **end if**
14: **end for**
15: Set $\lambda \leftarrow \lambda^{j+1}$.

Chapter 4

Proximal Bundle Method

The proximal bundle method (PBM) is a method to minimize an unbounded, continuous, convex, and possibly non-smooth function $f : \mathbb{R}^m \to \mathbb{R}$. The PBM can be used in combination with Lagrangean relaxation to approximate primal and dual solutions of linear programs.

We develop a variant of the PBM, that we call *inexact PBM,* which is suited to large scale LPs generated by column generation, in particular if the pricing problem can not be solved exactly. The novel extensions of the PBM involve the use of inexact subgradients and function evaluations of the Lagrangean duals. We are able to show that our inexact PBM produces a series of trial points that converges to a point with a function value which differs at most by an $\epsilon \geq 0$ from the optimal value. This ϵ depends on the quality of the approximated subgradients and function values. The inexact PBM can also be used for a new active set method that speeds up the PBM. This active set method utilizes subsets of columns of Lagrangean relaxations to calculate the approximated function values and ϵ-subgradients of the Lagrangean dual.

We use the inexact PBM to approximate LP-relaxations of model (ISP) (defined in Section 2.2.3) via the Lagrangean problem (L-ISP) of Section 2.4.1. The corresponding computational results can be found in Chapter 7. The LP-relaxation of (ISP) is in general too large to be solved by standard solvers such as the barrier algorithm or the dual simplex of CPLPEX 10.0 because theses LPs consist in general of millions of columns for the duties and deadheads and hundreds of thousands of rows for the coupling constraints.

This chapter is organized as follows: It describes the idea behind the PBM and discusses algorithmic details in Section 4.1. The next section compares the PBM with other subgradient methods. Section 4.3 gives an overview of

a number of extensions of the PBM that we use in IS-OPT, such as a method to exploit block structures of functions, handling of bounded functions, and calculating primal information for LPs. Section 4.4 shows the novel active-set extension of PBM developed to solve LPs with many columns. Applications of the PBM from the literature as well as the ones originating from solving the ISP are discussed in Section 4.5. Then follows the description of the inexact PBM in Section 4.6.

We approximate subproblems occurring in IS-OPT, namely RCSP, see Section 5.5.1 and DSP (Section 1.10.3) by the PBM. Computational results on LP-relaxations of set-partitioning-problems are shown in Section 4.7, we show that the inexact PBM is able to approximate even large problems very fast, and that it is in most cases faster than other subgradient approaches or exact methods.

4.1 Description

We sketch here the idea of the PBM and outline facts that we need to explain our adaptions. The PBM is a method to minimize any unbounded convex function f. In our context f is the Lagrangian dual of a combinatorial problem. Thus, we denote in this thesis the argument of f by λ or μ analogously to the variables of dual linear programs. By x or y we denote a variable of the original problem.

An overview on convex optimization algorithms including bundle methods can be found in Hiriart-Urruty & Lemaréchal [1993a,b]. A detailed description of the bundle method itself can be found in Kiwiel [1990] and of its quadratic subproblem solver in Kiwiel [1994].

4.1.1 Idea and Properties

We will first give a "high level" description of the PBM. The details will follow as needed in the subsequent sections.

As mentioned above we want to minimize a convex function f. The idea of the PBM is to collect a set J^i of linearizations $\bar{f}^i$ of f. These sets are called *bundle*, these are the bundles which give the proximal *bundle* method its name. The linearizations in the bundles are used to construct a model $\hat{f}_{J^i}$ of f. This model is called polyhedral model or sometimes also cutting plane model. At each iteration i of the PBM we improve the model by adding

a new linearization at a new trial point λ^{i+1} which is the minimizer of the sum of the current polyhedral model $\hat{f}_{J^i}$ and a quadratic term that penalizes the deviation from a current so called *stability center* $\bar{\lambda}^i$. This sum is also called *quadratic model* of f. If the new minimizer improves the function value enough to guarantee convergence, the stability center is moved to this trial point. This is called *serious step*. Otherwise a so called *null step* occurs which only improves the polyhedral model.

It is known that the series of stability centers $(\bar{\lambda}^i)$ converges to a minimizer of f. If f is polyhedral, as it is the case for Lagrangean relaxation of LPs, it is also known that the PBM converges in a finite number of iterations. However, in practice the convergence is in our cases too slow to run the PBM until optimality, such that we must terminate the PBM heuristically.

The PBM produces for the Lagrangian relaxation of LPs also a series of approximations of primal solutions of the original LP, which converges to an optimal primal solution. See Kiwiel [1990] for proofs of these results.

The PBM can, besides its relations to subgradient methods, be interpreted as a cutting plane algorithm (see [Hiriart-Urruty & Lemaréchal 1993b, Chapter XII, 4.2]). In cutting plane algorithms at each iteration an affine approximation of the minimized function f is calculated. Subsets and convex combinations of these linearizations define a cutting plane model of f that is used to find the next trial point.

The time critical parts of the PBM are the minimization of the quadratic subproblem and the calculation of the function values and the subgradients of f. The computation time for the quadratic problem depends mostly on the number of active linearizations in the bundle. The time for the function evaluations and subgradient calculations varies depending on the problems and Lagrangean relaxation used. For set partitioning problems we need only milliseconds to calculate a subgradient, because this only involves optimizing over a box. For the ISP the same step may need a couple of minutes or sometimes also hours because we have to minimize the LP-relaxations of a vehicle and a duty scheduling problem to calculate a subgradient.

In the next Section we describe the polyhedral model in detail. In Section 4.1.3 we describe the problem of finding the next trial point λ^{i+1}, in Section 4.1.4 we state the PBM as pseudo-code. In Section 4.1.5 we state our weight updating strategy. At last we sketch a proof of convergence for the PBM.

4.1.2 Subgradients, Linearizations, and Cutting Plane Models

The main concept exploited in the PBM is a polyhedral model $\hat{f}_J$ of $f : \mathbb{R}^m \to \mathbb{R}$ defined by the bundle J of affine functions. Every function $\bar{f}$ in the bundle J minorizes f, that is $\bar{f}(\lambda) \leq f(\lambda)$ for all $\lambda \in \mathbb{R}^m$. The functions in J arise by linearizations of f or convex combinations of linearizations.

A linearization $\bar{f}_{\lambda^i}$ of f at a point $\lambda^i \in \mathbb{R}^m$ is an affine function of the following type:

$$\bar{f}_{\lambda^i}(\lambda) := f(\lambda^i) + g_f(\lambda^i)^\mathsf{T}(\lambda - \lambda^i), \qquad \lambda \in \mathbb{R}^m. \tag{4.1}$$

Here $g_f(\lambda^i) \in \mathbb{R}^m$ is a subgradient of f at λ^i, i.e., $g_f(\lambda^i)$ fulfills the following inequality:

$$f(\lambda) \geq f(\lambda^i) + g_f(\lambda^i)^\mathsf{T}(\lambda - \lambda^i), \quad \text{for all } \lambda \in \mathbb{R}^m. \tag{4.2}$$

There may exist more than one subgradient for non-smooth functions f at a certain point. The set of all subgradients of f at λ^i is denoted by $\partial f(\lambda^i)$. Every subgradient defines a linearization of f. For our purpose an arbitrary subgradient suffices, however more subgradients may help to increase the speed of the algorithm.

A polyhedral approximation $\hat{f}_J$ of f called *cutting plane model* or shortly *model* of f for a given bundle J of affine functions is defined by

$$\hat{f}_J(\lambda) := \max_{\bar{f} \in J} \bar{f}(\lambda). \tag{4.3}$$

Clearly, we have $\hat{f}_J(\lambda) \leq f(\lambda)$ for all $\lambda \in \mathbb{R}^m$. Figure 4.1 shows an example of a convex function f, a linearization, and a cutting plane model of f.

4.1.3 Quadratic Subproblem

In every iteration i of the PBM the current bundle J^i is used to find a "good" next trial point λ^{i+1} that has a low *model value* (i.e. low $\hat{f}_{J^i}(\lambda^{i+1})$) and is in the vicinity of the current stability center $\bar{\lambda}^i$. To obtain such a point the following problem is solved:

$$\lambda^{i+1} \in \arg\min_{\lambda \in \mathbb{R}^m} \hat{f}_{J^i}(\lambda) + u^i \|\lambda - \bar{\lambda}^i\|^2/2. \tag{4.4}$$

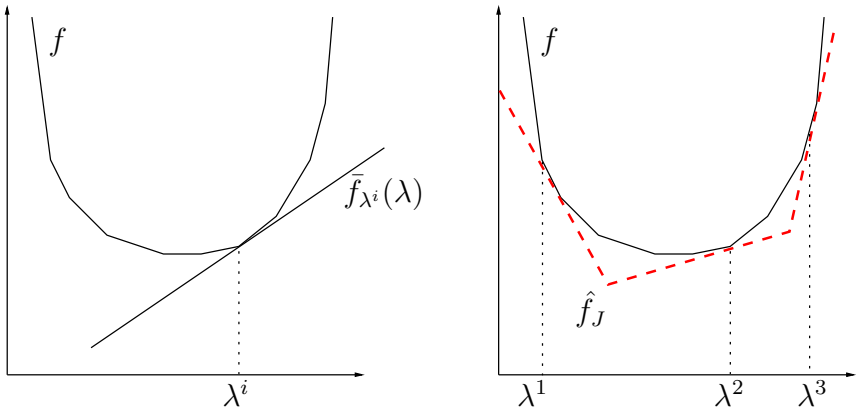


Figure 4.1: Linearization and cutting plane model of f

Here $u^i > 0$ is a weighting of the quadratic term and J^i is the current bundle at iteration i.

It is difficult to solve the quadratic problem (4.4) directly because m is, in general, large; the associated dual problem (QP)

$$\alpha \in \underset{\alpha \geq 0,\ \sum_{\bar{f} \in J^i} \alpha_{\bar{f}} = 1}{\arg\max} \left(\sum_{\bar{f} \in J^i} \alpha_{\bar{f}} \bar{f}(\bar{\lambda}^i) - \frac{1}{2u^i} \|g^i(\alpha)\|^2 \right), \tag{4.5}$$

however, has smaller dimension.

The components of α are the coefficients of a convex combination of elements of J. That is $\alpha_{\bar{f}} \geq 0$ for all $\bar{f} \in J^i$ and $\sum_{\bar{f} \in J^i} \alpha_{\bar{f}} = 1$.

The *aggregated subgradient* $g^i(\alpha)$, or shortly g^i, is defined by a solution α^i of (4.5):

$$g^i(\alpha^i) := \sum_{\bar{f} \in J^i} \alpha^i_{\bar{f}} \nabla \bar{f}. \tag{4.6}$$

Here $\nabla \bar{f} = \nabla \bar{f}(\lambda)$ is the unique gradient for $\bar{f}$ at any point λ because $\bar{f}$ is affine. If $\bar{f} = \bar{f}_{\lambda^i}$, i.e., $\bar{f}$ stems from a linearization of type (4.1), then is $\nabla \bar{f} = g_{\bar{f}}(\lambda^i)$.

The proof that (4.5) is the dual of (4.4) is shown as the proof of Lemma 4.1.

Lemma 4.1. Problems (4.4) and (4.5) are strongly dual: They have the same optimal solution value if they are both feasible. If (4.4) is infeasible then (4.5) is unbounded, and vice versa.

Proof. We rewrite the quadratic model (4.4) as follows:

$$\min_{\lambda} \hat{f}_{J^i}(\lambda) + \frac{u^i}{2}\|\lambda - \bar{\lambda}^i\|^2 = \min_{\lambda} \max_{\bar{f}\in J^i} \bar{f}(\lambda) + \frac{u^i}{2}\|\lambda - \bar{\lambda}^i\|^2 \tag{4.7}$$

$$= \min_{\lambda, z: \bar{f}(\lambda)\le z, \forall \bar{f}\in J^i} z + \frac{u^i}{2}\|\lambda - \bar{\lambda}^i\|^2. \tag{4.8}$$

Denoting the $k := |J^i|$ elements of J^i by $\bar{f}_1, \ldots, \bar{f}_k$ and setting

$$c := \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad Q := u^i \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \\ 0 & 0 & & 1 \end{pmatrix}, \quad A := \begin{pmatrix} 1 & (\nabla \bar{f}_1)^\mathsf{T} \\ \vdots & \vdots \\ 1 & (\nabla \bar{f}_k)^\mathsf{T} \end{pmatrix},$$

$$b := \begin{pmatrix} \bar{f}_1(0) + \nabla \bar{f}_1{}^\mathsf{T} \bar{\lambda}^i \\ \vdots \\ \bar{f}_k(0) + \nabla \bar{f}_k{}^\mathsf{T} \bar{\lambda}^i \end{pmatrix} = \begin{pmatrix} \bar{f}_1(\bar{\lambda}^i) \\ \vdots \\ \bar{f}_k(\bar{\lambda}^i) \end{pmatrix}, \text{ and } x := \begin{pmatrix} z \\ \lambda - \bar{\lambda}^i \end{pmatrix},$$

(4.8) can be written as:

$$\begin{aligned} \min \;\; & c^\mathsf{T} x + \tfrac{1}{2} x^\mathsf{T} Q x\,, \\ \text{s.t.} \;\; & Ax \ge b. \end{aligned}$$

Its dual according to Section 3.2.3 or Dorn [1960] is then

$$\begin{aligned} \text{(DQ)} \max \; & \alpha^\mathsf{T} b - \tfrac{1}{2} x^\mathsf{T} Q x, \\ \text{s.t.} \; & A^\mathsf{T}\alpha + Qx \;\; = c, \\ & \alpha \qquad\qquad \ge 0. \end{aligned}$$

This is equivalent to

$$\begin{aligned} \max \; & \sum_{\bar{f}\in J^i} \alpha_{\bar{f}} \bar{f}(\bar{\lambda}^i) - \tfrac{u^i}{2}\|\lambda - \bar{\lambda}^i\|^2, \\ \text{s.t.} \; & \sum_{\bar{f}\in J^i} \alpha_{\bar{f}} = 1, \\ & g^i(\alpha) + u^i(\lambda - \bar{\lambda}^i) = 0, \\ & \alpha \ge 0. \end{aligned}$$

Using the second equality to replace $\bar{\lambda}^i - \lambda$ by $\frac{g^i(\alpha)}{u^i}$ proves the claim. □

Lemma 4.2. If $\alpha^i \in \mathbb{R}^{J^i}_+$ is an optimal solution of (4.5) then

$$\lambda^{i+1} = \bar{\lambda}^i - \frac{1}{u^i} g^i(\alpha^i)$$

is an optimal solution of (4.4).

Proof. The theorem about complementary slack implies that

$$\alpha^i_{\bar{f}} = 0 \text{ or } \bar{f}(\lambda^{i+1}) = \hat{f}_{J^i}(\lambda^{i+1}). \tag{4.9}$$

Using this it follows that

$$\sum_{\bar{f}\in J^i} \alpha^i_{\bar{f}} \bar{f}(\bar{\lambda}^i) - \frac{1}{2u^i}\|g^i(\alpha^i)\|^2 = \hat{f}_{J^i}(\lambda^{i+1}) + u^i\|\lambda^{i+1} - \bar{\lambda}^i\|^2/2.$$

By Lemma 4.1 now follows that λ^{i+1} is an optimal solution of (4.4). □

The model value at the new trial point can be directly expressed by the linearizations in the current bundle J^i and their convex multipliers α^i. This is shown in the next corollary:

Corollary 4.3. Let the cutting plane model $\hat{f}_{J^i}$ of f be defined by (4.3) and let α be an optimal solution of problem (4.5), then the following holds

$$\hat{f}_{J^i}(\lambda^{i+1}) = \sum_{\bar{f}\in J^i} \alpha_{\bar{f}} \bar{f}(\lambda^{i+1}).$$

Proof. This follows directly from the definition of $\hat{f}_{J^i}$ and (4.9). □

The aggregated subgradient $g^i(\alpha^i)$ and the model value at the new trial point $\hat{f}_{J^i}(\lambda^{i+1})$ define an affine function

$$\tilde{f}^{i+1}(\lambda) := \hat{f}_{J^i}(\lambda^{i+1}) + g^i(\alpha^i)^\mathsf{T}(\lambda - \lambda^{i+1}) \tag{4.10}$$

that minorizes by construction $\hat{f}_{J^i}$ and therefore also f. Hence $g^i(\alpha^i)$ is a subgradient of $\hat{f}$ at λ^{i+1}.

The stopping criterion of the PBM is the predicted descent $v^i := f(\bar{\lambda}^i) - \hat{f}(\lambda^{i+1})$ in the objective value by moving from $\bar{\lambda}^i$ to λ^{i+1}. The predicted descent is non-negative and if $v^i = 0$, then $\bar{\lambda}^i$ is already an optimal solution of our problem.

Problem (4.5) can be interpreted as finding a convex combination of the affine functions in J^i that has an aggregated subgradient with small norm and a large function value at the current stability center. It is easy too see that every convex combination of affine functions below f is again an affine function minorizing f. We will later explain how this is used to keep the number of elements in J limited in the course of the PBM.

4.1.4 Algorithm

Algorithm 3 states the (generic) PBM of Kiwiel [1990] as pseudo-code. Step 1 initializes the counters, the bundle, and the starting trial point λ^0. This point could be set arbitrarily, however a good estimation can reduce the computation time significantly. In Step 2 a quadratic problem has to be solved to find a new trial point. Then it is tested, if the current trial points is good enough. Step 4 decides whether the stability center changes or not by the predicted descent v^i. The predicted descent depends on the weight u_i. If u_i is large, the distance to the stability center is weighted heavier and therefore the new trial point is closer to the trial point with increasing u_i. Therefore also v_i is smaller. This test ensures that the descent of the function value is large enough to guarantee the convergence of the PBM (see Section 4.1.6). In Step 5 the new linearization is calculated and added to the bundle. Step 6 updates the weight of the quadratic term of the quadratic model. This step is optional. At last we increase the iteration counter and go back to Step 2.

Algorithm 3 PBM

Input: A convex and continuous function $f : \mathbb{R}^m \to \mathbb{R}$, a stopping tolerance ϵ, and a parameter $m_L \in (0, 0.5]$, which controls the frequency of serious steps.

Output: A point $\lambda \in \mathbb{R}^m$ and an objective value $f(\lambda)$.

1: *(Initialization)* Set the iteration-counter $i \leftarrow 0$, select a "good" starting solution λ^i, calculate a linearization $\bar{f}^i$ at λ^i, initialize the bundle $J^i \leftarrow \{\bar{f}^i\}$, set the stability center $\bar{\lambda}^i \leftarrow \lambda^i$. Set the weight $u^i \leftarrow 1$.
2: *(Direction finding)* Solve problem (4.5). This results in convex multipliers α^i, a new aggregated subgradient $g^i(\alpha^i)$, and a new trial point λ^{i+1}.
3: *(Stopping criterion)* If $f(\bar{\lambda}^i) - \hat{f}_{J^i}(\lambda^{i+1}) < (1 + |f(\bar{\lambda}^i)|)\epsilon$ terminate.
4: *(Descent test)* If $f(\lambda^{i+1}) \leq f(\bar{\lambda}^i) - m_L(f(\bar{\lambda}^i) - \hat{f}_{J^i}(\lambda^{i+1}))$ then $\bar{\lambda}^{i+1} \leftarrow \lambda^{i+1}$ else $\bar{\lambda}^{i+1} \leftarrow \bar{\lambda}^i$.
5: *(Linearization updating)* Select J^{i+1} containing a linearization of f at λ^{i+1}, the aggregated linearization $g^i(\alpha^i)$, and arbitrary elements of J^i.
6: *(Weight updating)* Select a weight u^{i+1}.
7: $i \leftarrow i + 1$.
8: Go to Step 2.

4.1.5 Weight updating

Any constant $u^i > 0$ suffices to guarantee convergence of the PBM. However, in the PBM-version of Kiwiel the weight u^i depends, as indicated by notation, on the iteration i. The adaption of the weight in the course of the algorithm can lead to speedups.

Kiwiel [1990] proves that any sequence of weights that only increases at iterations with serious steps and is constant or decreasing otherwise leads to a convergent algorithm. In this article also a weight updating scheme based on a quadratic approximation of f is proposed. In our case a simpler scheme worked better: We increase the weight by a factor of 1.5 if fewer than i_{serious} iterations occurred between two serious steps, and we decrease the weight by a factor of 0.75 if more than i_{null} null steps in a row occurred. The values of i_{serious} and i_{null} were set empirically for each class of problems.

4.1.6 Notes On The Convergence

We will sketch the proof that the sequence of stability centers $(\bar{\lambda}^i)$ generated by the PBM converges to a point in the set $\arg\min_{\lambda\in\mathbb{R}^m} f(\lambda)$ if this set is not empty and if the stopping tolerance ϵ is set to zero. As far as we know, Kiwiel [1990] gives the first proof of convergence of this kind of bundle methods, see also Hiriart-Urruty & Lemaréchal [1993a] for a deep examination of bundle methods including a convergence proof, or Frangioni [2002] for a proof of convergence of a generalized class of bundle algorithms including the PBM.

The proof consists in all cases of the following steps: It is shown that the algorithm stops with an optimal λ^i if it terminates after a finite number of iterations. That is: it is shown that if at a certain iteration i we have $f(\bar{\lambda}^i) = \hat{f}(\lambda^{i+1})$and hence $\bar{\lambda}^i$ is optimal.

Then it is shown that for a constant stability center $\bar{\lambda}^i$ (assuming that at a certain point in the course of the algorithm an infinite series of null steps occurs) the algorithm converges to an optimum of the quadratic subproblem (4.4). That is

$$\lambda^i \to \arg\min_{\lambda}\max_{x} L(\lambda, x) = \arg\min_{\lambda}\left(f(\lambda) + \frac{u^i}{2}\|\lambda - \bar{\lambda}^i\|^2\right).$$

Further it is shown that v^i, that is, the gap between model value $\hat{f}(\lambda^i)$ and function value $f(\lambda^i)$, converges to zero. This implies that either a descent step is triggered after a finite number of iterations or the stability center is

already optimal. Finally it is proven that in the case of an infinite number of descent steps the sequence of stability centers has an optimal cluster point. Denote: if f is polyhedral, as in our setting, we have only a finite number of serious steps.

The actual proof needs some concepts of convex analysis, which are not the topic of this work, or alternatively a lengthy technical proof, which in our opinion does not give new insight to the PBM, therefore we do not want to go into the details here.

4.2 Comparison with other Subgradient Methods

The proximal bundle method is related to the subgradient method, which was proposed by Shor in 1962. The subgradient method was used by Held & Karp [1970, 1971] to solve a Lagrangean relaxation of the traveling salesman problem. Since then the subgradient method is often used to calculate bounds on the optimal values and dual information of LPs and combinatorial problems by approximating their Lagrangean relaxations. The advantages of the subgradient method are that it is easy to implement in comparison to interior point or simplex methods and that it is generally very fast in calculating bounds on the objective value of LPs. A comparison of calculating bounds of various LPs with CPLEX, PBM, and subgradient methods can be found in Section 4.7.

A major disadvantage is that subgradient methods in general are not able to give lower bounds of the same quality as the simplex or the barrier algorithm. They also lack a good termination criterion because the quality of the lower bounds found cannot be controlled. In addition, the convergence speed of the subgradient method is usually rather slow if the objective value approaches its optimum. Finally, the subgradient method does not provide primal information which can be useful as input for primal heuristics.

These disadvantages are addressed by the PBM. It comes at the price that at every iteration of the PBM a quadratic problem has to be solved. Solving this quadratic problem is rather difficult and time consuming. However, the PBM needs in general fewer iterations to reach the same objective value as the subgradient method hence it is often faster with respecting the total running time.

4.3 Modifications and Extensions

In the next sections we describe modifications of the PBM that either intend to accelerate it for functions with special structure or that enhance its usefulness. We describe in Section 4.3.1 the technique of Kiwiel [1995] to handle separable functions more efficiently. We need this to approximate LPs with block-structure like the LP-relaxation of (ISP). In Section 4.3.2 we explain a straightforward method to recover primal information if the function f stems from the Lagrangean relaxation of an LP or ILP. We need the primal information to guide the branching decisions of our ILP-heuristic of Chapter 6. In Section 4.4 we show our active set technology to accelerate the PBM if f stems from the Lagrangean relaxation of an LP with many columns. Section 4.3.3 shows the handling of functions with restricted domain. This technique was developed by Helmberg & Kiwiel [2002] for problems stemming from semi-definite optimization problems. We adapt it to approximate LPs with inequality constraints.

4.3.1 Separable Functions

The PBM can handle separable functions in a particularly efficient way to speed up the convergence. The idea is to provide for each subfunction its own bundle. We now formalize this:

Let the function $f : \mathbb{R}^m \to \mathbb{R}$, which we want to minimize, be the dual function of a separable function. That is: we want to maximize a separable primal problem

$$\begin{aligned} \text{(SP)} \max \quad & \textstyle\sum_{j=1}^{k} \zeta_j(x_j), \\ \text{s.t.} \quad & \\ & \textstyle\sum_{j=1}^{k} \phi_{ij}(x_j) \geq 0,\ j = 1, \ldots, m, \\ & x = (x_1, x_2, \ldots, x_k) \in X := X_1 \times X_2 \times \cdots \times X_k \end{aligned}$$

where ζ_j and ϕ_{ij} are closed concave functions on nonempty closed convex sets. Denote: This is exactly the setting of the ISP.

The dual function that to (SP) that should be minimized is then

$$f(\lambda) := \sup\left\{\sum_{j=1}^{k}\left(\zeta_j(x_j) + \sum_{i=1}^{m}\lambda_i\phi_{ij}(x_j)\right) : x = (x_1,\ldots,x_k) \in X\right\}$$
$$= \sum_{j=1}^{k}\sup_{x_j\in X_j}\left(\zeta_j(x_j) + \sum_{i=1}^{m}\lambda_i\phi_{ij}(x_j)\right).$$

Here the Lagrangean multipliers are $\lambda \in \mathbb{R}^m$, and we will write in the following

$$f_j(\lambda) := \sup_{x_j\in X_j}\left(\zeta_j(x_j) + \sum_{i=1}^{m}\lambda_i\phi_{ij}(x_j)\right).$$

Then f_j is convex and continuous for all $j = 1,\ldots,k$.

The PBM is able to manage for every function f_j its own bundle J_j^i of linearizations. Problem (4.5) can be written as

$$\alpha^i = \underset{\substack{\alpha\geq 0,\\ \sum_{\bar{f}\in J_j^i}\alpha_{\bar{f}}^j = 1,\forall j=1,\ldots,k}}{\arg\min}\left(\sum_{j=1}^{k}\sum_{\bar{f}\in J_j^i}\alpha_{\bar{f}}^j\bar{f}(\bar{\lambda}^i) + \frac{1}{u^i}\|\sum_{j=1}^{k}g_j(\alpha^j)\|^2\right) \tag{4.11}$$

and the aggregated subgradient $g_j(\alpha)$ of function f_j over the bundle J_j^i is defined analogously to $g(\alpha)$:

$$g_j(\alpha) := \sum_{\bar{f}\in J_j^i}\alpha_{\bar{f}}^j g_{\bar{f}}.$$

The special handling of separable functions makes the Direction finding Step 2 of Algorithm 3 more expensive, but in general reduces the total number of iterations needed. A more complete description of this technique can be found in Kiwiel [1995].

4.3.2 Primal Approximation of Linear Programs

If f is defined by a Lagrangean relaxation of a linear program we are able to approximate a primal solution of it. We show in this section how this can be

accomplished. Let us look at the following linear program:

$$\begin{aligned} &(\mathrm{P}) \quad \max\ c^{\mathsf{T}}x, \\ &\qquad \text{s. t.} \\ &(\mathrm{i}) \qquad Ax = a, \\ &(\mathrm{ii}) \qquad Bx = b, \\ &(\mathrm{iii}) \qquad x \geq 0. \end{aligned}$$

Here $A \in \mathbb{R}^{m_1 \times n}$ and $B \in \mathbb{R}^{m_2 \times n}$ are matrices and $c, x \in \mathbb{R}^n$ and $a \in \mathbb{R}^{m_1}$, $b \in \mathbb{R}^{m_2}$ are appropriate vectors.

We relax constraints (i) by Lagrangean relaxation resulting in the following problem (L):

$$\min_{\lambda \in \mathbb{R}^{m_1}} \max_{Bx=b, x\geq 0} c^{\mathsf{T}}x + \lambda^{\mathsf{T}}(a - Ax). \tag{4.12}$$

The convex continuous function $f : \mathbb{R}^{m_1} \to \mathbb{R}$ which we want to minimize is defined by

$$\begin{aligned} f(\lambda) :&= \max_{Bx=b, x\geq 0} c^{\mathsf{T}}x + \lambda^{\mathsf{T}}(a - Ax) \\ &= \lambda^{\mathsf{T}}a + \max_{Bx=b, x\geq 0} (c^{\mathsf{T}} - \lambda^{\mathsf{T}}A)x. \end{aligned}$$

To evaluate f at a certain trial point λ^* we have to solve the problem

$$\arg\max_{Bx=b,\ x\geq 0} (c^{\mathsf{T}} - \lambda^{*\mathsf{T}}A)x. \tag{4.13}$$

If this problem is significantly easier to solve than the original problem (P), the PBM is a reasonable approach to solve problem (L).

If x^* is a solution of (4.13) a linearization of f at $\lambda \in \mathbb{R}^{m_1}$ is given by

$$\bar{f}(\lambda) := c^{\mathsf{T}}x^* + \lambda^{\mathsf{T}}(a - Ax^*), \quad \forall \lambda \in \mathbb{R}^{m_1} \tag{4.14}$$

and the subgradient of f at λ^* by $a - Ax^*$. Observe that in general x^* is not unique and therefore the subgradient is also not unique.

The primal approximation of (P) can now be calculated by convex combinations of solutions of (4.13) associated with the linear functions in the bundle J^i and the solution α^i of the according problem (4.5). That is: Let $x_{\bar{f}}$ be the solution of (4.13) that leads to the linearization $\bar{f}$, or if $\bar{f}$ is the result of a convex combination of such linearizations, let $x_{\bar{f}}$ be the convex

combination of the according primal information associated with the linear functions. Then we define

$$x(\alpha^i) := \sum_{\bar{f} \in J^i} \alpha^i_{\bar{f}} x_{\bar{f}}. \tag{4.15}$$

If we only want to calculate a primal solution of a certain subfunction f_j of a separable function f stemming from the Lagrange-relaxation of a LP with block-structure, $x_j(\alpha^i)$ is defined analogously to $g_j(\alpha^i)$:

$$x_j(\alpha^i) := \sum_{\bar{f} \in J^i_j} \alpha^j_{\bar{f}} x_{\bar{f}}.$$

The series $(x(\alpha^i))_{i=1,2,\ldots}$ converges to an optimal solution of (P) (see Kiwiel [1990]).

4.3.3 Handling of Bounded Functions

The bundle method can also be used to minimize a convex function f over a box. That is we look at the problem

$$\min_{\ell \le \lambda \le u} f(\lambda). \tag{4.16}$$

We use this for the Lagrangian multipliers associated with the base constraints of problem (DSP), see Section 1.10, (dual-DSP). Only Step 2, Direction Finding, of Algorithm 3 has to be adapted. We introduce in the following the idea and a description of our implementation. Mathematical proofs and further details for a similar method used in semidefinite programming can be found in Helmberg & Kiwiel [2002].

We write Problem (4.16) as the unbounded problem

$$\min_{\lambda} f(\lambda) + \imath_{[\ell,u]}(\lambda). \tag{4.17}$$

The indicator function $\imath_Q(\lambda)$ is 0 if $\lambda \in Q$ and ∞ otherwise. Let us also define an interval of vectors coordinatewise: Let $\ell, u \in \mathbb{R}^m$, then $[\ell, u] := \{\lambda \in \mathbb{R}^m \mid \ell_i \le \lambda_i \le u_i, \forall i \in \{1, \ldots, m\}\}$. The indicator function can then be described by

$$\imath_{[\ell,u]}(\lambda) = \max_{\eta^-, \eta^+ \in \mathbb{R}^m_+} (\ell - \lambda)^\mathsf{T} \eta^- + (\lambda - u)^\mathsf{T} \eta^+ \tag{4.18}$$

A linearization of f analogously to equation (4.1) at a point λ^i is:

$$\bar{f}_{\lambda^i}(\lambda) := f(\lambda^i) + (g_{\bar{f}_{\lambda^i}} - \eta^-_{\lambda^i} + \eta^+_{\lambda^i})^\mathsf{T}(\lambda - \lambda^i) + (\ell - \lambda^i)^\mathsf{T}\eta^-_{\lambda^i} + (\lambda^- - u)^\mathsf{T}\eta^+_{\lambda^i} \tag{4.19}$$

$$= f(\lambda^i) + g_{\bar{f}_{\lambda^i}}{}^\mathsf{T}(\lambda - \lambda^i) + (\ell - \lambda)^\mathsf{T}\eta^-_{\lambda^i} + (\lambda - u)^\mathsf{T}\eta^+_{\lambda^i} \tag{4.20}$$

with

$$\eta^-_{\lambda^i} \in \arg\max_{\eta \geq 0}(\ell - \lambda^i)^\mathsf{T}\eta \text{ and} \tag{4.21}$$

$$\eta^+_{\lambda^i} \in \arg\max_{\eta \geq 0}(\lambda^i - u)^\mathsf{T}\eta. \tag{4.22}$$

The variables η^- and η^+ can be seen as Lagrangean multipliers of the lower and upper bounds on λ.

Now we define a function that is always smaller than or equal to the linearization $\bar{f}_{\lambda^i}$ by selecting arbitrary non-negative values for $\eta^+_{\lambda^i}$, and $\eta^-_{\lambda^i}$:

$$\bar{f}_{\lambda^i}(\lambda, \eta^+, \eta^-) := f(\lambda^i) + g_{\bar{f}_{\lambda^i}}{}^\mathsf{T}(\lambda - \lambda^i) + (\ell - \lambda)^\mathsf{T}\eta^- + (\lambda - u)^\mathsf{T}\eta^+. \tag{4.23}$$

This can also be written as

$$\bar{f}_{\lambda^i}(\lambda, \eta^+, \eta^-) = f(\lambda^i) - g_{\bar{f}_{\lambda^i}}{}^\mathsf{T}\lambda^i + \ell^\mathsf{T}\eta^- + u^\mathsf{T}\eta^+ + (g_{\bar{f}_{\lambda^i}} - \eta^- + \eta^+)^\mathsf{T}\lambda.$$

One can see that for fixed η^- and η^+ the function $\bar{f}(\cdot, \eta^+, \eta^-)$ is affine, has the gradient $g_{\bar{f}_{\lambda^i}} - \eta^- + \eta^+$ and at the origin the function value $f(\lambda^i) - g_{\bar{f}_{\lambda^i}}{}^\mathsf{T}\lambda^i + \ell^\mathsf{T}\eta^- + u^\mathsf{T}\eta^+$. Obviously the following holds by construction of these functions:

$$f(\lambda) \geq \bar{f}_{\lambda^i}(\lambda) \geq \bar{f}_{\lambda^i}(\lambda, \eta^+, \eta^-), \quad \forall \lambda, \lambda^i \in \mathbb{R}^m, \eta^+, \eta^- \in \mathbb{R}^m_+.$$

We define a family of cutting plane models $\hat{f}^{\eta^+,\eta^-}_J$ of f using its affine minorants $\bar{f}_{\lambda^i}(\cdot, \eta^+, \eta^-)$:

$$\hat{f}^{\eta^+,\eta^-}_J(\lambda) := \max_{\bar{f} \in J} \bar{f}(\lambda, \eta^+, \eta^-) \tag{4.24}$$

and a corresponding family of quadratic models

$$\tilde{f}^{\eta^+,\eta^-}_J(\lambda, \bar{\lambda}^i) := \hat{f}^{\eta^+,\eta^-}_J(\lambda) + \frac{w}{2}\|\lambda - \bar{\lambda}^i\|^2. \tag{4.25}$$

We optimize over the family of functions $\tilde{f}^{\eta^+,\eta^-}_J$ in a two stage approach. First, we keep the pair (η^+, η^-) fixed and calculate a minimizer $\lambda^{i+1}_{\eta^+,\eta^-}$ of (4.25). That is, analogously to (4.5) we want to solve

$$\alpha^i = \arg\max_{\alpha \geq 0, \sum_{\bar{f} \in J^i} \alpha_{\bar{f}} = 1} \left(\sum_{\bar{f} \in J^i} \alpha_{\bar{f}} \bar{f}(\bar{\lambda}^i, \eta^+, \eta^-) - \frac{1}{2w^i}\|g(\alpha) - \eta^- + \eta^+\|^2 \right) \tag{4.26}$$

and

$$\lambda^{i+1} = \bar{\lambda}^i - \frac{1}{w^i}\left(g(\alpha^i) - \eta^- + \eta^+\right). \tag{4.27}$$

Then we find the new $(\eta^{+,i+1}, \eta^{-,i+1})$ by fixing λ to λ^{i+1} and solving the problem

$$(\eta^{+,i+1}, \eta^{-,i+1}) := \underset{\eta^+\geq 0, \eta^-\geq 0}{\arg\min}\ \tilde{f}_J^{\eta^+,\eta^-}(\lambda^{i+1}). \tag{4.28}$$

Using (4.27) we get

$$\eta^{+,i+1} := \max\{0, -w^i\bar{\lambda}^i - g(\alpha^i) - w^i u\}, \tag{4.29}$$

$$\eta^{-,i+1} := \max\{0, w^i\bar{\lambda}^i + g(\alpha^i) + w^i \ell\}. \tag{4.30}$$

The new point $(\lambda^{i+1}, \eta^{+,i+1}, \eta^{-,i+1})$ together with the aggregated linearization defined by $g(\alpha^i)$ and $f(\lambda^{i+1})$ would be already enough to ensure convergence of the bundle method. However, alternatively solving (4.26) with fixed η^-, η^+ and (4.28) with a fixed α^i can further improve the next trial point. The direction finding part of the algorithm implementing this idea is presented as Algorithm 4.

Algorithm 4 Direction Finding for bounded functions.

Input: A function f, a bundle J, a stability center $\bar{\lambda}^i$, Lagrangean multipliers for the box constraints $\eta^{+,i}$ and $\eta^{-,i}$, a weight w^i. A maximum number of iterations in the Direction Finding $j_{\max}$, a threshold ϵ.

Output: A new trial point λ^{i+1}, and new variables $\eta^{+,i+1}$, $\eta^{-,i+1}$.

1: $f(\lambda_{-1}) = -\infty$, $j \leftarrow 0$, $k \leftarrow 0$, $\eta_j^+ \leftarrow \eta^{+,i}$, $\eta_j^- \leftarrow \eta^{-,i}$.
2: **repeat**
3: Solve problem (4.26) with respect to η_k^+, η_k^-.
4: Set η_{j+1}^+ and η_{j+1}^- according to (4.29) and (4.30).
5: Set $\lambda_j \leftarrow \bar{\lambda}^i - \frac{1}{w^i}\left(g(\alpha^i) - \eta_j^- + \eta_j^+\right)$.
6: **if** $\hat{f}_J^{0,0}(\lambda_j) - \hat{f}^{\eta_{j+1}^+,\eta_{j+1}^-}(\lambda_j) > 0.1(f(\bar{\lambda}^i) - \hat{f}\eta_{j+1}^+, \eta_{j+1}^-(\lambda_j))$ **then** set $k \leftarrow j$.
7: $j \leftarrow j + 1$.
8: **until** $j = j_{\max}$ or $f(\lambda_{j-1}) - f(\lambda_{j-2}) < \epsilon$

4.4 Active Set Method

Active set methods are a general technique for algorithms that solve MIPs or LPs. The idea is to select a subset of columns of the problem and then

to make the computations on the smaller problem induced by this subset. The active set will be adapted throughout the course of the algorithm. We describe in this section our active set method for the PBM.

For set covering and set partitioning problems an active set heuristic is described by Caprara et al. [1996]. Similar methods are also used in simplex algorithms under the name sifting, e.g., in the standard solver CPLEX. The restricted problem of a column generation approach (see Section 3.1) can also be interpreted as an active set of the master problem.

4.4.1 Description

We look at an LP of the following type:

$$\begin{aligned} &(\mathrm{P}) \quad \max \quad c^{\mathsf{T}}x, && c \in \mathbb{R}^n, x \in \mathbb{R}^n, \\ &\qquad \text{s.t.} \\ &(\mathrm{i}) && Ax = b,\ A \in \mathbb{R}^{m \times n}, \\ &(\mathrm{ii}) && 0 \leq x \leq \mathbb{1}. \end{aligned}$$

We relax constraints of type (i) in a Lagrangean way to get the Lagrangean dual function $f : \mathbb{R}^m \mapsto \mathbb{R}$ defined by

$$f(\lambda) := \lambda^{\mathsf{T}} b + \max_{0 \leq x \leq 1} (c - \lambda^{\mathsf{T}} A)x. \tag{4.31}$$

We now want to minimize f by the PBM.

If n is very much larger than m the computation of $\lambda^{\mathsf{T}} A$ consumes the main part of the evaluation of f at a certain point $\lambda \in \mathbb{R}^m$. Therefore we use at an iteration i of the PBM a subset $I^i \subset \{1, \ldots, n\}$ of all columns called *active set*. The active set I implies a function f_I by replacing the matrix A in the definition of f by a matrix $A_{\cdot I}$ restricted to columns in I and by replacing x with x_I. Then obviously $f_I(\lambda) \leq f(\lambda)$ holds. It is easy to see that $f_I(\lambda) = f(\lambda)$ if the reduced cost $\bar{c}_j := c_j - \lambda^{\mathsf{T}} A_{\cdot j}$ are non-positive for all columns j which are not included in I.

The selection of the right active set is critical to the performance of the PBM with active sets, because large active sets consume much time evaluating f_I and missing columns in the active sets lead to additional iterations of the PBM in which the active set is corrected. We use the following strategy to select the active sets: The active set is only adapted at serious steps of the PBM or if a termination is possible. The rationale behind this is that

at null steps the Lagrangean multipliers only change slightly because the quadratic stabilization term keeps the new trial point in the vicinity of the stability center. Therefore also the active set remains meaningful for the actual problem.

Algorithm 5 PBM with active set

Input: A convex and continuous function $f : \mathbb{R}^m \to \mathbb{R}$, a stopping tolerance ϵ, and a parameter $m_L \in (0, 0.5]$, which controls the frequency of serious steps.

Output: A point $\lambda \in \mathbb{R}^m$ and an objective value $f(\lambda)$.

1: *(Initialization)* Set the iteration-counter $i \leftarrow 0$, select a "good" starting solution λ^i, select an active set $I^i \subset \{1, \ldots, n\}$, calculate a linearization $\bar{f}^i_{I^i}$ at λ^i, initialize the bundle $J^i \leftarrow \{\bar{f}^i_{I^i}\}$, set the stability center $\bar{\lambda}^i \leftarrow \lambda^i$. Set the weight $u^i \leftarrow 1$.
2: *(Direction finding)* Solve problem (4.5).
3: Set $I^{i+1} \leftarrow I^i$.
4: **if** $f_{I^i}(\bar{\lambda}^i) - \hat{f}_{J^i}(\lambda^{i+1}) < (1 + |f(\bar{\lambda}^i)|)\epsilon$ **then**
5: **if** $f(\bar{\lambda}^i) - \hat{f}_{J^i}(\lambda^{i+1}) < (1 + |f(\bar{\lambda}^i)|)\epsilon$ **then**
6: terminate.
7: **else**
8: Adapt the active set I^{i+1}.
9: **end if**
10: **end if**
11: **if** $f_{I^i}(\lambda^{i+1}) \leq f(\bar{\lambda}^i) - m_L(f(\bar{\lambda}^i) - \hat{f}_{J^i}(\lambda^{i+1}))$ **then**
12: **if** $f(\lambda^{i+1}) \leq f(\bar{\lambda}^i) - m_L(f(\bar{\lambda}^i) - \hat{f}_{J^i}(\lambda^{i+1}))$ **then**
13: $\bar{\lambda}^{i+1} \leftarrow \lambda^{i+1}$.
14: Remove some elements from the active set I^{i+1}.
15: **else**
16: $\bar{\lambda}^{i+1} \leftarrow \bar{\lambda}^i$.
17: Adapt the active set I^{i+1}.
18: **end if**
19: **else**
20: $\bar{\lambda}^{i+1} \leftarrow \bar{\lambda}^i$.
21: **end if**
22: *(Linearization updating)* Select J^{i+1} containing a linearization of f_{I^i} at λ^{i+1}, the aggregated linearization $g^i(\alpha^i)$, and arbitrary elements of J^i.
23: *(Weight updating)* Select a weight u^{i+1}.
24: $i \leftarrow i + 1$.
25: Go to Step 2.

Algorithm 5 shows the PBM with active sets as pseudo-code. It has the

following differences to the PBM as shown in Algorithm 3: If at an iteration i the criterion for an abortion is fulfilled by f_{I^i} the method checks at step 4 whether this criterion is also fulfilled by f. If this is not the case that implies that $f_{I^i}(\lambda^{i+1}) < f((\lambda^{i+1})$ and this is caused by columns with positive reduced cost in $\bar{I}^i := \{1, \ldots, n\} \setminus I^i$. Then we add columns with positive reduced cost from $\bar{I}^i$ to I^{i+1}. We keep also all columns with positive reduced cost that are already in I^i.

If at an iteration i the criterion for a serious step is fulfilled by f_{I^i} the method checks at step 12 if this criterion is also fulfilled by f. If this is the case we only remove some columns from I^{i+1} with non-positive reduced cost. Otherwise, again some columns with positive reduced cost are missing in the current active set, so some are added. If the criterion for serious steps is not fulfilled by f_{I^i}, a null step is performed without changing the active set.

In each iteration, we add only a certain number of columns per row of matrix A to the active set. We select per row the five columns with largest non-negative reduced cost that cover this row, add add them to I^{i+1}. This keeps the number of columns in the active set small even though the absolute number of columns with positive reduced cost may be large.

If the active set changes it is possible that the linearizations in the current bundle J^i are no longer valid for the new function $f_{I^{i+1}}$. This could be prevented by removing all linearizations from the bundle at serious steps, and the PBM would still converge (see Kiwiel [1990]). However, this slows down the PBM. So we keep the bundle and "shift" the linearizations if we detect an invalid one. The details of this strategy will be described in Section 4.6.

Our modified PBM still converges, because n is finite and we add columns to the active set if we recognize that some are missing, and we only remove columns after "real" serious steps. Therefore it is still guaranteed that either a serious step occurs or the abortion criterion is fulfilled after a finite number of iterations.

4.4.2 Exact Active Sets

We now propose an exact method for selecting columns into the active set, such that no column not included can get positive reduced cost until the next serious step of the PBM.

For given Lagrangean multipliers $\lambda^i \in \mathbb{R}^m$ it is easy to solve the problem

$$\max_{0 \le x \le 1} c^\mathsf{T} x + (b - Ax)^\mathsf{T} \lambda^i. \tag{4.32}$$

Any optimal solution x^* of (4.32) depends only on the reduced cost $\bar{c}^i$. All x^* of the following form are optimal solutions of (4.32): Let $j = 1, \ldots, n$, then let

$$x_j^* := \begin{cases} 0, & \bar{c}_j^i > 0, \\ 1, & \bar{c}_j^i < 0, \\ \text{an arbitrary value in } [0,1], & \bar{c}_j^i = 0. \end{cases}$$

In the PBM we find at each iteration a new trial point λ^{i+1} in the vicinity of the stability center $\bar{\lambda}^i$. This implies, that the reduced costs $\bar{c}^{i+1}$ are also staying in the region of the reduced cost of the last stability center. More precisely, let i^s be the iteration of the last serious step, then it holds using $\lambda^{i+1} = \lambda^{i^s} - \frac{1}{u^i} g^i$:

$$\bar{c}^{i+1} = c - A^\mathsf{T} \lambda^{i^s} - \frac{1}{u^i} A^\mathsf{T} g^i.$$

We define now an upper bound β^i on the 1-norm of the aggregated subgradients of all iterations after an iteration i and before the next serious step at an iteration i^*:

$$\beta^i \geq \sum_{j=1}^{m} |g_j^\ell|, \ \forall i \leq \ell \leq i^*.$$

Then we know that the reduced cost of a column j changes in the iterations until the next serious step at most by $\frac{\beta^i \|A_{\cdot j}\|}{u^i}$. Here we take the 1-norm $\|A_{\cdot j}\| := \sum_{i=1}^{m} |A_{ij}|$. This follows directly from Cauchy-Schwarz.

If now at a serious step of the PBM in iteration k the reduced cost of a column j is smaller than $-\frac{\beta^i \|A\|}{u^i}$ we know, that until the next serious step the reduced cost $\bar{c}_j^\ell, \ \ell > k$ will be smaller than zero. This means, that the variable of the corresponding column will be staying at zero. Thus we can safely ignore this column. Therefore we only take the columns in an active set $I^i := \{j \in \{1, \ldots, n\} \mid \bar{c}_j^i > -\frac{\beta^i \|A\|}{u^i}\}$ into account for the calculations.

However, $\beta^i \|A\| / u^i$ is in our cases too large to be of practical relevance, i.e., so many columns are in the active set that no speed up of the algorithm was observable. Therefore we use the heuristic active sets described above.

4.5 Applications

In the literature the PBM has been applied to solve semidefinite relaxations of combinatorial optimization problems (Helmberg [2000]; Fischer et al. [2003]),

linear matrix inequalities (Scott [2001]), and Lagrangian relaxations of combinatorial problems (Lemaréchal [2001]).

We present now our adaption to the PBM to approximate the LP-relaxations of the duty scheduling problem (DSP) and the integrated vehicle and duty scheduling problem (ISP).

4.5.1 Approximating the Duty Scheduling Problem

In this Section we will collect some facts about approximating the LP-relaxation of problem (DSP) of Section 1.10.3 with the PBM. We have replaced the cost vector c of problem (DSP) of Section 1.10.3 by c^i because we will later use this model as a subproblem of a PBM to approximate the LP-relaxation of the ISP. In this version of the PBM the new cost vector c^i depends on the iteration i or more exactly on a current trial point λ^i. That is, we set

$$c^i := c + \lambda^{i\mathsf{T}} B. \tag{4.33}$$

Problem (DSP^i) is now:

$$\begin{array}{lll}
(\mathsf{DSP}^i) & \min c^{i\mathsf{T}} x + \gamma^{\mathsf{T}} z, & \\
& \text{s. t.} & \\
\text{(i)} & Ax & = \mathbb{1}, \\
\text{(ii)} & Rx - & z \le r, \\
\text{(iii)} & x & \in \{0,1\}^D, \\
\text{(iv)} & & z \ge 0.
\end{array}$$

Here $c^i \in \mathbb{R}^D$ are the costs of duties, $\gamma \in \mathbb{R}^{\mathcal{B}}$ are costs for the deviation of a resource consumption from its target, $r \in \mathbb{R}^{\mathcal{B}}$ are the targets of the resource consumptions, $A \in \{0,1\}^{V_{\mathsf{DSP}} \times D}$ is the task-duty-incidence matrix, and $R \in \mathbb{R}^{\mathcal{B} \times D}$ gives the resource consumptions of the duties. The variables $x \in \mathbb{R}^D$ are one if the corresponding duty is used and zero otherwise, the slack variables $z \in \mathbb{R}^{\mathcal{B}}$ give the deviations of the resource consumptions of a duty schedule from its targets.

We now look at a Lagrangean function of (DSP^i) which arises by relaxing constraints (i) and (ii):

$$f^i_{\mathsf{DSP}}(\mu,\nu) := \min_{x \in \{0,1\}^D} [c^{i\mathsf{T}} x + \mu^{\mathsf{T}}(\mathbb{1} - Ax) + \nu^{\mathsf{T}}(r - Rx)]. \tag{4.34}$$

The slack variables z are transformed into bounds on the Lagrangean multipliers corresponding to equations (ii). Thus, the domain of the function f^i_{DSP} is bounded to the set

$$\mathrm{dom}(f^i_{\mathsf{DSP}}) = \{(\mu,\nu) \in \mathbb{R}^{V_{\mathsf{DSP}}} \times \mathbb{R}^{\mathcal{B}} \mid \nu_r \in [-\gamma_r, 0], \forall r \in \mathcal{B}\}.$$

The optimum of the Lagrangean dual of the LP-relaxation of (DSP^i)

$$\max_{(\mu,\nu)\in\mathrm{dom}(f_{\mathsf{DSP}})} f^i_{\mathsf{DSP}}(\mu,\nu) \tag{4.35}$$

has the same value as the minimum of the LP-relaxation of (DSP^i). And every linearization of $f^i_{\mathsf{DSP}}(\cdot,\cdot)$ at a point $(\mu,\nu) \in \mathrm{dom}(f_{\mathsf{DSP}})$ can be written as

$$\bar{f}^i_{\mathsf{DSP}}(\mu,\nu;\lambda) := c^{i\mathsf{T}}x^i + \mu^{\mathsf{T}}(\mathbb{1} - Ax^i) + \nu^{\mathsf{T}}(r - Rx^i)$$

where x^i is a minimizer of $\min_{x\in[0,1]^D}(c^{i\mathsf{T}} - \mu^{\mathsf{T}}A - \nu^{\mathsf{T}}R)x$.

We now maximize f^i_{DSP} with the modified PBM for bounded functions (see Section 4.3.3). This results in an approximate primal solution $\tilde{x}^i \in [0,1]^D$ given by equation (4.15) and a solution $(\mu^i,\nu^i) \in \mathrm{dom}(f^i_{\mathsf{DSP}})$ that is hopefully in the vicinity of a maximizer of f^i_{DSP}. We define the residuum $h(x)$ of $x \in [0,1]^D$ by

$$h(x) := \begin{pmatrix} \mathbb{1} - Ax \\ \min\{r - Rx, 0\} \end{pmatrix}. \tag{4.36}$$

Then $h(x^i)$ is by construction (see Section 4.1.3) a subgradient at (μ^i,ν^i) of the cutting plane model of f^i_{DSP} used in the last iteration of the PBM, denoted by $\hat{f}^i_{\mathsf{DSP}}$, and therefore also an ϵ-subgradient of f^i_{DSP}. By Corollary (4.3) the value of $\hat{f}^i_{\mathsf{DSP}}$ at the new trial point can be expressed as a convex combination of linearizations in the current bundle:

$$\hat{f}^i_{\mathsf{DSP}}(\mu^i,\nu^i) = c^{i\mathsf{T}}\tilde{x}^i + h(\tilde{x}^i)^{\mathsf{T}} \begin{pmatrix} \mu^i \\ \nu^i \end{pmatrix}. \tag{4.37}$$

Therefore the following equation holds for all $(\mu,\nu) \in \mathrm{dom}\, f_{\mathsf{DSP}}$:

$$c^{i\mathsf{T}}\tilde{x}^i + h(\tilde{x}^i)^{\mathsf{T}} \begin{pmatrix} \mu \\ \nu \end{pmatrix} \geq \hat{f}^i_{\mathsf{DSP}}(\mu,\nu) \geq f^i_{\mathsf{DSP}}(\mu,\nu). \tag{4.38}$$

In the following (DSP^i) occurs as a subproblem of the Lagrangean relaxation of (ISP). We will need equation (4.38) to proof the convergence of the inexact PBM for (DSP) as well as (ISP).

4.5.2 Approximating the Integrated Duty and Vehicle Scheduling Problem

We now state the Lagrangean dual of the LP-relaxation of the problem (ISP) which will be solved by our adaption of the PBM. We recapitulate (ISP) here briefly:

$$
\begin{aligned}
&(\mathsf{ISP}) \quad \min\ c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z + d^{\mathsf{T}}y,\\
&\qquad \text{s.t.}\\
&\text{(i)} \qquad Ax \qquad\qquad\ \ = \mathbb{1},\\
&\text{(ii)} \qquad Rx - \quad z \qquad\ \ \leq r,\\
&\text{(iii)} \qquad\qquad\quad Ny = b,\\
&\text{(iv)} \qquad Bx \quad - \quad My = 0,\\
&\text{(v)} \qquad x \in \{0,1\}^D, \quad z \geq 0, \quad y \in \{0,1\}^{V^D_{\mathsf{DSP}}}.
\end{aligned}
$$

Constraints (i) ensure the covering of necessary tasks by duties, constraints (ii) provide the resource constraints of the duty scheduling problem, constraints (iii) model the vehicle scheduling problem, and constraints (iv) couple the duty and the vehicle scheduling problem. More details can be found in Section 2.2.3.

Let now be

$$P_{\mathsf{DSP}} := \{(x,z) \in [0,1]^D \times \mathbb{R}^{\mathcal{B}} \mid Ax = \mathbb{1}, Rx - z \leq r, z \geq 0\} \tag{4.39}$$

the polyhedron associated with the LP-relaxation of the DSP and

$$P_{\mathsf{VSP}} := \{y \in [0,1]^{V^D_{\mathsf{DSP}}} \mid Ny = b\} \tag{4.40}$$

the polyhedron associated with the LP-relaxation of the VSP. Then a Lagrangean relaxation with respect to the coupling constraints (ISP) (iv) and a relaxation of the integrality constraints (v) results in the Lagrangean dual

$$(\mathrm{L}) \quad \max_{\lambda} \left[\min_{y \in P_{\mathsf{VSP}}} (d^{\mathsf{T}} - \lambda^{\mathsf{T}}M)y + \min_{(x,z) \in P_{\mathsf{DSP}}} \left((c^{\mathsf{T}} + \lambda^{\mathsf{T}}B)x + \gamma^{\mathsf{T}}z \right) \right].$$

We now define the following functions and associated arguments by

$$
\begin{aligned}
f_V : \mathbb{R}^{V^D_{\mathsf{DSP}}} &\to \mathbb{R},\\
\lambda &\mapsto \min (d^{\mathsf{T}} - \lambda^{\mathsf{T}}M)y;\ y \in P_{\mathsf{VSP}},\\
f_D : \mathbb{R}^{V^D_{\mathsf{DSP}}} &\to \mathbb{R},\\
\lambda &\mapsto \min (c^{\mathsf{T}} + \lambda^{\mathsf{T}}B)x + \gamma^{\mathsf{T}}z;\ (x,z) \in P_{\mathsf{DSP}},\\
f\ &:= f_V\ +\ f_D,
\end{aligned}
\tag{4.41}
$$

and

$$y(\lambda) \in \operatorname*{arg\,min}_{y \in P_{\mathsf{VSP}}} f_V(\lambda),$$
$$(x(\lambda), s(\lambda)) \in \operatorname*{arg\,min}_{(x,s) \in P_{\mathsf{DSP}}} f_D(\lambda).$$

We take arbitrary minimizers of the problems. With this notation, (L) becomes

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

The functions f_V and f_D are concave and piecewise linear. Their sum f is therefore a separable, concave, and piecewise linear function; f is, in particular, nonsmooth. This is precisely the setting for the proximal bundle method. The main obstacle to use the PBM straightforward is that f_V and f_D are computational expensive to evaluate. Therefore we will only evaluate them approximately. If we use the inexact subgradients and approximated function values straightforward in the PBM it does not converge anymore. We will describe in the next section how we handle this inexactness such that the convergence of the inexact PBM to a near optimal minimizer of f is guaranteed.

4.6 Inexact Bundle Method

In this Section we want to describe how the PBM can be used if f is evaluated approximately and also ϵ-subgradients are used to calculate linearizations. Approximating f is useful if function evaluation requires the solution of an $\mathcal{NP}$-hard or at least numerical difficult problem. Inexact evaluation may also accelerate the PBM, especially if exactness of the solution is not so important. We will in this Section also review the literature about inexact bundle methods.

For the integrated vehicle and duty scheduling problem the situation is as follows: Two obstacles prevent the straightforward application of the proximal bundle method to the function f defined in equation (4.41). First, the component problem for duty scheduling is $\mathcal{NP}$-hard, even in its LP-relaxation; second, the vehicle scheduling LP is computationally at least not easy. We can therefore not expect that we can compute the function values $f_V(\lambda)$ and $f_D(\lambda)$ and the associated subgradients $g_V(\lambda)$ and $g_D(\lambda)$ exactly. We use the algorithms in Löbel [1997b] to approximate $f_V(\lambda)$ and $g_V(\lambda)$, see also Section 1.9; and we use a column generation approach, sketched in Section 1.10, to approximate $f_D(\lambda)$ and $g_D(\lambda)$.

We now review the literature for approximate subgradient calculation in bundle methods and then explain our approaches to handle the inexactness in the evaluation of the component functions f_V and f_D.

4.6.1 Literature

The literature gives different versions of approximate bundle methods that can deal with inexact evaluations of the component functions. In Kiwiel [1995] a version of the PBM is stated that asymptotically produces a solution, given that ϵ-linearizations of the function f to be minimized can be found at every trial point $\mu \in \mathbb{R}^m$ for all $\epsilon > 0$, i.e., one can find an affine function $\bar{f}_\epsilon(\lambda;\mu) := f_\epsilon(\mu)+g_\epsilon(\mu)^\mathsf{T}(\lambda-\mu)$ such that $f_\epsilon(\mu) \geq f(\mu)-\epsilon$ and $f(\lambda) \geq \bar{f}_\epsilon(\lambda;\mu)$ for all $\lambda \in \mathbb{R}^m$.

In Hintermüller [2001] another version is given, which replaces exact subgradients of f by ϵ-subgradients. In this method it is not necessary to know or control the value of ϵ; it produces solutions that are as good as the supplied ϵ-subgradients. They converge, in particular, to the optimum if the linear approximation also converges to the original function during the algorithm.

In principle, we could use these approaches 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. The information that can be derived from the subgradients associated with this single-depot relaxation was unfortunately 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 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 Section 5.8. 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.

In a recent approach, see Kiwiel [2006], a variant of the proximal bundle method is given that can handle approximate evaluation of f and its sub-

gradients. The setting is: Let S be a nonempty closed convex set in $\mathbb{R}^n$. For fixed, but probably unknown, accuracy tolerances $\epsilon_f \geq 0$ and $\epsilon_g \geq 0$ we can find for every $\lambda \in S$ an approximate value f_λ and an approximate subgradient g_λ of f that produces a approximate linearization of f:

$$\bar{f}_\lambda(\mu) := f_\lambda + g_\lambda{}^\mathsf{T}(\mu - \lambda) \leq f(\mu) + \epsilon_g \text{ with } \bar{f}_\lambda(\lambda) = f_\lambda \geq f(\lambda) - \epsilon_f.$$

There also the convergence to an $(\epsilon_f + \epsilon_g)$-optimal solution of the minimization problem is proven. The variant only needs an easy to implement modification of the original PBM. However, we are not able to guarantee in our setting that $f_\lambda \geq f(\lambda) - \epsilon_f$ because the inexact column generation for the (DSP) does not generate lower bounds, see Section 4.6.3 for details.

4.6.2 Vehicle Scheduling Component Function

We now examine the approximate evaluation of the vehicle scheduling component function f_V at a trial point λ^i. This involves solving VSP^i, a VSP with modified objective function with respect to the trial point at iteration i.

$$\begin{aligned}
&(\mathsf{VSP}^i) && \min\ (d^\mathsf{T} - \lambda^{i\mathsf{T}} M)y, \\
& && \text{s. t.} \\
&\text{(i)} && y \in P_{\mathsf{VSP}}, \\
&\text{(ii)} && y \in \{0,1\}^{A_{\mathsf{VSP}}}.
\end{aligned}$$

The vehicle scheduling polyhedron P_{VSP} is defined by equation (4.40). Let y^i be a feasible, preferable near-optimal, solution of (VSP^i). Then we define an affine function $\bar{f}_V^i$ that approximates f_V in the vicinity of λ^i:

$$\begin{aligned}
\bar{f}_V^i &: \mathbb{R}^{V_{\mathsf{DSP}}^D} \mapsto \mathbb{R}, \\
\bar{f}_V^i(\lambda) &:= (d^\mathsf{T} - \lambda^\mathsf{T} M)y^i.
\end{aligned}$$

The gradient g_V^i of $\bar{f}_V^i$ can be computed by the solution y^i as follows:

$$g_V^i := -My^i.$$

It holds that $\bar{f}_V^i(\lambda) \geq f_V(\lambda)$ for all λ because y^i is feasible. We can use the functions $\bar{f}_V^j$, $j = 1, \ldots, i$ to create a concave approximation $\hat{f}_V^i \geq f_V$ of f_V by setting

$$\hat{f}_V^i(\lambda) \ := \min_{j=1,\ldots,i} \bar{f}_V^j(\lambda).$$

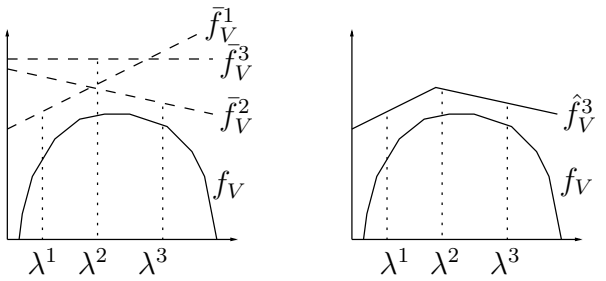


Figure 4.2: The functions f_V, $\bar{f}_V^i$, $i = 1, \ldots, 3$ and $\hat{f}_V^3$.

We use this approximation in the PBM (Algorithm 3) by replacing f_V by $\hat{f}_V^i$. The relations of f_V, $\bar{f}_V^i$, and $\hat{f}_V^i$ are depicted exemplarily in Figure 3.

Since the function $\hat{f}_V^i$ depends on all linearizations calculated until the current iteration, we must also recalculate its value $\hat{f}_V^i(\bar{\lambda}^i)$ at the stability center in the stopping criterion and the ascent test (Steps 4 and 6) of the PBM at every iteration. Since we have a finite number of valid vehicle schedules for a given VSP we have also only a finite number of linearizations $\bar{f}_V^i(\lambda)$. So, after some iteration i^*, all linearizations are known and the function $\hat{f}_V^i$ is identical to all functions $\hat{f}_V^j, j > i$. Hence after that iteration i^* we are in the setting of the standard PBM that provably converges.

Potentially, storing all linearizations $\bar{f}_V^i$ is very memory consuming. However, for the convergence of the algorithm it suffices to store the linearizations that cut off a current stability center otherwise they can be recomputed if necessary. We further assume that it is also sufficient to store at each iteration i only the linearizations used in the current bundle J_V^i to prove convergence of the PBM. However, the proof of this assumption is still open. Nevertheless, our computations seem to back this assumption.

4.6.3 Duty Scheduling Component Function

In this section we describe how we approximate the subgradients and function values of the duty scheduling component function f_D of the ISP at the trial points of the PBM again with the PBM. Then we examine the influence of inexact column generation on the calculated subgradients and function values. At last we show the convergence of our modified PBM.

Approximating the Subgradient

We approximate the duty scheduling component function f_D at the trial points $\lambda^i \in \mathbb{R}^{V_{\mathsf{DSP}}^D}$, $i = 1, 2, \ldots$ of the PBM by approximating the corresponding LP again by the PBM. This gives us, as described in Section 4.5.1, an approximation $(\tilde{x}^i, \tilde{z}^i)$, $\tilde{z}^i := \max\{0, r - R\tilde{x}^i\}$ of the arguments of f_D, and a lower bound of $f_D(\lambda^i)$ by $f^i_{\mathsf{DSP}}(\mu^i, \nu^i)$, as defined in (4.34), where (μ^i, ν^i) is the last stability center of the PBM used to approximate $f_D(\lambda^i)$. We now want to motivate that the calculated $\tilde{x}^i$ helps us to approximate (ISP) using the PBM.

We define

$$L(\lambda, \mu, \nu) := \min_{x \in \{0,1\}^D} \left[(c + \lambda^\mathsf{T} B)x + \mu^\mathsf{T}(\mathbb{1} - Ax) + \nu^\mathsf{T}(r - Rx)\right] \tag{4.42}$$

for all $\lambda \in \mathbb{R}^{V_{\mathsf{DSP}}^D}$ and all $(\mu, \nu) \in \operatorname{dom} f_{\mathsf{DSP}}$. Then is, by definition, $L(\lambda^i, \mu, \nu) = f^i_{\mathsf{DSP}}(\mu, \nu)$, for all trial points λ^i and all $(\mu, \nu) \in \operatorname{dom} f_{\mathsf{DSP}}$.

Proposition 4.4. Let $\tilde{x}^i$ be the approximate solution of (DSP^i) calculated by the PBM and let $(\lambda^i, \mu^i, \nu^i)$ be the last trial point. Then

$$g(\tilde{x}^i) := \begin{pmatrix} B\tilde{x}^i \\ \mathbb{1} - A\tilde{x}^i \\ \min\{r - R\tilde{x}^i, 0\} \end{pmatrix}$$

is an ϵ_D-subgradient of L at $(\lambda^i, \mu^i, \nu^i)$ for some $\epsilon_D \geq 0$.

Proof. Let

$$\bar{L}^i(\lambda, \mu, \nu) := L(\lambda^i, \mu^i, \nu^i) + g(\tilde{x}^i)^\mathsf{T} \begin{pmatrix} \lambda - \lambda^i \\ \mu - \mu^i \\ \nu - \nu^i \end{pmatrix}$$

be the affine function for which we want to show that for a fixed $\epsilon_D \geq 0$ holds: $\bar{L}^i + \epsilon_D$ is above L. Let x^i be an optimal solution of (DSP^i). Then we have

$$\bar{L}^i(\lambda, \mu, \nu) = f^i_{\mathsf{DSP}}(\mu^i, \nu^i) + g(\tilde{x}^i)^\mathsf{T} \begin{pmatrix} \lambda - \lambda^i \\ \mu - \mu^i \\ \nu - \nu^i \end{pmatrix}, \tag{4.43}$$

now we use (4.38) setting $\epsilon_D := c^{i\mathsf{T}}\tilde{x}^i + h(\tilde{x}^i)\begin{pmatrix}\mu^i\\ \nu^i\end{pmatrix} - f^i_{\mathsf{DSP}}(\mu^i,\nu^i) > 0$:

$$= c^{i\mathsf{T}}\tilde{x}^i + h(\tilde{x}^i)^{\mathsf{T}}\begin{pmatrix}\mu^i\\ \nu^i\end{pmatrix} + g(\tilde{x}^i)^{\mathsf{T}}\begin{pmatrix}\lambda-\lambda^i\\ \mu-\mu^i\\ \nu-\nu^i\end{pmatrix} - \epsilon_D \tag{4.44}$$

$$\tag{4.45}$$

by the definitions (4.33) of c^i and (4.36) of h we get

$$= c^{\mathsf{T}}x^i + g(x^i)^{\mathsf{T}}\begin{pmatrix}\lambda\\ \mu\\ \nu\end{pmatrix} - \epsilon_D \tag{4.46}$$

$$\geq L(\lambda,\mu,\nu) - \epsilon_D. \tag{4.47}$$

Equation (4.47) stems from replacing x^i with the minimal argument of $L(\lambda,\mu,\nu)$. □

The size of ϵ_D depends on the quality of the primal solution $\tilde{x}^i$ and the dual solution (μ^i,ν^i) of the PBM. If both are optimal ϵ_D is zero. Now we use Proposition 4.4 to estimate the quality of the linearizations

$$\bar{f}^i_D(\lambda) := f^i_{\mathsf{DSP}}(\mu^i,\nu^i) + (B\tilde{x}^i)^{\mathsf{T}}(\lambda-\lambda^i)$$

of f_D at a given λ^i. To do that we define (DSP^i) analogously to (VSP^i) as follows:

(DSP^i) $\quad \min\ (c^{\mathsf{T}} + \lambda^{i\mathsf{T}}B)x + \gamma^{\mathsf{T}}z,$

s. t.

(i) $\quad (x,z) \in P_{\mathsf{DSP}},$

(ii) $\quad x \in \{0,1\}^D.$

Proposition 4.5. Let $\tilde{x}^i$ be an approximate solution of (DSP^i) calculated by the PBM. Let $\pi^i := \begin{pmatrix}\mu^i\\ \nu^i\end{pmatrix}$ be the last trial point of the PBM to approximate $f_D(\lambda^i)$, and let

$$(\pi^\lambda) := \begin{pmatrix}\mu^\lambda\\ \nu^\lambda\end{pmatrix}, \quad (\mu^\lambda,\nu^\lambda) \in \operatorname*{arg\,min}_{(\mu,\nu)\in\operatorname{dom} f_{\mathsf{DSP}}} L(\lambda,\mu,\nu)$$

a minimizer of $L(\lambda,\cdot,\cdot)$. Then it follows

$$\bar{f}^i_D(\lambda) \geq f_D(\lambda) + h(\tilde{x}^i)^{\mathsf{T}}(\pi^i - \pi^\lambda) - \epsilon_D.$$

Proof. We have

$$\bar{f}_D^i(\lambda) = L(\lambda^i, \mu^i, \nu^i) + (B\tilde{x}^i)^\mathsf{T}(\lambda - \lambda^i) \tag{4.48}$$
$$\geq L(\lambda, \mu^\lambda, \nu^\lambda) - h(\tilde{x}^i)^\mathsf{T}(\pi^\lambda - \pi^i) - \epsilon_D \tag{4.49}$$
$$= f_D(\lambda) - h(\tilde{x}^i)^\mathsf{T}(\pi^\lambda - \pi^i) - \epsilon_D. \tag{4.50}$$

Inequality (4.49) follows by Proposition 4.4. The remainder follows by using the definitions of the used functions. □

Proposition 4.5 shows how the quality of the linearization of f_D at λ^i depends on the quality of the approximate solution $\tilde{x}^i$. If we would find a solution x^* in P_{DSP} the residuum would be $h(x^*) = 0$. This would imply by Proposition 4.5 that Bx^* is an ϵ_D-subgradient of f_D at λ^i. In practice, we are only able to find an $\tilde{x}^i$ such that $h(\tilde{x}^i)$ is very close to the null-vector. If $\mathrm{dom}(f_{\mathsf{DSP}})$ is bounded, we can give an upper bound ϵ_h on the term $h(\tilde{x}^i)^\mathsf{T}(\pi^\lambda - \pi^i)$. Then it follows by Proposition 4.50 that $g(\tilde{x}^i)$ is an $(\epsilon_D + \epsilon_h)$-subgradient of f_D at λ^i. In our case $\mathrm{dom}(f_{\mathsf{DSP}})$ is in general not bounded, but in practice the dual variables (μ^i, ν^i) do not change dramatically when moving from λ^i to λ^{i+1}. Therefore we simply ignore the term $h(\tilde{x}^i)^\mathsf{T}(\pi^\lambda - \pi^i)$ since it is close to zero.

Alternatively to ignoring this term, we could replace the duty scheduling component function f_D by L. If now $x_{\lambda,\mu,\nu}$ is a minimizer of (4.34) then $g(x_{\lambda,\mu,\nu})$ defines a subgradient of L at λ. This requires to store larger subgradients and computational tests show that the overall performance does not improve.

Column Generation

Unfortunately, we do not know all relevant duties to approximate f_D at a given trial point λ and we are also not able to compute them in reasonable time, because the corresponding pricing problem is $\mathcal{NP}$-hard (see Chapter 5). Hence, we do not know the complete set of duties needed to approximate (ISP), but only a subset of it denoted by I. We therefore consider here a restriction of the duties used in problem (DSP^i) leading to a problem (DSP_I^i), where I is a subset of all duties D.

$$\begin{aligned}
(\mathsf{DSP}_I^i) \qquad & \min\ (c_I{}^\mathsf{T} + \lambda^{i\mathsf{T}} B_{\cdot I})x_I + \gamma^\mathsf{T} z, \\
& \text{s.t.} \\
\text{(i)} \qquad & (x_I, z) \in P_{\mathsf{DSP}}^I, \\
\text{(ii)} \qquad & x_I \in \{0,1\}^I
\end{aligned}$$

and P^I_{DSP} is the associated polyhedron:

$$P^I_{\mathsf{DSP}} := \{(x,z) \in [0,1]^D \times \mathbb{R}^{\mathcal{B}} \mid A_{.I}x_I = \mathbb{1}, R_{.I}x_I - z \leq r, z \geq 0\}. \quad (4.51)$$

Here $X_{.I}, I \subset N$ denotes the matrix that consists of the columns I of a matrix $X \in \mathbb{R}^{M \times N}$, where M and N are arbitrary index-sets. Analogously a vector $x_I \in \mathbb{R}^I$ consists of the elements $i \in I$ of $x \in \mathbb{R}^N$. And we treat a vector $x_I, I \subset N$ as a vector $x \in \mathbb{R}^N$ by setting $x_i = 0$ for all $i \in N \setminus U$.

The set of duties I will be adapted dynamically after each serious step of the PBM. We do not generate duties at every iteration of PBM because, even though it is a heuristic, the pricing algorithm presented in Chapter 5 is still the most time consuming part of our ISP-algorithm. The optimum of the LP-relaxation of (ISP_I) may be above that of (ISP) because I may not contain all duties with negative reduced cost. That causes also that an optimal solution x^*_I of (ISP_I) may not define a subgradient of f_D.

Now we formalize our approach. We introduce the function f^I_D:

$$\begin{aligned} f^I_D : \mathbb{R}^{V^D_{\mathsf{DSP}}} &\to \mathbb{R}, \\ \lambda &\mapsto \min_{(x_I,z) \in P^I_{\mathsf{DSP}}} (c_I{}^{\mathsf{T}} + \lambda^{\mathsf{T}} B_{.I}) x_I + \gamma^{\mathsf{T}} z \end{aligned}$$

Cutting Plane Model of f_D

Let be I^i now the set of duties used in iteration i of the PBM. We approximate the function f_D at the current trial point again by the PBM as described in Section 4.5.1. Only now we restrict the duties to I^i. That is, we calculate an approximated maximizer (μ^i, ν^i) of the Lagrangean function L^i_I of DSP^i restricted to the duties I^i

$$\begin{aligned} &L^i_I(\mu,\nu) := \\ &\qquad \min_{x_{I^i} \in [0,1]^{I^i}} \left[(c_{I^i} + (\lambda^i)^{\mathsf{T}} B_{.I^i}) x_{I^i} + \mu^{\mathsf{T}} (\mathbb{1} - A_{.I^i} x_{I^i}) + \nu^{\mathsf{T}} (r - R_{.I^i} x_{I^i}) \right]. \end{aligned}$$

Again, λ^i denotes the current trial point. The approximate function value $L^i_I(\mu^i, \nu^i)$ and the primal approximation $\tilde{x}^i$ of ($\mathsf{DSP}^i_{I^i}$) is then used to build a linear approximation of f_D in the vicinity of λ^i:

$$\begin{aligned} g^i_D &:= B_{.I^i} \tilde{x}^i, \\ \bar{f}^i_D(\lambda) &:= L^i_I(\mu^i, \nu^i) + \langle g^i_D, \lambda - \lambda^i \rangle. \end{aligned} \quad (4.52)$$

Two problems can occur at an iteration i if we use these linearizations to build a cutting plane model of f_D:

1. Since $L_I^i(\mu^i, \nu^i)$ is only a lower bound of DSP_I^i it may happen that some old linearization $\bar{f}_D^j$, $j < i$ cuts off the new trial point λ^i. That is $\bar{f}_D^i(\lambda^i) > \bar{f}_D^j(\lambda^i)$.

2. Let $\bar{\lambda}^i$ be the current stability center and $f_D^{*,i}$ the approximated value of f_D at the stability center. Then it is in general not true that $\bar{f}_D^i(\bar{\lambda}^i) \geq f_D^{*,i}$ because we only compute ϵ-subgradients. That is, it may happen that the new linearization cuts off the stability center.

These two cases may prevent convergence of the PBM, therefore we handle them when we update the bundle J_D^i of linearizations, which defines the new cutting plane model. In both cases we translate the new linearization such that it no more cuts off the relevant points or is cut off by an old linearization. In the first case, we translate the function $\bar{f}_D^i$ by $\bar{f}_D^i(\lambda^i) - \bar{f}_D^j(\lambda^i)$. In the second case we translate it by the observed difference $\bar{f}_D^i(\lambda^i) - f_D^{*,i}$. That is we add the function

$$\bar{f}_D^{*,i} := \bar{f}_D^i + \max\{0, \max_{j=1,\ldots,i} \left(\bar{f}_D^i(\lambda^i) - \bar{f}_D^j(\lambda^i)\right), \bar{f}_D^i(\lambda^i) - f_D^{*,i}\}$$

to the bundle J_D^i. The duty scheduling bundle update in Step 5 of Algorithm 3 is then for the inexact bundle method implemented as

$$J_D^{i+1} = J \cup \{\tilde{f}_D^i\} \cup \{\bar{f}_D^{*,i+1}\} \tag{4.53}$$

and $J \subset J_D^i$. The aggregated linearization $\tilde{f}_D^i$ is defined analogously to (4.10). The bundle has to include the last linearization and the aggregated linearization $\tilde{f}_D^i$ to converge. The cutting plane model at iteration i finally is:

$$\hat{f}_D^i(\lambda) := \min_{\bar{f} \in J_D^i} \bar{f}(\lambda). \tag{4.54}$$

Convergence

We now want to show under which circumstances the inexact PBM for the combined vehicle and duty scheduling function converges.

We assume that $I^1 \subset I^2 \subset I^3 \subset \ldots$. That is, we never remove a duty that we have generated. In practice we have to remove duties sometimes because of memory shortages but we only remove duties at an iteration i that are unlikely to be non-zero in solutions of the duty scheduling problems (DSP^j), with $i < j < i_{\text{serious}}$, where i_{serious} is the iteration of the next serious step.

We only generate new columns of the underlying duty scheduling problem at serious steps. Therefore we know that between two subsequent serious steps at iterations i and i_{serious} the set of duties is constant, i.e., $I^i = I^{i+1} = \cdots = I^{i_{\text{serious}}-1}$. This implies that the value f_D^k, $i \leq k < i_{\text{serious}}$ is a lower bound of $f_D^I(\lambda^k)$, for $I := I^i$.

Corollary 4.6. Let $h(\tilde{x}^i)^\mathsf{T}(\pi^i - \pi^\lambda) \geq -\epsilon_h$, for a fixed $\epsilon_h \geq 0$, for all i, and all λ. Then there exists an $\epsilon \geq 0$ with

$$\bar{f}_D^i(\lambda) \geq f_D^I(\lambda) - \epsilon.$$

for all $\lambda \in \mathbb{R}^{V_{\mathsf{DSP}}^D}$.

Proof. This follow directly by Proposition 4.5 if we set $D = I^i$ and $\epsilon = \epsilon_D + \epsilon_h$. □

By this corollary it follows that the current stability center, which was set in iteration i, is cut off by new linearizations at an iteration $k \in [i, j]$ by at most ϵ, that is $\max_{k=i,\ldots,j}(\bar{f}_D^k(\bar{\lambda}^i) - f_D^{*,i}) \leq \epsilon$. Therefore we have to translate the new linearizations that we add to the bundle also at most by ϵ. Since $\bar{f}_D^k(\lambda^k)$ is a lower bound of $f_D^{I^k}(\lambda^k)$ for all k we also know that the value of the next stability center is at most $f_D^{I^j}(\lambda^j) + \epsilon$. That means that the value of the new stability center overestimates the real function value by at most ϵ.

The number of duties is finite. Hence, at some iteration i, no more new duties are added. That is $I^i = I^{i+1} = I^{i+2} = \ldots$. Let now ϵ_f be the largest difference between $f_D(\lambda^i)$ and $f_D^{I^i}(\lambda^i)$, i.e., the largest deviation arising by the inexact column generation of the duty scheduling problem. Then holds $f_D(\lambda^i) + \epsilon_f \geq f_D^{I^i}(\lambda^i)$.

Together with Corollary 4.6 we deduce that

$$f_D(\lambda^i) + \epsilon_f \geq \bar{f}_D^i(\lambda^i) \geq f_D^I(\lambda^i) - \epsilon. \tag{4.55}$$

That is, the value of every trial point overestimates the real function value at most by ϵ_f and underestimates it by at most ϵ. Therefore all linearizations that were added throughout the algorithm together with their translations build a non-smooth concave function that is an approximation of the function to optimize. For this function the convergence results of Kiwiel [1990] hold.

4.6.4 Combined Functions

The combined approximated function to optimize and its cutting plane model are finally

$$\begin{aligned} f &:= f_V + f_D \\ \hat{f}^i &:= \hat{f}^i_V + \hat{f}^i_D. \end{aligned}$$

We further define symbols for the aggregated subgradients of the component functions and the combined function:

$$\begin{aligned} &\tilde{g}^i_V := \textstyle\sum_{\bar{f} \in J^i_V} \alpha^i_{\bar{f}} \nabla \bar{f}, \\ &\tilde{g}^i_D := \textstyle\sum_{\bar{f} \in J^i_D} \alpha^i_{\bar{f}} \nabla \bar{f}, \mathit{and} \\ &\tilde{g}^i := g^i_V + g^i_D. \end{aligned} \tag{4.56}$$

The convex multipliers α^i are the solution of the subproblem at iteration i according to (4.11). The norm of g^i gives a measure for the quality of the primal approximation

$$(\tilde{\tilde{x}}^i, \tilde{\tilde{z}}^i, \tilde{y}^i) \in [0,1]^{I^i} \times \mathbb{R}^{\mathcal{B}}_+ \times [0,1]^{A_{\mathsf{VSP}}},$$

of (ISP) where

$$\tilde{\tilde{x}}^i := \sum_{\bar{f} \in J^i_D} \alpha^D_{\bar{f}} x_{\bar{f}}, \quad \tilde{\tilde{z}}^i := \max\{0, r - R\tilde{\tilde{x}}^i\}, \quad \tilde{y}^i := \sum_{\bar{f} \in J^i_V} \alpha^V_{\bar{f}} y_{\bar{f}},$$

Here $\bar{f}$ are the linearizations in the current bundles J^i_D and J^i_V, respectively, (α^D, α^V) is the solution of the quadratic subproblem, cf. (4.11), at iteration i, and $x_{\bar{f}}$ and $y_{\bar{f}}$ are the approximated primal solutions and aggregations corresponding to $\bar{f}$.

At last, we need the sum of the objective values of the new linearizations at the new trial point at each iteration i: $f^i := \hat{f}^i_V(\lambda^i) + \hat{f}^i_D(\lambda^i)$. The objective value at the stability center at iteration i is denoted by $\bar{f}^i$.

The PBM to maximize f is now stated as Algorithm 6.

The inexact PBM in Algorithm 6 is stated for the combined function of the Lagrangean relaxation of model (ISP). However, it is possible to use it for other (non-separable) concave functions f. The inexact PBM has the following differences to the PBM (Algorithm 3): The method does not compute exact values of $f(\lambda^i)$ at Steps 1 and 3 but approximates these values. They are denoted by $\bar{f}^i_V(\lambda^i)$ and $\bar{f}^i_D(\lambda^i)$. This notation indicates also that these values are used to generate the approximate linearizations of f. In Steps 1

Algorithm 6 Inexact PBM with Column Generation.

Input: Starting point $\lambda^0 \in \mathbb{R}^{V_{\mathsf{DSP}}^D}$, duty set I^0, weights $u^0, m > 0$, optimality tolerance $\epsilon \geq 0$.

1: Initialization: $i \leftarrow 0$, $J_V^i \leftarrow \{\bar{f}_V^i(\lambda^i)\}$, $J_D^i \leftarrow \{\bar{f}_D^i(\lambda^i)\}$, and $\bar{\lambda}^i = \lambda^i$. Set $\bar{f}^i \leftarrow \bar{f}_V^0(\lambda^0) + \bar{f}_D^0(\lambda^0)$.
2: Direction Finding: Compute λ^{i+1}, $\tilde{g}_V^i$, $\tilde{g}_D^i$ by solving problem (QP) (equation (4.11)).
3: Function evaluation: Compute $\bar{f}_V^{i+1}(\lambda^{i+1})$, g_V^{i+1}, $\bar{f}_D^{i+1}(\lambda^{i+1})$, g_D^{i+1}, and I^{i+1}.
4: Stopping Criterion: If $\hat{f}^i(\lambda^{i+1}) - \bar{f}^i < \epsilon(1 + |\bar{f}^i|$ output $\bar{\lambda}^i$, terminate.
5: Linearization Updating: Select J_V^{i+1} like in the standard PBM and J_D^{i+1} as stated in (4.53).
6: Ascent Test: **if** $\hat{f}^{i+1}(\lambda^{i+1}) - \bar{f}^i > m_L(\hat{f}^i(\lambda^{i+1}) - \bar{f}^i)$ **then** set $\bar{\lambda}^{i+1} \leftarrow \lambda^{i+1}$ and $\bar{f}^{i+1} \leftarrow f^{i+1}(\lambda^{i+1})$ **else** set $\bar{\lambda}^{i+1} \leftarrow \bar{\lambda}^i$ and $\bar{f}^{i+1} \leftarrow \bar{f}^i$.
7: Weight Update: Compute u^{i+1}.
8: $i \leftarrow i + 1$, goto step 2.

and 3 we also compute ϵ-subgradients of f_D and f_V denoted by g_V^i and g_D^i. The approximate linearizations are defined by:

$$\bar{f}_{V/D}^i(\lambda) := \bar{f}_{V/D}^i(\lambda^i) + \langle g_{V/D}^i, \lambda - \lambda^i \rangle.$$

In the stopping criterion of Step 4 and the ascent test of Step 6 the function value at the stability center is replaced by the smallest function value of all known approximate linearizations of f at the stability center denoted by $\bar{f}^i$. For the linearization updating (Step 5) we have to eventually translate linearizations in the bundle to guarantee convergence.

Theorem 4.7 (Convergence of the inexact PBM)**.** The inexact PBM (Algorithm 6) produces a series of stability centers $(\bar{\lambda}^i)_{i=1,2,\ldots}$ that converges to a point $\bar{\lambda}^*$ such that $|f(\bar{\lambda}^*) - f(\lambda^*)| \leq \epsilon$ for an optimal solution λ^* of $\max_{\lambda \in \mathbb{R}^{A_{\mathsf{VSP}}}} f(\lambda)$ and an $\epsilon > 0$ dependend on the quality of the approximations of f_D and f_V.

Proof. Algorithm 6 is constructed in such a way that the linearizations $\bar{f}_V^i(\lambda^i)$ and $\bar{f}_D^{*,i}$ that are added to the bundles are forming a concave function. More precisely

$$f'(\lambda) := \min_{i=1,2,\ldots} \bar{f}_V^i(\lambda) + \min_{i=1,2,\ldots} \bar{f}_D^{*,i}(\lambda)$$

is a concave function in λ. Algorithm 6 can be interpreted as an exact PBM that optimizes over f'. And therefore the series of stability centers converges to a maximizer $\bar{\lambda}^*$ of f'.

Setting ϵ_1 appropriately it follows by equation (4.55) that $|\bar{f}_D^i(\lambda^i) - f(\lambda^i)| \leq \epsilon_1$. Further we set $\epsilon_2 = \max_{i=1,2,\ldots,} f_V^i(\lambda^i) - f_V(\lambda^i)$, (cf. Section 4.6.2). Thus, at all trial points λ^i the following holds: $|f(\lambda^i) - f'(\lambda^i)| \leq \epsilon_1 + \epsilon_2$. If we now set $\epsilon = \epsilon_1 + \epsilon_2$ the claim follows. □

4.7 Computational Results

In this section, we compare our PBM code with and without the active set method (see Section 4.4) on some set partitioning problems with the Volume algorithm (see Barahona & Anbil [1998]), a dual ascent algorithm, sometimes also called coordinate ascent, (see Wedelin [1995]) enhanced with active sets, a classical subgradient approach (see Byun [2001]), and the dual simplex and the barrier algorithm of CPLEX 10.0.

To test the Volume Algorithm we used the version from the COIN-Homepage[1]. This implementation of the Volume Algorithm is part of a library of frameworks and solvers for problems in Operations Research, see Lougee-Heimer [2003]. We removed all stopping criteria and set the maximum number of iterations to 10,000. The PBM was implemented as described in Kiwiel [1990] with the solver for the quadratic direction finding problem described in Kiwiel [1994]. We also used the same parameters, which are $m_l = 0.1$, $m_r = 0.5$. The bundle size is limited to 5. The stopping criterion was removed and the maximum number of iterations was also set to 10,000. We use a simplified weight updating algorithm: If we have more than 200 subsequent null steps we multiply the current weight with 3/4. If between two serious steps are less than 3 null steps we multiply the weight with 4/3. If the weight is less than 0.1 the weight is set to 0.1. This worked in our tests better then the weight updating scheme described in Kiwiel [1990]. Additionally, we present also the results of the PBM with our Active-Set-Method of Section 4.4.

The computational results of the inexact bundle method for the ISP can be found in Chapter 7.

4.7.1 Testbed

Our testbed consists of set partitioning problems of various size created by column generation algorithms. We selected only problems without base con-

[1] see `http://www.coin-or.org`

problem	columns	rows	non-zeros	nz/col
ivu01	2,549	81	13,404	5.26
ivu02	25,412	185	191,819	7.55
ivu04	37,764	346	291,222	7.71
ivu05	159,438	847	1,450,704	9.10
ivu06	980,578	1,177	10,565,680	10.77
ivu41	169,524	3,570	1,031,701	6.09
ivu41b	838,500	3,570	8,796,292	10.49
ivu59	2,569,996	3,436	36,186,332	14.08
aa01	8,904	823	72,965	8.19
aa04	7,195	426	52,121	7.24
us01	1,053,137	145	13,636,541	12.95

Table 4.1: Characteristics of the Testbed

straints, because the methods of Wedelin [1995] and Byun [2001] are not able to cope with them.

The problems beginning with "ivu" are stemming from duty scheduling problems and were generated by our DSP-solver DS-OPT. They consist of all columns that were generated throughout the solution process. The problems ivu41 and ivu41b represent different settings of DS-OPT for the same scenario. For ivu41 the column generation process was handled more restrictive than for ivu41b, therefore, ivu41b has more columns than ivu41. We selected these two instances to exemplarily show the influence of the number of columns to the running time of the various algorithms.

We also show results on set-partitioning instances from the OR-Library, see Beasley [1990]. We selected three instances considered as hard in van Krieken et al. [2004], namely us01, aa01, and aa04. The characteristics of these instances are shown in Table 4.1.

4.7.2 Results

Our main goal is to calculate lower bounds of set partitioning problems in short time. For this reason, we think that it is not useful to compare only the results at the termination of the algorithms but the development of the lower bounds throughout the running time. A second goal is to calculate primal information useful for branching decisions in a branch-and-bound framework.

This is only accomplished by the PBM, the Volume method, the dual simplex, and the barrier method. The dual simplex and the barrier method are exact approaches, that is, they calculate feasible primal solutions. The PBM and the Volume algorithm only approximate primal solutions such that the set partitioning constraints are only nearly fulfilled. Therefore we show in Table 4.2 for the PBM and the Volume algorithm the maximum violation of constraints $Ax = \mathbb{1}$; x is here the current primal solution and $A \in \{0,1\}^{m \times n}$ is the coefficient matrix of the SPP. We show the value of $\max_{i=1,\ldots,m} |1 - A_i x|$

For the two scenarios ivu41 and ivu41b, we plot the graphs of the lower bounds over the time for all tested algorithms in Figure 4.3. The upper figure shows the graphs for ivu41 the lower one the graphs for ivu41b. Here one can see that the dual ascent has the fastest start but has problems to close the gap to the optimal value. The subgradient method is also fast at the beginning and then suffers from low convergence speed. The lower figure shows the characteristic sharp bend in the curve as the step length of the subgradient method is adapted. Our variant of the PBM is nearly as fast as the subgradient method. Additionally, it is better able to close the gap to the optimum. The PBM with active sets is faster than the PBM without them, and the difference becomes more significant when the number of columns gets larger.

The Volume algorithm is slower than the other subgradient algorithms. This may be caused by the implementation, which seems to be not optimized for speed but for readability. The other disadvantage of the Volume algorithm in comparison to the PBM is the lack of convergence to the optimal value of the SPP.

The barrier method is slower than the other methods in the beginning due to the expensive factorization of the constraint matrix and converges then very fast to the optimum. However, a major drawback of the barrier method is the large memory consumption, the barrier method ran out of memory on our computer while solving problem ivu59. At last, the dual simplex needs noticeable more time than the other algorithms.

The running time of most of the algorithms seems to be roughly proportional to the number of columns in our scenario. Only the running time of the dual simplex increases disproportionately to the number of columns.

In Table 4.2 the results of all scenarios in the test bed are shown. The times are wall clock time in seconds. The computations where done on a Intel(R) Pentium(R) 4 CPU 3.80GHz PC with 2GB RAM. The operating system was Linux 2.16. Our algorithms were all compiled with gcc 4.1.0 with full optimization and loop unrolling.

prob	ϵ	dual	bar	pbm		pbm_as		Volume		sub	ca
		time	time	time	$\|\cdot\|_1$	time	$\|\cdot\|_1$	time	$\|\cdot\|_1$	time	time
ivu01	0.05	0.03	0.02	0.02	0.21	0.01	0.48	0.02	1.59	0.02	0.00
	0.01	0.03	0.03	0.05	0.20	0.03	0.16	0.05	0.19	0.03	0.03
	0.001	0.04	0.03	0.11	0.08	0.04	0.05	0.07	0.03	0.06	0.07
	last	14.223	1.000	1.000	0.00	1.000	0.10	1.000	0.00	1.000	0.999
ivu02	0.05	1.23	0.45	0.47	0.23	0.30	0.26	0.31	1.96	0.26	0.08
	0.01	2.01	0.55	0.97	0.22	0.45	0.06	0.62	0.57	1.24	–
	0.001	2.86	0.59	2.75	0.08	0.64	0.03	0.92	0.06	2.35	–
	last	30.018	1.000	1.000	0.04	1.000	0.02	1.000	0.01	0.999	0.971
ivu04	0.05	2.55	0.87	0.76	0.37	0.34	0.33	0.62	1.32	0.72	0.13
	0.01	4.24	0.98	1.32	0.28	0.64	0.29	1.11	0.25	2.76	–
	0.001	5.92	1.08	3.76	0.09	0.98	0.09	2.21	0.04	–	–
	last	44.789	1.000	1.000	0.04	1.000	0.04	1.000	0.01	0.999	0.970
ivu05	0.05	56.95	5.41	4.63	0.99	1.68	0.32	3.30	2.59	2.42	1.24
	0.01	92.03	6.85	7.97	0.28	2.93	0.18	6.03	0.47	10.39	–
	0.001	134.54	9.21	16.84	0.15	4.18	0.09	25.68	0.01	–	–
	last	96.104	1.000	1.000	0.05	1.000	0.04	0.999	0.01	0.998	0.962
ivu06	0.05	1163.63	41.43	32.74	0.40	11.86	0.43	21.88	1.89	20.97	–
	0.01	2359.98	55.28	64.72	0.20	18.47	0.18	36.45	1.03	–	–
	0.001	3671.59	72.21	171.20	0.12	28.65	0.06	–	–	–	–
	last	136.651	1.000	1.000	0.05	1.000	0.06	0.999	0.01	0.983	0.946
ivu41	0.05	10.03	20.22	2.65	0.75	1.98	1.00	4.43	5.92	1.50	0.62
	0.01	25.36	21.80	7.97	0.36	4.52	0.29	22.64	0.03	–	3.14
	0.001	44.89	25.01	49.34	0.20	12.93	0.20	–	–	–	–
	last	447.510	1.000	1.000	0.07	1.000	0.07	0.998	0.01	0.985	0.997
ivu41b	0.05	659.49	111.05	19.84	1.00	8.74	1.00	47.01	0.71	15.45	4.29
	0.01	1209.06	120.52	60.29	0.45	21.96	0.46	170.54	0.01	74.87	50.59
	0.001	1745.22	129.80	639.99	0.09	64.81	0.10	–	–	–	–
	last	441.363	1.000	0.999	0.07	1.000	0.08	0.999	0.03	0.990	0.990
ivu59	0.05	4630.36	–	64.03	1.00	62.54	1.00	54.58	1.29	45.02	19.46
	0.01	10303.94	–	161.74	0.78	75.47	0.54	98.51	0.36	462.32	77.12
	0.001	15638.01	–	2482.63	0.10	140.19	0.10	1023.12	0.02	–	–
	last	884.457	–	0.999	0.07	0.999	0.07	0.999	0.02	0.995	0.996
sppaa01	0.05	0.88	0.50	0.25	0.23	0.24	0.12	0.22	0.32	0.15	0.42
	0.01	1.53	0.62	0.68	0.03	0.48	0.04	0.42	0.08	–	–
	0.001	1.86	0.69	–	–	–	–	5.24	0.01	–	–
	last	55535.436	1.000	0.999	0.03	0.999	0.05	0.999	0.01	0.989	0.969
sppaa04	0.05	0.21	0.20	0.20	0.08	0.20	0.04	0.11	0.35	0.07	0.04
	0.01	0.40	0.23	0.43	0.05	0.34	0.10	0.23	0.10	0.49	–
	0.001	0.53	0.26	1.12	0.03	0.47	0.05	1.74	0.03	–	–
	last	25877.609	1.000	1.000	0.02	1.000	0.04	1.000	0.01	0.996	0.987
sppus01	0.05	9.91	17.47	27.73	0.44	12.61	0.10	16.78	1.34	21.33	34.58
	0.01	11.52	18.42	52.11	0.16	15.25	0.09	33.43	0.20	97.78	171.84
	0.001	12.33	18.89	176.67	0.08	18.75	0.05	54.27	0.07	–	–
	last	9963.067	1.000	1.000	0.02	1.000	0.02	1.000	0.01	0.997	0.995

Table 4.2: Comparing Lower Bounds on Set Partitioning Problems

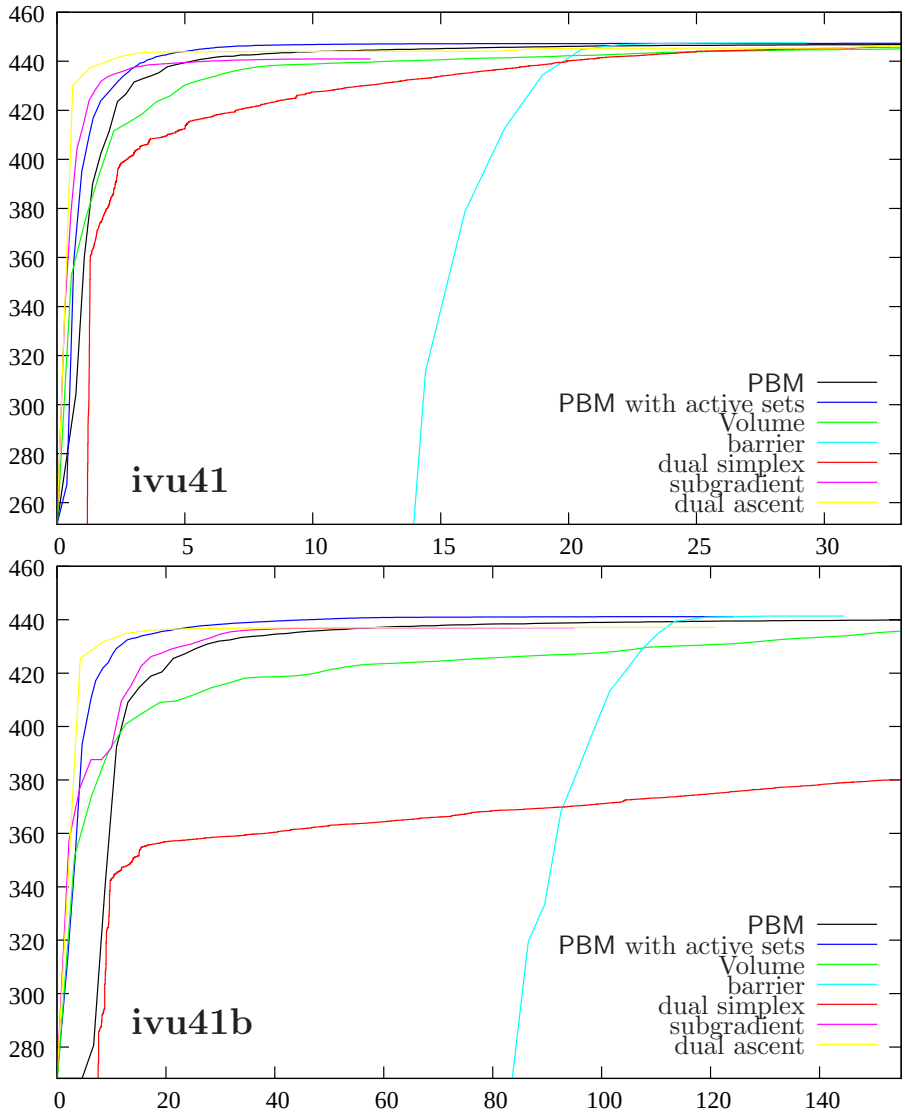


Figure 4.3: Comparison of LP-Algorithms on scenarios ivu41 and ivu41b

The first column in the table shows the name of the problem. For every combination of problem and algorithm the computation time needed to reach a certain optimality gap is shown. The column "ϵ" gives the gap that should be reached. The row with a certain ϵ contains the running time until the respective algorithms found for the first time a lower bound that has at least a value of $1 - \epsilon$ times the optimal value. If no such bound was found, the respective field of the table contains a "–". We show the times for reaching 5%, 1%, and 1‰ gaps. The rows labeled with "last" contains the quotient of

the best value of the corresponding algorithm in comparison to the optimal value calculated by the dual simplex of CPLEX 10.0. The column of the dual simplex contains the optimal function value in this rows.

The row "dual" contains the results of the dual simplex algorithm, "bar" stands for barrier algorithm, "pbm" for proximal bundle method, "pbm_as" for PBM with active set, "Volume" for the Volume algorithm, "sub" for a subgradient algorithm, and "ca" for coordinate ascent. For the Volume algorithm and the PBM we give not only the time until the thresholds are reached, but also the maximal violation of the set partitioning constraints.

The primal approximation of the Volume algorithm is slightly better than the ones of the PBM. This is due to the fact that the Volume algorithm combines the subgradients with the only goal to minimize the norm of the subgradient, while in the PBM a weighted sum of the dual objective value and the negative norm should be maximized. This can be compensated by performing at the end of the PBM some steps with very small weight u^i in the direction finding subproblem 4.4.

The results on the other large problems are similar to the results on ivu41 and ivu41b. On small problems such as ivu01, ivu02, and sppaa04 and on problems with only few rows such as sppus01, the barrier method is competitive with the subgradient methods, and on very small problems even the dual simplex is very fast. On large problems such as ivu41b and ivu59, the PBM has the best results at the 1% threshold, which in our experience suffices to produce good results with our integral heuristic proposed in Chapter 6. But also for the other thresholds the PBM is in general the fastest method to approximate lower bounds on the testbed.

Using active sets additionally accelerates the PBM, which can be seen best at the scenarios with many columns as ivu41b, ivu59, and sppus01. So we can conclude that the PBM with our Active-Set-Method is for large set partitioning problems by far the best solver if the running time is important and exactness of the solution is not necessary.

Chapter 5

The Generation of Duties

This chapter describes the generation of duties in the duty scheduling part of the integrated vehicle and duty scheduling problem (ISP). The methodology described here is also used in the duty scheduling solver DS-OPT, which is part of the Microbus suite of the IVU Traffic Technologies AG.

Generating duties is the most time consuming component of our ISP algorithm. It involves various difficult side constraints on the feasibility and the costs of duties, that are different for virtually every public transit company. Mathematically, duty generation translates into a non-linear shortest path problem that is not only difficult to solve, but also has to be solved very often.

This chapter discusses duty generation in a pure DSP, because Lagrangean relaxation decomposes the ISP into vehicle and duty scheduling parts that are only connected by costs. That is, we have to solve in our ISP-solver IS-OPT a series of duty scheduling problems with varying costs.

5.1 Motivation and Notation

We describe in this section a framework for our discussion of duty generation: We set up the master problem, motivate why it is in general solved by column generation, and formalize the pricing problem.

5.1.1 Master Problem

Recall from Section 1.10 that we have modeled the master problem of our column generation DSP as a set-partitioning-problem with additional base constraints:

$$
\begin{aligned}
&(\mathsf{DSP}) \quad \min\ c^{\mathsf{T}}x + \gamma^{\mathsf{T}}s, \\
&\qquad \text{s. t.} \\
&\text{(i)} \qquad Ax \qquad\quad = 1, \\
&\text{(ii)} \qquad Rx - \quad s \leq r, \\
&\text{(iii)} \qquad x \in \{0,1\}^D, s \geq 0.
\end{aligned}
$$

The set D is the set of feasible duties of DSP. The task set $\bar{V}_{\mathsf{DSP}}$ contains all mandatory tasks that have to be performed by the drivers. These are the atoms of the duty scheduling problem, i.e., each such task has to be assigned to exactly one driver. The matrix $A \in \{0,1\}^{D \times \bar{V}_{\mathsf{DSP}}}$ contains the duty-task-incidence matrix according to constraints (i) and (ii) of (DSP_{y^*}), i.e., the j-th entry in column i of A is one, if task $j \in \bar{V}_{\mathsf{DSP}}$ is contained in duty i and zero else. In particular, a fixed deadhead a with $y^*_a = 1$ is treated like a mandatory task and duties containing deadheads a with $y^*_a = 0$ are fixed to zero. A solution x^* of (DSP) corresponds to a duty schedule which contains exactly the duties d with $x^*_d = 1$. This implies that every mandatory task and every deadhead a with $y^*_a = 1$ is covered exactly once by a duty. The base constraints (ii) model requirements on the entire duty schedule, such as limited number of drivers at certain depots or desired average duty times. $\mathcal{B}$ denotes the set of all base constrains. The entry R_{bd} of matrix $R \in \mathbb{R}^{\mathcal{B} \times D}$ gives the consumption of a resource constraint by base-constraint b of a duty d. The variables $s \in \mathbb{R}^{\mathcal{B}}$ are the slack variables for the base constraints. The cost vector $\gamma \geq 0$ penalizes excess in a base constraint.

5.1.2 Size of the Master Problem

The number of feasible duties of a typical DSP is in general far too large to be stored explicitly in the memory of a computer. It is, for example, possible to construct instances of DSP where each subset of tasks is also a feasible duty. In this case the number of duties is $2^{|\bar{V}_{\mathsf{DSP}}|}$, which is also the trivial upper bound on the number of different duties of one type for a DSP with $|\bar{V}_{\mathsf{DSP}}|$ tasks.

Instances of real world duty scheduling problems have significantly fewer duties than this upper bound, since, among other restrictions, the number

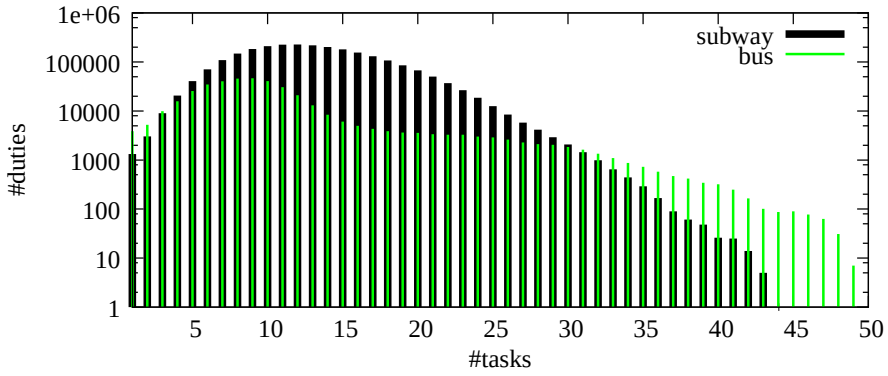


Figure 5.1: Histogram of numbers of duties per number of tasks covered

of tasks in a duty is limited. This is implied by the maximum duty duration arising from legal conditions and by the observation that tasks have a length of at least one minute. If we denote this maximum number of tasks in a duty by k we would get a number of different duties of one type that is bounded by $O(n^k)$, i.e., the number of duties may grow polynomial with a degree equal to the maximum number of tasks in a duty. However, this number is still huge. Once we tried to enumerate all duties of a tiny DSP with $|\bar{V}_{\mathsf{DSP}}| = 81$ that normally is solved in about 10 seconds. After the generation of about 1,500,000 different duties, the memory of the used computer was exhausted.

In our bus transit DSP instances duties may contain up to 50 tasks and the average number of tasks per duty in a typical solution is about 10. The average number of tasks strongly depends on the average length of a task and the maximum duty duration.

In duty scheduling scenarios arising in subway traffic drivers can be relieved at virtually every station. In addition, the average driving time between two stations is fairly small. Thus, for this case the average number of tasks in a duty is substantially larger than in regional scenarios, which have, in general, few relief points. Here we have on average 16 tasks per duty in solutions. In Figure 5.1, a histogram of the number of tasks in unique duties that were generated in the solution process for a DSP of a subway line (with 211 duties in the solution and 3,436 tasks) and of a bus company (417 duties, 3,570 tasks) are shown. In the bus scenario, relatively few duties with many tasks were generated (the y-axis is scaled logarithmically). Those are expensive due to overtime premia and therefore rarely used in a duty schedule. In the subway scenario, the duty with the maximum number of tasks has fewer tasks because a maximum duty duration of only six hours was given there. However, on the average, the duties in the solution contain more tasks than

in the bus scenario, because the tasks are shorter on average.

5.1.3 The Pricing Problem

The dual problem (dual-DSP) of the LP-relaxation of (DSP) is:

$$\begin{aligned} &(\text{dual-DSP}) \quad \max \ \lambda^{\mathsf{T}} 1 + \mu^{\mathsf{T}} r, \\ &\qquad\qquad \text{s.t.} \\ &\text{(i)} \qquad\qquad \lambda^{\mathsf{T}} A + \mu^{\mathsf{T}} R \leq c, \\ &\text{(ii)} \qquad\qquad\qquad -\mu \leq \gamma \end{aligned}$$

Here $\lambda \in \mathbb{R}^{\bar{V}_{\mathsf{DSP}}}$ are the dual variables corresponding to the mandatory tasks of the DSP and $\mu \in \mathbb{R}^{\mathcal{B}}$ are the dual variables for the base constraints.

By the optimality criterion of LP-theory, a dual solution $\lambda \in \mathbb{R}^{\bar{V}_{\mathsf{DSP}}}, \mu \in \mathbb{R}^{\mathcal{R}}$ and a primal solution $x \in [0,1]^D, s \geq 0$ with $\lambda^{\mathsf{T}} 1 + \mu^{\mathsf{T}} r = c^{\mathsf{T}} x + d^{\mathsf{T}} s$ are optimal for the LP-relaxation of (DSP). Thus, if we have optimal primal and dual solutions for a subproblem that contains a subset A' of the columns of A, we only have to prove that there exists no column i of A, such that $\lambda^{\mathsf{T}} A_{\cdot i} + \mu^{\mathsf{T}} R_{\cdot i} > c_i$. Define now

$$\tilde{c}_i := c_i - \lambda^{\mathsf{T}} A_{\cdot i} - \mu^{\mathsf{T}} R_{\cdot i} \tag{5.1}$$

to be the reduced cost of a duty i. Then the optimality criterion is equivalent to: Saying that there exists no duty i with reduced cost $\tilde{c}_i$ smaller than zero. This gives rise to the *pricing problem* to find the column i of A with least reduced cost. If the objective value of this problem is larger than zero, the solution of the subproblem, called *restricted problem* (see also Section 3.1), is also an optimal solution of the original problem. Otherwise, we may add duties with negative reduced cost to the restricted problem, solve the new LP-relaxation again, and so on.

If the pricing problem is solvable in polynomial time the LP can also be solved in polynomial time (e.g.by the ellipsoid method), for a proof see [Grötschel et al. 1993, Chapter 6].

5.2 Description of Duties

We examine in this section properties of duties in detail and how we can efficiently find duties with small reduced cost. We classify duties by types, look at the various attributes of a duty, the resources needed by them, and we show how to calculate their cost.

5.2.1 Duty Elements

A *duty* for a bus driver in public transport is the set of tasks that a driver has to perform throughout a working day. A duty has to fulfill various conditions given by laws and agreements with unions or workers councils. We are focusing in this work on German regulations, although many of them are very similar to the ones in other European countries due to directives of the European Union.

Working conditions for drivers in the EU are regulated by the EU directive 3820/85 which is specified more precisely in national laws. In Germany, these are the working time law (Arbeitszeitgesetz) and the driving time directive (Lenkzeitverordnung), see, e.g., Rang [2006]. These regulations specify conditions that include maximum working and driving time, maximum driving time without a break, and the number and lengths of breaks. Further, each public transit company has certain agreements with the unions or their workers council about additional rules on duties.

A single duty is identified for the purpose of this chapter by a type, by the tasks it contains, and by a cost. The tasks include driving activities arising from timetabled trips as well as other tasks, such as sign-on and sign-off times, times to take a break outside of a vehicle, or getting from one place to another.

Each task has a number of attributes that are needed to decide the feasibility of a duty including this task:

- duration, i.e., the time-span between the begin and the end of the task,
- driving time, i.e., the time of the task, at which the driver is actually driving.
- working time,
- paid time,
- break time, i.e., the time, in which the driver has the possibility to take a break,
- break location, i.e., the location where the driver stays during a possible break. The break location is relevant, because breaks may be evaluated differently depending on the environment of the break location.

The duration and the driving time of a task are independent of the duty in which the task is included. In contrast, the paid time, and the working time can depend on the break rule used. In fact, the break time can be either paid or unpaid working time or no working time at all. The specific rules to calculate the paid time and the break time for a duty differ from one company to another. However, driving time is actually also working time, and working time is paid time, but paid time does not necessary count as working time, and there may other work in a duty than driving.

At last, subsequent tasks in a duty have to be *compatible*, i.e., a single driver must be able to perform them directly one after the other.

5.2.2 Duty types

Often companies do not have one set of rules that applies to all duties, but different sets of rules. These rule-sets partition the set of all feasible duties into sets of duties of the same *duty type*.

There are in general two methodologies for partitioning duties into types. On the one hand, a duty can be categorized by its beginning and starting time. The typical distinction is between *early*, *middle*, and *late* duties, which have to begin and end in certain time windows. This is done to simplify the next planning step, duty rostering. Here a specific driver is assigned to a set of duties for the next weeks. These duty rosters have to fulfill certain requirements, e.g., a late duty may not be followed by an early duty on the next day to give the driver enough time to recover.

Another common method to partition duties into types is by the duration of a duty and the number of its *parts of work*. A part of work of a duty or simply a *part of a duty* is an inclusion maximal continuous time-span, which is completely covered by a duty. A duty may be split into two or sometimes three parts. The time between two parts, called *interruption time*, does not count for the duration of the duty. It has a minimum length of typically at least an hour. If a duty consists of more than one part we call it a *split duty* contrary to a *continuous duty* that consists of one part. Split duties in general have a longer span between the beginning of the duty and its end, called *shift time*, than continuous duties. Common maximum shift times for continuous duties are about 9 hours, whilst shift times of split duties are up to fourteen hours, the most common duration is twelve hours. The total working time of both duty types is typically about the same, the difference in the shift time stems from the interruption time.

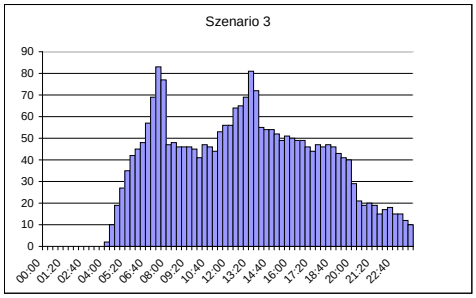


Figure 5.2: Number of timetabled trips throughout a typical work day.

Split duties are needed, since a typical work day has two peaks in the demand for public transport. In Germany, the first one is in the morning, when pupils get to school and the working shifts begin, the second one is in the afternoon, not as high as the first one, but with a larger breadth, when school is over. Bus timetables follow these peaks, i.e., in general, the intervals between timetabled trips are shorter in the peaks than in quieter times of the day. This can be seen in Figure 5.2, where the number of timetabled trips at certain times of a typical workday of an urban traffic company is shown. The interruption time of split duties lies between the peak times, where fewer drivers are needed. Often the total number of split duties is limited by agreements with unions or workers councils, since the drivers want to have a continuous working time, to come home as early as possible. In some countries, e.g., in France, split duties are preferred, since then the drivers have the possibility to eat at home.

Another type of duties are the *short duties*, which have a significantly shorter maximum duration than regular duty types. Short duties are often used in companies which have part time employees with shorter contractual working times. These also help to handle the peak times.

Common are also mixtures of these classifications, such as “early split duties” and “late split duties”, which are both split duties with different time windows for their starting and ending times.

The set of duty types is denoted in the following by $\mathcal{D}$.

5.2.3 Resources

Many of the conditions on duties mentioned above are minimum and maximum constraints on times of different activities or on other quantities in a duty. The most important constraints are concerning the following attributes of duties:

- shift duration, this is the time span between the begin and the end of a duty,
- duty duration, this is the shift duration without the interruption time,
- interruption time,
- working time, this is the time, in which the driver officially works. It includes sign-on and sign-off times as well as the driving times or times for other activities like reserves. It is relevant for various rules,
- driving time, the time that a driver actually drives a vehicle. For example, a driver is not allowed to drive more than 4 and a half hour without break.
- Break time, this is the time, in which a driver officially takes his break,
- paid time, this is the time of a duty that a driver assigned to gets paid. Usually working time is paid time. A duty may also include paid breaks. Sometimes also only a part of a break is paid, to globally compensate for delays,
- number of *pieces of work*, a piece of work is a inclusion maximal continuous subset of a duty, which a driver performs on the same vehicle,
- number of parts of work, usually one for continuous duties or two for split duties.
- driving distance.

We call these different quantities *resources* and say that a resource is *consumed* by a duty. Many rules on duties can be formulated as constraints on the consumption of a resource by a duty. Later we will see that we are able to treat this kind of constraints in an effective way if the consumption of a resource is linear in the tasks used by the duty.

5.2.4 Break rules

Besides the differences in national laws, virtually every traffic company, even in the same country, uses their own specialization and interpretation of the laws and directives concerning break rules for drivers. The following are the most common rules. The first three are arising by directives of the European Union, the last one is a German specialty refining European law:

- The working time of a driver is at most 9 hours per duty.
- The non-preemptive driving time of a driver is at most 4 1/2 hours, after that the driving time must be interrupted by a break.
- Block-breaks: It is possible to take one break of at least 30 minutes (1-block-breaks), two breaks of at least 20 minutes (2-block-breaks), or three breaks of at least 15 minutes (3-block-break).
- Quotient-rules: German law allows to replace block-breaks by breaks of the so-called *quotient-rule*: A duty fulfills the quotient rule, if at least the sixth part of the working time of each continuous time span after a certain working time on the beginning of the duty is break time. Each of these break times must have a duration of at least 10 minutes.

Each part of work has to fulfill these rules separately. Further there are different rules about the minimum and maximum duration of working and driving time between breaks.

Block pauses are in general taken at facilities, where the driver can take a meal and is able to use restrooms. Block pauses are often unpaid. Pauses according to the quotient-rule are normally taken at the turning times of vehicles at the endpoints of lines. Here a driver takes his break “on the bus”. These kinds of breaks can be advantageous for a bus company, since the driver has nothing to do anyway, but they are not liked by the drivers, because in general there is no infrastructure for recovery at the endpoints of lines. Therefore, this kind of breaks often counts as paid time.

5.2.5 Cost of a Duty

The obvious cost for a duty is the sum of money, which the public transit company has to pay to the driver. This quantity is mostly proportional to the

working time of the duty. However, there may be extra charges for overtime and paid break times that are not working time. Which fraction of the break time is paid depends on the break rule.

However, the amount of money paid to the driver is not the only criterion for the quality of a duty. In addition, the drivers should be content with the duty schedule. This does not only improve the working atmosphere, but also reduces the average number of sick days of the staff. This can be modeled by adding costs to duties that are not liked by the drivers. We can, e.g., add a fixed cost to certain duty types such as split duties, or we can penalize the deviation of a resource from a target value if, e.g., a certain duty duration or paid time per duty is wanted. Sometimes also the use of unpaid break time is penalized.

Furthermore, the duty schedule should be robust, i.e., a delay of a single driver should have only small effects on other duties. This can be accomplished by penalizing the use of tight connections of tasks which are prone to delays or by penalizing the change from one vehicle to another, since this is spreading a delay throughout the network.

At last, each duty has typically a fixed cost depending on the duty type to model the preference of public transport companies to not use an unnecessary large number of duties to operate a network.

Let now the constant c_{fix} be the fixed cost of each duty of the current type, c_{pt} is the cost of one unit of paid time. (We computed our solutions with units of one second.) The function $w_{\text{pt}}(i)$ gives the paid time of duty i. The set $\mathcal{R}$ is the set of *resources* of this duty. The constants κ_r^u and κ_r^ℓ are factors to penalize up and down deviation of a resource from its target b_r. The functions $w_r(i)$ give the consumption of resource r by duty i. Then the cost c_i of a duty i can be written as

$$c_i := c_{\text{fix}} + c_{\text{pt}} w_{\text{pt}}(i) + \sum_{r \in \mathcal{R}} \left(\max\{\kappa_r^\ell (b_r - w_r(i)), 0\} + \max\{\kappa_r^u (w_r(i) - b_r), 0\} \right). \tag{5.2}$$

All constants are depending on the duty type of the specific duty. We omit this dependency in the notation for the sake of simplicity.

5.3 Models for the Pricing Problem

In this section, the pricing problem of the DSP is modeled as a shortest path problem with linear and non-linear side-constraints on a pricing network

derived from the duty scheduling network of Section 1.10.2. In particular, we examine subnetworks valid for single duty types and we introduce timelines to reduce the number of arcs of the networks.

5.3.1 Pricing Networks

The generation of duties is done in a pricing network dependent on the duty type, whose nodes correspond to tasks. Its arcs correspond to compatible tasks or to beginnings or endings of duties. These tasks and compatibilities of tasks give rise to one directed and acyclic network for each duty type d, which we call DSP-network of duty type d or d-DSP-network. For this purpose, each task usable by duty type d corresponds to a node in the d-DSP-network. These networks have in general a large number of arcs. In the next section, however, we will discuss a technique to aggregate sets of arcs to compress these graphs, see Section 5.3.2.

The remainder of this and the following sections focuses on an arbitrary but fixed duty type d. That means that the following definitions depend on the duty type d. However, for the sake of simplicity, we will drop duty types from the notation.

A d-DSP-network $N_d := (V_{\mathsf{DSP}} \cup \{s, t\}, A_{\mathsf{DSP}})$ can be formally described as follows: Given is a set of tasks V_{DSP} representing nodes of N. There are two additional artificial nodes s and t, which are the source and the sink of the network. The tasks are connected by a set of arcs A_{DSP}. The node s is connected by arcs to nodes corresponding to tasks that may be the begin of a duty, and the node t is connected to nodes corresponding to tasks that may be the end of a duty. The arcs beginning in s are called pull-in arcs (denoted by B^{duty}) and arcs ending in t are called pull-out arcs (denoted by E^{duty}). We assume that N_d does not contain parallel arcs. This is not a restriction of the generality of this model, because parallel arcs can be removed by adding additional nodes.

Each task $v \in V_{\mathsf{DSP}}$ has a starting time denoted by $\text{st}(v)$ and an ending time denoted by $\text{et}(v)$. The *duration* of a task is given by the difference of starting and ending time. We assume that all tasks have a duration larger than zero. The tasks and arcs also have the list of attributes described in Section 5.2.1. These are needed to model resource consumptions and break rules.

The d-DSP-networks have a natural topological order by starting and ending times of the nodes in V_{DSP}. Thus, they are acyclic, because no arc "goes back in time".

A duty corresponds to a directed st-path in N_d, but not every such a path is also a duty, since a duty has to fulfill requirements in addition to task compatibility, see Section 5.2.

5.3.2 Timelines

A straight forward implementation of the network of Section 5.3.1, adding explicitly all arcs that correspond to immediate subsequent execution of tasks would result in a network that is too large to be handled efficiently. Some important sources of complexity are the following:

- For split duties, there are in most cases lots of possibilities to select a task that begins the next part of work after ending a part of work.
- Urban scenarios, in particular for subways, often feature a large number of relief points that create a large number of possibilities for a driver to change a vehicle.
- In integrate duty and vehicle scheduling (see Section 2) the d-DSP-graph is significantly larger than in the case of sequential planning, because all potential deadheads are included as tasks. Additionally there are more possibilities for a driver to change a vehicle. In fact, a driver may have the possibility to leave the vehicle after each deadhead and may be able to continue his duty with an arbitrary deadhead in the future. Since the number of deadheads may be quadratic in the number of timetabled trips, and the number of arcs corresponding to changes of vehicles, is quadratic in the number of tasks (including deadheads), this could result in $O(|V^T_{\mathsf{DSP}}|^4)$ arcs.

Literature

One approach to deal with large input data is decomposition: Gaffi & Nonato [1997] do not store the possibilities to connect pieces or parts of duties explicitly, but generate a set of pieces or parts of work, and then combine these components of a duty are combined based on rules which define for the compatibility of pieces. But this brings up new problems. Namely, the number of components may become quite large. And the problem of combining the components may be difficult to solve, since the cost and the feasibility of a duty is in general not separable with respect to its components.

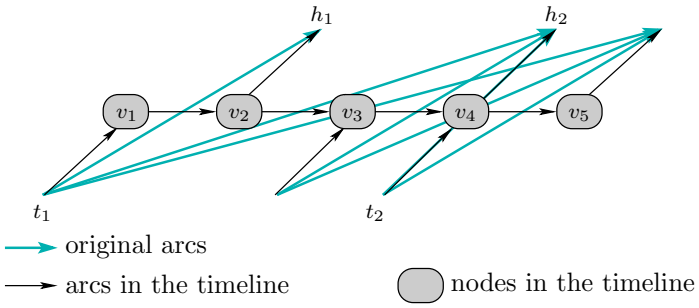


Figure 5.3: Original arcs and their timeline

A large number of arcs was also a problem in the vehicle scheduling approach of Löbel [1997b], where the number of deadhead links, especially the so called long arcs, which represent the combination of a pull-in- and a pull-out-trip, became too large to handle them explicitly. There, the problem was solved by generating the needed arcs dynamically like in a column generation.

Another method to reduce the number of arcs in vehicle scheduling problems are so called *timelines*. They are also used in duty scheduling, see Kliewer et al. [2004]. A timeline is a sub-network of a DSP-network whose nodes model the dwelling of a vehicle or a driver at a certain location. Its arcs either stand for staying at a location, going to it, or leaving it. To our knowledge timelines are first mentioned in Desrosiers et al. [1982].

Timelines in our Algorithm

In our model a timeline $\bar{P}$ is a set of nodes $\bar{V} = \{v_1, \ldots, v_n\}$ and arcs $\bar{A}$ that replaces a set of arcs $A' \subset A_{\mathsf{DSP}}$. The nodes induce a line $v_1, v_2, \ldots, v_{|\bar{V}|}$ where two consecutive nodes are connected by an arc $v_i v_{i+1} \in \bar{A}$ for all $i = 1, \ldots, |\bar{V}| - 1$. These nodes together with the arcs between them model the dwelling of a driver at a certain location p. The nodes have start and end times that are multiples of minutes, e.g., $\mathrm{st}(v_i) = \mathrm{et}(v_i) = i$ minutes. The arcs between the nodes $\bar{V}$ have all a duration of one minute. For every arc $a = uw$ that is an element of A' two arcs are added: One arc uv_i that models the time that a driver needs to get after the completion of task u to the location p. Another arc $v_j w$ models the routing time from location p to the start location of task w. The path from v_i to v_j models the dwelling time of the driver at location p. Obviously $i \leq j$ must hold otherwise a would not be a valid arc. Figure 5.3 shows an example of a set of arcs and its timeline.

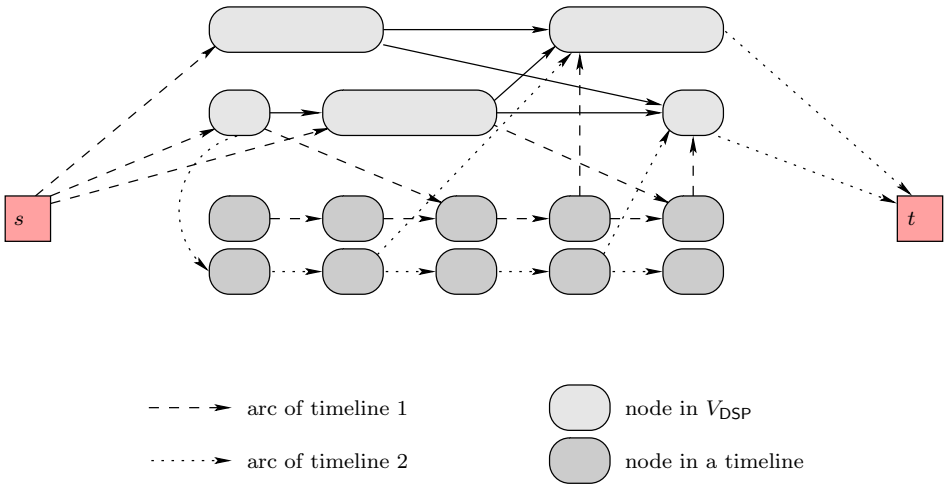


Figure 5.4: d-DSP-network

We use timelines in the pricing problem to model transfers of a driver from one vehicle to another one with a possible break at a rest facility in between. For each rest facility we get one timeline. Taking a break at the rest facility is modeled by the arcs between nodes of the timeline, the other arcs stand for transfers to and from the rest facility. The interruption between to parts of work in a duty is also modeled by timelines. Here the nodes stand for the location where a part of work can end or begin. Timelines could also be used to model deadheads which consist of driving the vehicle to a parking facility, a standing time, and leaving the facility to begin the next trip. However, this is not implemented yet. Figure 5.4 shows a d-DSP-network with two timelines that, e.g., model two different locations where parts of work can end.

Properties of Timelines

A timeline can be seen as an replacement of a set of arcs by a tree such that each original arc is represented by a path in the timeline and vice versa. Therefore the set of arcs that should be replaced by timelines must be selected careful to guarantee that it is able to generate the same duties with both networks. It is, e.g., not possible to restrict the length of paths in the timeline: An arc that enters the timeline can be combined with all arcs leaving the timeline at a later point in time to generate duties. If, e.g., in Figure 5.3 arc t_1h_1 and arc t_2h_2 are elements of A' then also a path from t_1 to h_2 exists, this implies that also arc t_1h_2 is an element of A'.

Another problem are attributes and cost of arcs in A' versus the same costs and attributes of paths in $\bar{P}$. The cost of an arc in A' must be linear in the components of the corresponding path in $\bar{P}$, so that the arc in A' and the corresponding path in $\bar{P}$ have the same cost and attributes. Thus, it is not possible to penalize certain arbitrary arcs in A' costwise if modeled as a timeline. Such penalties are sometimes used to prevent unnecessary line-changes of drivers or to penalize long routing times.

We will now compare the number of arcs in A', versus the arcs in $\bar{P}$ in the best case. We assume that our directed network has m nodes. Since it does not contain directed cycles, it may have at most $\frac{m(m-1)}{2}$ arcs. The timeline needs at most $t_{\max} - 1$ arcs, where $t_{\max}$ is the latest start time of any task. Eventually, the number of arcs in the timeline can be reduced by preprocessing to $\max\{t_{\max} - 1, m\}$ by aggregating subpaths to arcs. There are at most $2m$ arcs entering $\bar{V}$ or leaving $\bar{V}$ from outside the timeline. I.e., we have reduced the theoretical maximum number of arcs from $\frac{m(m-1)}{2}$ to $t_{\max} + 2m - 1$ by aggregating a class of arcs to a timeline.

5.3.3 IP Model

We now present a MIP-model (PRICE) for the pricing problem of (DSP) to find duties with negative reduced cost in the network N_d or to prove that no such duties exists.

We model the pricing problem, denoted by (PRICE) as a shortest path problem with a non-linear objective function, additional resource constraints, constraints for special subpaths, and a possibly exponential number of infeasible-path-constraints. (PRICE) can be seen as a flow-based model as explained in Section 1.3.1.

$\mathcal{R}$ is the set of all relevant resources of duties as explained in Section 5.2.3. Variable y_a, $a \in links$ is one if arc a is used in the solution of (PRICE) and zero otherwise, variables z_r^u and z_r^ℓ, $r \in \mathcal{R}$, give the deviation of the consumption of a resource r from its target. For this model ζ_a, $a \in A_{\mathsf{DSP}}$ is the cost of using arc a, in Section 5.3.4 it is explained how ζ_a is composed. Further, κ_r^u and κ_r^ℓ, $r \in \mathcal{R}$ are the penalties for using more or less of a resource r than a certain target consumption u_r or ℓ_r, respectively. The function $f(y)$ models the non-linear fraction of the objective function. It stems from determining the paid time in a duty dependent on the used break rules and resource consumptions. The set $\mathcal{IP}$ is a family of arcs in infeasible paths.

$$
\begin{array}{lllll}
(\mathsf{PRICE}) & \min & \langle \zeta, y\rangle + \langle \kappa^u, z^u\rangle + \langle \kappa^\ell, z^\ell\rangle + f(y), & & \\
& \text{s. t.} & & & \\
\text{(i)} & & \delta^{\text{out}}(s) & = 1, & \\
\text{(ii)} & & \delta^{\text{in}}(v) - \delta^{\text{out}}(v) & = 0, & \forall v \in V(N_d) \setminus \{s, t\}, \\
\text{(iii)} & & \delta^{\text{in}}(t) & = 1, & \\
\text{(iv)} & & Qy - z^u + z^\ell & = b, & \\
\text{(v)} & & 0 \leq z^u_r & \leq u_r, & \forall r \in \mathcal{R}, \\
\text{(vi)} & & 0 \leq z^\ell_r & \leq \ell_r, & \forall r \in \mathcal{R}, \\
\text{(vii)} & & \textstyle\sum_{a \in I} y_a & \leq |I| - 1, & \forall I \in \mathcal{IP}, \\
\text{(viii)} & & y_a & \in \{0, 1\}, & \forall a \in A_{\mathsf{DSP}}.
\end{array}
$$

Flow Conservation Constraints The equations (i) to (iii) are flow conservation constraints, they force the path to begin at the source s and to end at the sink t.

Resource Constraints The matrix $Q \in \mathbb{R}^{\mathcal{R} \times A_{\mathsf{DSP}}}$ contains in row r and column a the use of a resource r by arc a and its head head(a). The constraints (iv), (v), and (vi) ensure that the consumption of resource r of a path is not less than $b_r - \ell_r$ and at most $b_r + u_r$. The base constraints of the master problem are linked to resource constraints: The resource consumption of a duty i used in a base constraint $b \in \mathcal{B}$ in problem (DSP) is denoted by R_{bi}. Every base constraint controls the consumption of exactly one resource $r \in \mathcal{R}$. It is required that the resource consumption of a duty is separable in its used arcs, therefore it holds that for every $b \in \mathcal{B}$ exists an $r(b) \in \mathcal{R}$ such that

$$R_{bi} = \sum_{a \in i} Q_{r(b)a}. \tag{5.3}$$

Infeasible Path Constraints The infeasible path constraints (vii) are used to model all non-linear side-constraints on duties. $\mathcal{IP}$ is a family of all sets of arcs, which can not be extended to a duty. Although it suffices to consider the inclusion-maximal subsets in $\mathcal{IP}$, $|\mathcal{IP}|$ is in the worst case exponential in the number of tasks.

5.3.4 Cost and Reduced Cost of Pairings and Links

The objective function of the pricing problem (PRICE) is dependent on a dual solution $(\lambda, \mu) \in \mathbb{R}^{\bar{V}_{\mathsf{DSP}}} \times \mathbb{R}^{\mathcal{R}}$ of a restricted dual problem D-DSP^I. We have to construct it in such a way that the objective value of a solution of (PRICE) is equal to the reduced cost of the resulting column in the master problem. We set:

$$\zeta_a := c_{\text{pt}}(\text{pt}(a) + \text{pt}(\text{head}(a)) + c_a - \lambda_{\text{head}(a)} - \sum_{b \in \mathcal{B}} \mu_b{}^{\mathsf{T}} Q_{r(b)a},$$

$c_{\text{pt}} \in \mathbb{R}_+$ is a cost factor for the paid time and $\text{pt}(a)$ and $\text{pt}(v)$ are the minimum times on arc a and task v that have to be paid independently from the actual duty in which a or v are used. These times are mostly the associated working time of an arc or a task. The difference between this time and the real paid time is mainly caused by paid breaks. The rules for paying a break are sometimes very complicated and may depend on break rules, the total break time in a duty, working time directly before or after a break, and various other attributes of a duty. Finally c_a is a cost of arc a that is independent from the the paid time. E.g., arcs leaving the source s get $c_a = c_{\text{fix}}$.

The non-linear function $f : D \mapsto \mathbb{R}_+$ gives for each duty $i \in D$ the difference between the cost of the minimum paid times and its actual paid time. That is:

$$f(i) := c_{\text{pt}} w_{\text{pt}}(i) - \left(\sum_{a \in A_{\mathsf{DSP}}(i)} c_{\text{pt}}(\text{pt}(a) + \text{pt}(\text{head}(a)) \right)$$

For arcs incident to the source s of N_d we add the fixed cost c_{fix} to the arc cost to model the fixed cost of a duty (see Section 5.2.5).

We have constructed the cost function of (PRICE) in such a way that for every duty defined by a solution of (PRICE) its cost is equal to the reduced cost of the same duty in the master problem. That is, the following proposition holds:

Proposition 5.1. Let y^i be a solution of (PRICE) that corresponds to column i of the master problem (DSP), that is, column i covers exactly the arcs $a \in A_{\mathsf{DSP}}$ with $y^i_a = 1$. Let further the resource consumption of duty i be linear in the arcs, that is $w_r(i) := Q_r{}^{\mathsf{T}} y^i$ for all resources $r \in \mathcal{R}$. Then holds

$$c_i - \lambda^{\mathsf{T}} A_{\cdot i} - \mu^{\mathsf{T}} R_{\cdot i} = \zeta^{\mathsf{T}} y^i + \langle \kappa^u, z^u \rangle + \langle \kappa^\ell, z^\ell \rangle + f(y^i)$$

Proof. In any optimal solution y of (PRICE) it holds:

$$z_r^\ell = \max\{b_r - Q_r y, 0\} \quad \text{and} \quad z_r^u = \max\{Q_r y - b_r, 0\}. \tag{5.4}$$

By definition of c_i, definition of f, equation (5.4), and definition of ζ_a it holds:

$$c_i = c_{\text{fix}} + c_{\text{pt}} w_{\text{pt}}(i) + \sum_{r\in\mathcal{R}} \big(\max\{\kappa_r^\ell(b_r - w_r(i)), 0\} + \max\{\kappa_r^u(w_r(i) - b_r), 0\}\big) \tag{5.5}$$

$$= f(i) + \sum_{a\in A_{\mathsf{DSP}}(i)} (c_{\text{pt}}(\text{pt}(a) + \text{pt}(\text{head}(a)) + c_a) + \langle\kappa^\ell, z^\ell\rangle + \langle\kappa^u, z^u\rangle \tag{5.6}$$

$$= f(i) + \sum_{a\in A_{\mathsf{DSP}}(i)} \left(\zeta(a) + \lambda_{\text{head}(a)} + \sum_{b\in\mathcal{B}} \mu_b{}^{\mathsf{T}} Q_{r(b)a}\right) + \langle\kappa^\ell, z^\ell\rangle + \langle\kappa^u, z^u\rangle \tag{5.7}$$

By (5.3) the claim follows. □

5.4 Literature

The pricing problem for the DSP is extensive discussed in the column generation literature. It is often called pairing generation in train or air traffic applications, since there a duty is also denoted by the term pairing.

The literature considers DSP-networks that are given implicitly by rules for the compatibility of tasks or explicitly as a network. Various constraints are discussed, which are mostly part of our model (PRICE). However, some constraints needed for our applications, such as quotient-break-rules, are quasi non-existent in the literature,

We will give now a short summary of different models and solution approaches in recent publications.

Caprara et al. [1997, 2001] describe an application for duty scheduling in railway companies. Their duties (called pairings) have to include exactly one break (like the 1-block-pause) per part and have to have a maximum working time and duty duration. The cost of a duty is either 1 or 2 , if it contains an external rest. Pricing is done in a depth-first-search algorithm with pruning by minimum time constraints. Overtime is checked by calculating the shortest vt-path with respect to the durations.

In Freling [1997]; Freling et al. [2001, 2003] the pricing problem is a shortest path problem with linear resource constraints. The deviations of resource constraints of targets are not considered, i.e., $\kappa^u = \kappa^\ell = 0$ in (PRICE). They also do not consider global constraints. They propose two RCSP-algorithms, a dynamic programming approach and a branch-and-bound algorithm. He recommends branch-and-bound in a depth-first-search manner for DSP's with small maximum duty length α. Here α is the maximum number of tasks any valid duty may contain. The number of paths of length α is at most $|V_{\mathsf{DSP}}|^\alpha$ which is polynomial for a fixed α and tractable for small α. Assumptions in Carresi et al. [1995] are $\alpha \leq 4$ for bus driver scheduling as well as airline crew pairing applications (ACPP). In our scenarios α is in general 40 or larger (see Figure 5.1) and not known in advance. The dynamic programming approach is equivalent to a labeling algorithm in the manner of Section 5.6. The running time of this algorithm depends on the number of labels, which is exponential in the input size. No pruning is mentioned.

In Huisman [2004]; Huisman et al. [2003b] duties have the following special structure: Each duty has zero, one, or two breaks of a certain minimum duration. The number of breaks depends on the duration of the duty. Additionally the duty must not exceed a certain maximum length. Changeovers, i.e. the change from one vehicle-block to another, are only allowed at breaks. The duration of the pieces of work has to be in a certain interval. The cost of a duty is fixed to one. They propose a two-stage algorithm. At first a set of feasible pieces of work is generated by a shortest path algorithm, which is called for each pair of nodes, which span a valid time interval. Then feasible duties are found by enumerating the valid piece combinations. This is possible, since the reduced cost of a duty is the sum of the reduced cost of its pieces plus the fixed cost per duty of 1, and the number of pieces is restricted to 2 or 3. Thus the pricing problem can be solved in polynomial time.

Gaffi & Nonato [1997] propose also a two stage approach, where at first feasible pieces are generated, which are paths in the network. These pieces together with their compatibilities are producing another graph. In this graph feasible duties can be found with a resource constrained shortest path algorithm considering duty time as a resource. It is mentioned that the large number of feasible pieces causes problems.

Our pricing algorithm is based on the procedure of Borndörfer et al. [2003]. Here duties are generate by a depth-first-search algorithm, which utilizes the results of the Lagrangean-relaxation of a resource-constraint-shortest-path relaxation of (PRICE), as well as heuristic pruning criteria.

5.5 Algorithm

Our pricing algorithm to solve the problem (PRICE) works in two phases:

1. At first we solve a Lagrangean relaxation of (PRICE). This gives lower bounds on the reduced cost of all duty segments beginning at tasks $v \in \bar{V}_{\mathsf{DSP}}$ and ending in the sink t of the planning network.
2. Then we do a (heuristically restricted) depth first search in the planning graph, using the information of step 1 to prune the search tree as early as possible.

We will discuss in the next section the calculation of the lower bounds on (PRICE) by Lagrangean relaxation, then follows the description of the depth-first-search algorithm that solves the actual pricing problem utilizing the lower bounds. In later sections we examine also another Lagrangean relaxations that treats some resources exactly.

5.5.1 The Resource Constrained Shortest Path Problem

We relax model (PRICE) to the LP-relaxation of the resource constraint shortest path problem (RCSP). We ignore equation (vii) and remove the non-linear function f from the objective function. Moreover we replace the integrality constraints (viii) by bounds. The resulting model (RCSP) is:

$$
\begin{array}{lll}
\text{(RCSP)} & \min \ \langle \zeta, y\rangle + \langle \kappa^u, z^u\rangle + \langle \kappa^\ell, z^\ell\rangle, & \\
& \text{s. t.} & \\
\text{(i)} & \delta^{\text{out}}(s) = 1 & \\
\text{(ii)} & \delta^{\text{in}}(v) - \delta^{\text{out}}(v) = 0 & \forall v \in V_{\mathsf{DSP}} \setminus \{s,t\} \\
\text{(iii)} & \delta^{\text{in}}(t) = 1, & \\
\text{(iv)} & Qy - z^u + z^\ell, = b, & \\
\text{(v)} & 0 \le z^u_r \le u_r, & \forall r \in \mathcal{R}, \\
\text{(vi)} & 0 \le z^\ell_r \le \ell_r, & \forall r \in \mathcal{R}, \\
\text{(vii)} & 0 \le y_a \le 1, & \forall a \in A_{\mathsf{DSP}}.
\end{array}
$$

The RCSP, also known as acyclic-constrained-shortest-path problem (ACSP), is well studied in the literature, see Mehlhorn & Ziegelmann [2000]

or Dumitrescu [2002] for surveys. The problem is $\mathcal{NP}$-hard already for a single resource constraint, see [Garey & Johnson 1979, A2.3, ND30]. For any fixed number of resources, fully polynomial approximation schemes exist, see Warburton [1987]. Pseudopolynomial algorithms have been developed and successfully used in practical applications, see Desrochers [1986] and others, including penalty treatment, see Desrochers et al. [1992]. Enumerative approaches using Lagrangean lower bounding techniques have been studied by Handler & Zang [1980] and Beasley & Christofides [1989]. Mehlhorn & Ziegelmann [2000] give a geometric algorithm with low computational complexity to solve the LP-relaxation of (RCSP) for the case of a single resource constraint.

5.5.2 Lagrangean Relaxation of all Resource Constraints

The paths found by (RCSP) are in our experience almost never feasible with respect to the break rules and driving- or work-time regulations. Note that for this reason a k-shortest-path-approach, which is often used in crew-scheduling for airlines, does not work properly in a bus transit context. However, it is possible to derive lower bounds for (PRICE) from Lagrangean relaxations of (RCSP). The lower bounds are crucial to guide a depth-first-search algorithm which finds solutions of (PRICE).

There are several ways to apply Lagrangean relaxation to the RCSP. Either we relax the resource constraints (iv) completely. This results in a polynomial subproblem, and one Lagrangean multiplier per resource constraint. Or we relax only some of the resource constraints. Then we get pseudopolynomial subproblems, depending on the number of the resources and the maximum resource consumptions. The more difficult subproblems, however give more information that can be used for pruning in the depth-first-search algorithm.

In our first model, which is also described in Borndörfer et al. [2003], we relax all resource constraints (iv) of (RCSP) in a Lagrangean way. This

results in the following Lagrange-function L:

$$\begin{aligned} L(\mu) = \min & \langle \zeta, y\rangle + \langle \kappa^u, z^u\rangle + \langle \kappa^\ell, z^\ell\rangle + \langle \mu, b - Qy + z^u - z^\ell\rangle, \\ & \text{s. t.} \\ & \delta^{\text{out}}(s) = 1, \\ \delta^{\text{in}}(v) - \delta^{\text{out}}(v) &= 0, \qquad \forall v \in V_{\mathsf{DSP}} \setminus \{s,t\}, \\ \delta^{\text{in}}(t) &= 1, \\ 0 \le z^u_r &\le u_r, \qquad \forall r \in \mathcal{R}, \\ 0 \le z^\ell_r &\le \ell_r, \qquad \forall r \in \mathcal{R}, \\ 0 \le y_a &\le 1, \qquad \forall a \in A_{\mathsf{DSP}}. \end{aligned}$$

We have

$$L(\mu) = \langle \mu, b\rangle + L_1(\mu) + L_2(\mu)$$

with

$$\begin{aligned} L_1(\mu) := \quad & \min \langle \zeta - \mu^{\mathsf{T}} Q, y\rangle, \\ & \text{s. t.} \\ & \delta^{\text{out}}(s) = 1, \\ \delta^{\text{in}}(v) - \delta^{\text{out}}(v) &= 0, \qquad \forall v \in V_{\mathsf{DSP}}, \\ \delta^{\text{in}}(t) &= 1, \\ 0 \le y_a &\le 1, \qquad \forall a \in A_{\mathsf{DSP}}. \end{aligned}$$

and

$$L_2(\mu) := \min_{0 \le z^u \le u} \langle \kappa^u + \mu, z^u\rangle + \min_{0 \le z^\ell \le l} \langle \kappa^\ell - \mu, z^\ell\rangle.$$

The first subproblem is a shortest path problem on an acyclic graph which can be solved in $O(|A_{\mathsf{DSP}}|)$. The second problem has the optimal solution $(\hat{z}^\ell, \hat{z}^u)$ where

$$\hat{z}^\ell_r := \begin{cases} 0, & \kappa^\ell_r - \mu_r \ge 0 \\ l_r, & \text{else} \end{cases} \tag{5.8}$$

$$\hat{z}^u_r := \begin{cases} 0, & \kappa^u_r + \mu_r \ge 0 \\ u_r, & \text{else} \end{cases} \tag{5.9}$$

This problem can be solved in time $O(|\mathcal{R}|)$. Thus, $L(\mu)$ is a lower bound of (RCSP) for each μ, which can be calculated in polynomial time, and $\max_\mu L(\mu)$ is equal to the optimal value of (RCSP), i.e., a lower bound of the reduced cost of the improving duties.

Additional lower bounds will be useful for pruning in duty construction. They can be derived from the duals associated with RCSP. More precisely consider the dual of L_1:

$$\begin{aligned}(\text{SPD}^\mu) \qquad \max \quad & \lambda_t - \lambda_s, \\ \text{s.t.} \quad & \lambda_v - \lambda_u + \mu^\mathsf{T} Q_{\cdot a} \leq \zeta_a, \quad \forall a = (u,v) \in A_{\mathsf{DSP}}.\end{aligned}$$

We denote the arcs that are covered by a path P by $A(P)$. We define the cost of a path P with respect to cost coeffiecients $\xi \in \mathbb{R}^{A_{\mathsf{DSP}}}$ by $\xi(P) := \sum_{a \in A(P)} \xi_a$.

Proposition 5.2. Let $\xi^\mu_a := \zeta_a - \mu^\mathsf{T} Q_{\cdot a}$ be the cost of arc a for all $a \in A_{\mathsf{DSP}}$. Then $\lambda_v - \lambda_s$ is a lower bound on the cost of a shortest path from source s to node v, for all $v \in V(N_d)$.

Proof. The proof is by induction on the length of the path. The claim is trivially true for paths with $v = s$ because the graph is acyclic. Suppose that we have a shortest path P from s to v, $v \neq s$. P can be segmented into a path P' from s to u and its last arc $a = uv$. P' is a shortest path from s to u. By the induction hypothesis $\xi^\mu(P')$ is not smaller than $\lambda_u - \lambda_s$. Therefore the following holds:

$$\xi^\mu(P) = \xi^\mu(P') + \xi^\mu_a \geq \lambda_u - \lambda_s + \xi^\mu_a.$$

Using the constraints of (SPD^μ) we get

$$\lambda_u - \lambda_s + \xi^\mu_a = \lambda_u - \lambda_s + \zeta_a - \mu^\mathsf{T} Q_{\cdot a} \geq \lambda_v - \lambda_s.$$

Thus, the claim $\xi^\mu(P) \geq \lambda_v - \lambda_s$ holds. □

This lower bound can be interpreted as a lower bound on the contribution of P' to the reduced cost of a duty including this subpath.

Analogously $\lambda_t - \lambda_v$ gives a lower bound on the reduced cost contribution of a path from node v to the sink t. We denote these bounds by $f_{sv}(\mu) := \lambda_v - \lambda_s$ and by $f_{vt}(\mu) := \lambda_t - \lambda_v$, respectively.

Finally, we are able to derive lower bounds for duties that cover an arbitrary node v.

Proposition 5.3. Let $\mu \in \mathbb{R}^\mathcal{R}$ be arbitrary Lagrangean multipliers of the Lagrangean function L, let $\lambda \in \mathbb{R}^{V_{\mathsf{DSP}}}$ be an optimal solution of (SPD^μ), and let v be an arbitrary node of the d-DSP-network. Then $f_{sv}(\mu) + f_{vt}(\mu) + \mu^\mathsf{T} b + L_2(\mu)$ is a lower bound on the reduced cost of all duties covering v.

Proof. By proposition 5.2 holds: $f(\mu)_{sv} \leq \xi^{\mu}(P')$ for all sv-paths P' and $f(\mu)_{vt} \leq \xi^{\mu}(P')$ for all vt-paths P'. Therefore $f(\mu)_{sv} + f(\mu)_{vt} \leq \xi^{\mu}(P)$ for all st-paths P that use node v. Let now y^P be the arc-incidence vector of P, that is $y^P_a = 1$ if $a \in A(P)$ and $y^P_a = 0$ otherwise.

Putting this together we get:

$$\begin{aligned} f_{sv}(\mu) + f_{vt}(\mu) + \mu^{\mathsf{T}} b + L_2(\mu) &\leq \xi^{\mu}(P) + \mu^{\mathsf{T}} b + L_2(\mu) \\ &= \zeta^{\mathsf{T}} y^P - \mu^{\mathsf{T}} Q y^P + \mu^{\mathsf{T}} b + L_2(\mu) \end{aligned}$$

And that is $L(\mu)$ if the shortest path uses node v and therefore a lower bound on all shortest paths that use node v. □

5.5.3 Depth-First-Search

The information from the Lagrangean-function L and from (SPD$^{\mu}$) can be exploited to accelerate a depth-first-search duty enumeration algorithm. We propose in the following such an algorithm, which we call DUTYSEARCH, that is used to generate duties with negative reduced cost.

We assume that the pricing network N_d is preprocessed, such that every node in $V(N_d)$ lies on an st-path. The algorithm DUTYSEARCH (see Algorithm 7) starts at source-node s and traverses the network N_d in a depth first search manner. We associate with each node $v \in V(N_d)$ a list of all its outgoing arcs denoted by $\delta^{\text{out}}(v)$. Every arc in a list $\delta^{\text{out}}(v)$ has a unique successor or is the last one in the list. We denote the first arc in $\delta^{\text{out}}(v)$ by $\text{first}(\delta^{\text{out}}(v))$ and a successor of arc a by $\text{succ}(a)$. The successor of an arc is well defined, since every arc is in exactly one list. The state of the algorithm is determined by a stack of active arcs S, which defines an sv-path, where v is the head of the last arc in S.

The loop defined by steps 2-11 adds arcs onto the stack S until S defines a duty or cannot be completed to one. In Step 12 and the following steps a backtrack is performed. Step 8 of algorithm DUTYSEARCH prunes paths utilizing the lower bound of section 5.5.1. There $\xi(S) := \sum_{a \in S} \zeta_a + \sum_{r \in \mathcal{R}} \mu_r Q_{ra}$ is the reduced cost contribution of the sv-path induced by S.

This algorithm is generic in several ways. We may vary the ordering of the arc lists. We can influence the search space by the parameter ϵ. And last but not least we have to efficiently implement the check if S can be completed to a feasible duty.

Algorithm 7 Dutysearch

Input: A duty type d and the network N_d.
Output: A set $\mathcal{D}$ of duties with cost smaller then $-\epsilon$.

1: $\mathcal{D} \leftarrow \emptyset$, S is empty. $v \leftarrow s$,
2: $a \leftarrow \text{first}(\delta^{\text{out}}(v))$.
3: Put a onto the stack S. Let $v \leftarrow h(a)$.
4: **if** $v = t$ **then**
5: If S implies a feasible duty, add it to $\mathcal{D}$.
6: goto step 12.
7: **end if**
8: **if** $\sum_{i \in S} \left(\zeta_i - \mu^{\mathsf{T}} R_i\right) + f_{vt}(\mu) + \mu^{\mathsf{T}} b + L_2(\mu) + \xi(S) > -\epsilon$ or S can not be completed to a feasible duty **then**
9: goto step 12.
10: **end if**
11: goto step 2.
12: **while** a has no successor and S has more then one element **do**
13: Remove a from S and let a be the new last arc of S
14: **end while**
15: **if** a has no successor **then**
16: terminate.
17: **else**
18: Let $a \leftarrow \text{succ}(a)$. Goto step 3.
19: **end if**

5.6 Labeling

We propose in this section a method to improve the lower bounds of Section 5.5.2 using labeling-techniques.

In fact the resource constraints of RCSP can be handled in an exact way by pseudo-polynomial algorithms using *labels* on the nodes of N_d. This produces better lower bounds for vt-paths depending on the resource consumption, but may increase the computation time. To control the computation time we scale and round at the cost of replacing exact bounds by approximations. The details are described in the following sections. We show the methods for the case of one resource to keep the notation simple, however, the techniques of this chapter can be used to handle more than one resource exactly.

5.6.1 Graph Construction

The starting point is that a resource constraints in problem (PRICE) can be handled exactly by transforming the network N_d into a label-network. The transformation is shown as Algorithm 8. It works as follows: let $r \in \mathcal{R}$ be

Algorithm 8 Creating the label-network

Input: A d-DSP-network N_d, a cost vector of its arcs $\zeta \in \mathbb{R}^{A(N_d)}$, resource consumptions of the arcs $Q \in \mathbb{R}^{\mathcal{R} \times A(N_d)}$ and a resource $r \in \mathcal{R}$.
Output: A label-network N_d^r.

```
Let λ_{s_0} ← 0, mark s.
while there are unmarked nodes in N_d do
   Let v be one of the topologically smallest unmarked nodes in N_d.
   for all arcs a ∈ δ^in(v) do
      for all nodes (tail(a)_p ∈ Λ  do
         Add node {v_{p+Q_ar}} to V(N_d^r).
         Add an arc with tail tail(a)_p, head v_{p+Q_ar}, cost ζ_a, and resource
         consumptions Q_·a to A(N_d^r).
      end for
   end for
end while
```

the resource that we want to treat exactly. Then we add for every node $v \in V(N_d)$ a set of nodes $\{v_q\}$ where q is a resource consumption of resource r. For every arc $a = uv \in A(N_d),\ u, v \in V(N_d)$ we add arcs with tail u_q and head $v_{q+Q_{ra}}$ to N_d^r if the corresponding nodes exist in $V(N_d^r)$. The arcs in the label-network inherit their costs and resource consumptions from the arcs in N_d. We call the resulting network the *label-network* and denote it by N_d^r. By construction every $s_0 v_q$-path in N_d^r consumes exactly q units of resource r.

An example of a label-network is shown in Figure 5.5. There v^1 - v^5 denote the original nodes. The nodes v_q^i are the nodes in the label-network that correspond to node v^i of the original network and need q units of resource r to reach s_0. The numbers on arcs give the resource consumption for using the corresponding arc.

The transformation is not polynomial in the input size of RCSP, since the size of N_d^r is $O(|W_r|(|A_{\mathsf{DSP}}|+|V_{\mathsf{DSP}}|))$. Here W_r is the set of potential resource consumptions of paths in N_d. The construction of a label-network is equivalent to label-algorithms like in Dumitrescu [2002] or Desrochers & Soumis [1988].

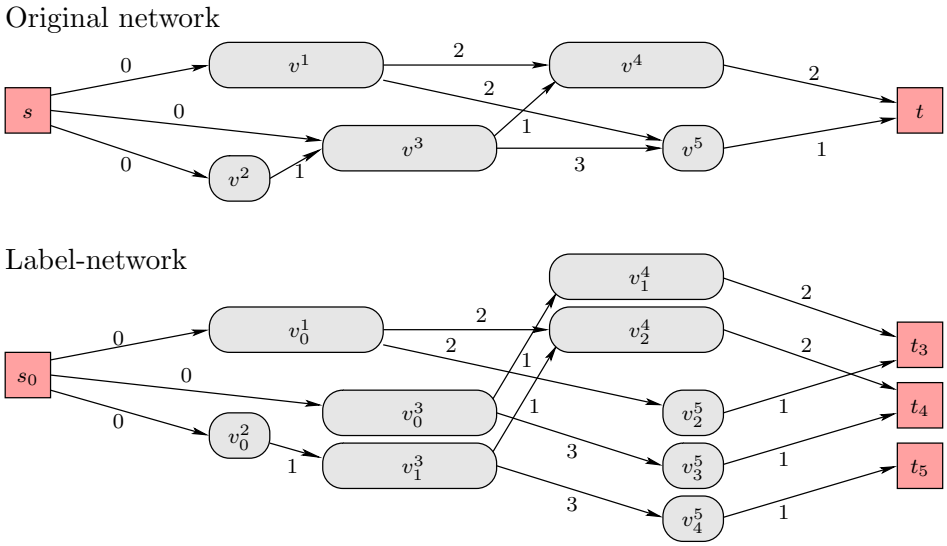


Figure 5.5: A network and its label-network

We use the label-network N_d^r to improve L_1 to a function L_1^r in which resource r is treated exactly. To this purpose, we simply replace the underlying network N_d by the network N_d^r:

$$\begin{aligned} L_1^r(\mu) := \quad & \min\langle \zeta - \mu^\mathsf{T} Q, y\rangle, \\ & \text{s. t.} \\ & \delta^{\text{out}}(s) = 1, \\ & \delta^{\text{in}}(v) - \delta^{\text{out}}(v) = 0, \qquad \forall v_j \in V^r_{\mathsf{DSP}}, \\ & \delta^{\text{in}}(t) = 1, \\ & 0 \le y_a \le 1, \qquad \forall a \in A^r_{\mathsf{DSP}}. \end{aligned}$$

The dual of L_1^r is then:

$$\begin{aligned} (\text{SPD}_\text{r}^\mu)\ & \max\ \lambda_t - \lambda_s, \\ & \text{s. t.} \\ \text{(i)}\quad & \lambda_{v,q} - \lambda_{u,q+\bar{Q}_{ra}} + \mu^\mathsf{T} \bar{Q}_{\cdot a} \le \zeta_a, \forall a = (u_q, v_{q+\bar{Q}_{r(u,v)}}) \in A^r_{\mathsf{DSP}}\,, \end{aligned}$$

Here we have one Lagrangean multiplier per node in V^r_{DSP}, that is $\lambda \in \mathbb{R}^{V^r_{\mathsf{DSP}}}$.

Proposition 5.4. If $\lambda \in \mathbb{R}^{V^r_{\mathsf{DSP}}}$ is an optimal solution of (SPD_r^μ), $\lambda_{v,q} - \lambda_s$ gives lower bounds on a shortest path from node s to v that uses exactly q units of resource r with respect to the objective function $\xi^\mu := \zeta_a - \mu^\mathsf{T} Q_{\cdot a}$.

Proof. By construction of N_d^r every node $v_q \in V_{\mathsf{DSP}}^r, q \in W_r$ can only be reached by paths starting at s_0 that are using exactly q units of resource r . Now proposition 5.4 follows by proposition 5.2. □

Analogously we are able to construct lower bounds on vt-paths that use exactly q units of a certain resource.

If we set: $f_{vt}^q(\mu)$ is a lower bound on the cost of all vt-paths using cost function ξ^μ which consume q units of resource r we are able to replace the condition of Step 8 of Algorithm 7 by:

$$\sum_{i \in S} \left(\zeta_i - \mu^\mathsf{T} R_i\right) + f_{vt}^q(\mu) + \mu^\mathsf{T} b + L_2(\mu) + \xi(S) > -\epsilon,$$

where $q := u_r - \sum_{a \in S} Q_{ra}$ is the maximal remainder of resource r. Since $f_{vt}^q(\mu) \geq f_{vt}(\mu)$ this prunes more of the search tree.

5.6.2 Node Dominance

Solving the shortest path problem on the network N_d^r fast is not easy even if this problem belongs to the computationally easy class of shortest path problems on acyclic networks, because the network N_d^r is in our applications very large, in particular, if we create label-networks for time dependent resources. Consider the following example: A feasible split duty may have a shift duration of up to 14 hours. The typical precision of our input data is one minute. So W_r corresponding to shift duration has 840 elements. A typical pricing-network N_d has about 50,000 nodes and 150,000 arcs. Thus, the label-network may have up to about 100,000,000 arcs and millions of nodes. Such large graphs turned out to be too memory consuming.

If only upper bounds on resources matter we are able to identify nodes that can be removed from N_d^r without changing the optimal value. Thus, we relax the lower bounds on the resource r of (RCSP), i.e., we set $\ell_r = M$, and $M \in \mathbb{R}$ is big enough. A candidate for M is, e.g., the sum of all non-positive entries of Q. In our test instances this relaxation does seldom change the objective value of the RCSP, because in most cases optimal solutions of RCSP tend to consume quantities of a resource that is equal to or near the upper bound.

We now formalize this approach:

Definition 5.5. Let $\lambda \in \mathbb{R}^{V(N_d^r)}$ be an optimal solution of (SPD_r^μ). A node $v_q \in V(N_d^r)$ is *dominated* by a node $v_p \in V(N_d^r)$ if $p \leq q$ and $\lambda_{v,p} \leq \lambda_{v,q}$.

By proposition 5.4 domination of v_q by v_r implies that the shortest path from t to v with resource consumption p is shorter than the shortest path from t to v with the larger resource consumption q. Then the shortest st-path using node v_p does not consume more of resource r and is also not more expensive than the shortest st-path using node v_q. Therefore we can remove the dominated node v_q without increasing the cost of the optimal path with respect to the maximal feasible resource consumption.

If we consider multidimensional labels for more than one resource, a label is dominated if the consumption of every considered resource is larger than or equal to the corresponding resource consumptions of the dominating label. In our experience, the growth in the number of labels is magnitudes larger than the reduction of labels by dominance, if we increase the number of considered resources.

In the reaching algorithm, a standard label-setting algorithm (see [Ahuja et al. 1993, Section 4.4]), dominated nodes can be removed, as soon as their dual variables (called labels there) are computed, since the labels $\lambda_{v,q}, v \in V(N_d^r), q \in W^r$ will not be changed in this algorithm after they have been computed. We show a modified version of this algorithm which dynamically generates the label-network and calculates the dual variables of all non-dominated nodes as Algorithm 9. The result of Algorithm 9

Algorithm 9 Shortest path reaching algorithm with label dominance.

Input: A duty type d, the (acyclic) network N_d, the resource consumption matrix Q, a resource r and a cost vector ζ.

Output: the value of a shortest st-path and labels $\lambda_{v,p}$ for each non-dominated node $v_p \in \Lambda \subset V(N_d^r)$.

1: $\Lambda \leftarrow \{s_0\}$, $\lambda_{s_0} \leftarrow 0$, mark s, $\lambda_{v,p} \leftarrow \infty$ for all $v \in V(N_d) \setminus \{t\}, p \in W_r$.
2: **while** there are unmarked nodes in N_d **do**
3: Let v be one of the topologically smallest unmarked nodes in N_d.
4: **for** all arcs $a \in \delta^{\text{in}}(v)$ **do**
5: **for** all nodes $(\text{tail}(a)_p \in \Lambda$, p arbitrary **do**
6: Let $\lambda_{v,p+Q_{ar}} \leftarrow \min\{\lambda_{v,p+Q_{ar}}, \lambda_{\text{tail}(a),p+Q_{ar}}\}$.
7: $\Lambda \leftarrow \Lambda \cup \{v_{p+Q_{ar}}\}$.
8: Remove dominated nodes from Λ.
9: **end for**
10: **end for**
11: **end while**

is a set $\Lambda \subset V(N_d^r)$ of all non-dominated nodes in N_d^r together with values $\lambda_i, i \in \Lambda$ that satisfy the constraints of (SPD_r^μ). Unfortunately $|\Lambda|$ is in the

worst case equal to the number of nodes in N_d^r and is also very large in our computations.

The dominance check of Step 8 of Algorithm 9 can be done efficiently for one resource. Let

$$\text{label}(v) := (w \in V(N_d^r) \mid w = v_q, q \text{ arbitrary})$$

be the ordered set of nodes of the label-network that corresponds to a node $v \in V(N_d)$ of the original network. Let the nodes v_q in label(v) be ascending sorted by the resource consumptions q. Maintaining the ordering throughout the algorithm costs $O(\log |\text{label}(v)|)$ computation time when adding a new node to Λ and therefore also to the appropriate set label(v). Now holds:

Lemma 5.6. Let label$(v) = (v_{q_1}, v_{q_2}, \ldots, v_{q_m})$, no node is dominated, and $\lambda \in \mathbb{R}^{V(N_d^r)}$ is an optimal solution of (SPD_r^μ). Let the q_i, $i = 1, \ldots m$ be sorted such that $q_1 < q_2 < \cdots < q_m$. Then the associated dual variables are sorted in descending order, i.e., $\lambda_{v,q_1} > \lambda_{v,q_2} > \cdots > \lambda_{v,q_m}$.

Proof. Assume we have to nodes $v_{q_i}, v_{q_j} \in$ label(v) with $i < j$ and $\lambda_{v,q_i} \leq \lambda_{v,q_j}$, then v_{q_j} dominates v_{q_i}. This is a contradiction to our assumption that all labels in label(v) are non-dominated. Therefore follows: $i < j$ is equivalent to $\lambda_{v,q_i} > \lambda_{v,q_j}$ for all $1 \leq i, j \leq m$. □

By this lemma the nodes v_p in the sets label(v) are also sorted by their dual values $\lambda_{v,p}$. Let us now insert node v_p into $V(N_d^r)$ and also into label$(v) = (v_{q_1}, v_{q_2}, \ldots, v_{q_m})$. For the dominance check we need to find an index i such that $q_i < p < q_{i+1}$ and an index j such that $\lambda_{v,q_j} > \lambda_{v,p} \geq \lambda_{v,q_{j+1}}$. If $i = j$ then v_p is not dominated and only dominated $v_{q_{j+1}}$ if $\lambda_{v,p} = \lambda_{v,q_{j+1}}$. If $i > j$ then follows that $q_k < p$ and $\lambda_{v,q_k} < \lambda_{v,p}$ for all $i \geq k > j$. Therefore all nodes v_k are dominated by v_p. If $i < j$ then node v_p is dominated by all nodes v_k with $i \leq k < j$. The indices i and j can be found in $O(\log(\text{label}(v))$, e.g., by binary search, because label(v) is sorted by resource consumption and by dual values. Therefore the dominance check needs for each insertion $O(\log(\text{label}(v))$ time.

5.6.3 Resource Scaling and Rounding

We have seen in Section 5.6.1 that the number of labels depends linearly on the cardinality of the resource domain. The size of the domain can be reduced by scaling and rounding of the resource consumptions.

A way to use rounding and scaling iteratively to calculate exact solutions of the RCSP is described first in Ribeiro & Minoux [1986] and also extensively in Dumitrescu [2002]. We are content with an approximation, because we are primarily interested in lower bounds given by dual-variables to guide the enumeration algorithm Dutysearch.

Scaling and rounding works as follows: assume the coefficients of the resource-consumption matrix Q are integer. We chose a scaling factor $k > 1$ and replace the constraints (RCSP)(iv) by

$$\left\lfloor \frac{Q}{k} \right\rfloor y - \frac{z^u}{k} \leq \left\lfloor \frac{b}{k} \right\rfloor . \qquad \text{(iv b)}$$

The resulting model is again a resource-constraint-shortest-path problem. We denote it by (RCSP^k).

Using constraints (iv b) instead of (iv) in model (RCSP) reduces the number of nodes in the labeling approach to about $1/k$ times the original number of nodes because the number of nodes is linearly dependent on the cardinality of the resource domain. By rounding and scaling this domains shrinks in the best case by the factor k. However, solutions of (RCSP^k) are not necessarily also solutions of (RCSP), because they may violate constrains (iv), and optimal solutions of (RCSP) are in general not optimal for (RCSP^k).

In particular, the lower bound of a minimum cost path calculated in this way may be arbitrarily bad in comparison to the optimal solution of RCSP. In fact, the rounded problem may yield a lower bound of zero even if no feasible path in the original problem exists due to a possible violation of resource constraints. One way to overcome this problem is to use a k-shortest-path-algorithm (see Dumitrescu [2002]). However, this is too time consuming on our instances of the RCSP arising by duty scheduling problems.

In the following we examine the resource consumption of solutions in (RCSP) in comparison to solutions of (RCSP^k). Let $y^* \in \{0,1\}^A_{\mathsf{DSP}}$ be a solution of (RCSP). Then y^* fulfills also all constrains of (RCSP)^k but constraints (iv b) may be violated. The resource consumption of resource $r \in \mathcal{R}$ of the path defined by y^* is then

$$Q_r(y^*) := \sum_{a \in A(N_d)} Q_{ra} y^*_a .$$

The rounded resource consumption is defined by

$$Q^k_r(y^*) := \sum_{a \in A(N_d)} k \left\lfloor Q_{ra}/k \right\rfloor y^*_a .$$

Trivially $Q_r(y^*) \geq Q_r^k(y^*)$ holds for all $k \geq 1$.

Proposition 5.7. Let $y^* \in \{0,1\}^A_{\mathsf{DSP}}$ be a solution of (RCSP) and $r \in \mathcal{R}$ an arbitrary resource. Then there is no factor κ such that $Q_r(y^*) = \kappa Q_r^k(y^*)$ for all $k > 1$.

Proof. Assume the optimal path respecting the rounded constraints (iv b) uses at least one arc and set $Q_{ra} = 1$ for all $a \in A_{\mathsf{DSP}}$. Then holds $Q_r(y^*) > 0$. Further is $\lfloor Q_{ra}/k \rfloor = 0$ and therefore $Q_r^k(y^*) = 0$. □

Thus, the approximation of the resource consumption of a solution of (RCSP) by (RCSP^k) can be arbitrarily bad.

We can also make a proposition about the difference of the resource consumptions:

Proposition 5.8. Let $r \in \mathcal{R}$ be an arbitrary resource. Then holds $Q_r(y^*) - Q_r^k(y^*) \leq n(k-1)$ for arbitrary solutions $y^* \in \{0,1\}^A_{\mathsf{DSP}}$ of (RCSP) with n non-zero elements.

Proof. The largest possible difference of Q_{ar} and $k \lfloor Q_{ar}/k \rfloor$ is $k-1$. Thus $Q_{ar} - k \lfloor Q_{ar}/k \rfloor \leq k-1$ holds for all $a \in A_{\mathsf{DSP}}$. Therefore is $Q_r(y^*) - Q_r^k(y^*) \leq n(k-1)$. Thus, the proposition holds. □

This proof also shows that $Q_r(y^*) - Q_r^k(y^*) = n(k-1)$ in the worst case.

This difference is unsatisfactory from a theoretical point of view. However, if we make an additional assumption on the structure of the network we are able to be more precise about the quality of the resource consumptions in (RCSP^k) in comparison to (RCSP).

Proposition 5.9. Let y^* be an optimal solution of (RCSP) using $n \geq 1$ arcs. Let the average resource consumption of resource r of arcs in y^* be $\bar{p} := Q_r(y^*)/n$. Then holds for all $k \geq 1$:

$$\frac{Q_r^k(y^*)}{Q_r(y^*)} \geq 1 - \frac{k-1}{\bar{p}}.$$

Proof. By proposition 5.8 $Q_r^k(y^*) \geq n\bar{p} - n(k-1)$ holds. Then follows

$$\frac{Q_r^k(y^*)}{Q_r(y^*)} \geq \frac{n(\bar{p} - k + 1)}{n\bar{p}} = 1 - \frac{k-1}{\bar{p}}.$$

□

If the scaling factor k is significantly smaller than the average resource consumption $\bar{p}$, the rounded resource consumption gives a reasonable approximation of the real resource consumption.

5.6.4 Cost scaling

A very similar idea to reduce the number of labels is to scale and round the arc costs, i.e., replacing the coefficients of the objective function of RCSP (assuming that $\zeta, \kappa^u, \kappa^\ell \in \mathbb{Z}$) by

$$\tilde{\zeta} := k \left\lfloor \frac{\zeta}{k} \right\rfloor, \quad \tilde{\kappa^u} := k \left\lfloor \frac{\kappa^u}{k} \right\rfloor, \quad \text{and } \tilde{\kappa^\ell} := k \left\lfloor \frac{\kappa^\ell}{k} \right\rfloor. \tag{5.10}$$

This operation tends to reduce the size of the label-network since each node $v \in V(N_d)$ can produce at most $|C|$ nodes in the label-network, where C is the set of potential cost values. The backside of the medal is that we underestimate the cost of an arc in the worst case by $k-1$. The worst and average gap between the rounded and the original RCSP are the same as for the weight scaling. One has only to replace the resource consumptions on arcs by the costs in the propositions and proofs.

If we use cost scaling in a subgradient algorithm to maximize over a function $L^r(\mu) := L_1^r(\mu) + L_2(\mu)$ another problem occurs. Namely, at each iteration of the subgradient algorithm problem RCSP must be solved with different arc costs to evaluate L_1^r and calculate a subgradient of it. Rounding costs, we cannot guarantee anymore that the minimizing path yields a subgradient for the original formulation. However, using the approximate subgradient technique developed in Chapter 4, we are still able to produce good lower bounds of $\max_\mu L^r(\mu)$.

5.7 Computational Results

We analyze in this section results for solving different instances of the RCSP which occur as subproblems in real world duty scheduling problems. We demonstrate in this way the influence of different scaling factors for the cost and resource consumptions on arcs. We also want to compare the computation times and the approximation quality of different Lagrangean relaxations of (RCSP) solved by a bundle method. Finally, we examine the influence of the lower bounds calculated in the RCSP-algorithm on the solution of the pricing problem.

scenario	ivu08			RVB-Mo-Fr	
duty type	short	normal	split	normal	split
min. duty time	1:00	4:00	6:00	6:30	6:30
max. duty time	6:00	8:00	8:00	9:00	9:00
max. shift time	6:00	8:00	13:00	9:00	12:00
max. num. pieces	5	5	10	3	5
nodes	9,012	9,261	10,286	61,169	155,653
arcs	272,023	272,434	230,292	391,556	461,402

Table 5.1: Characteristics of the test data

5.7.1 Testbed

Our testbed consists of a set of pricing problems arising as subproblems in pure duty scheduling problems and in integrated vehicle and duty scheduling problems. The objective function depends on the dual variables of a restricted LP. We selected dual variables for the restricted (DSP) or (ISP) that were generated in the iteration before we aborted the column generation process to enter the primal phase of the algorithm. We selected these dual variables because they produce a cost structure that makes the pricing problem difficult. In the first iterations of the column generation, we find many different duties with negative reduced cost in short time. This makes it difficult to compare the quality of the different approaches. Also, most of the computation time is needed to find duties when the restricted problem already includes most of the needed duties.

We have generated in these scenarios one instance of (PRICE) for each duty type. The following resources are considered in model (RCSP) for the test-scenarios: shift duration, duty duration, number of parts of work, and number of pieces of work. The shift duration differs from the duty duration only for split duties, also the number of parts is different from one only for split duties. Minimal and maximal allowed resource consumptions for the resources considered in (RCSP) and the size of the graphs are listed in Table 5.1.

The computations were done on a PC with a Intel Xeon CPU with 2.40 GHz and 3GB RAM. The operating system is Linux 2.16. Our algorithms were all compiled with gcc 4.1.0 with full optimization and loop unrolling.

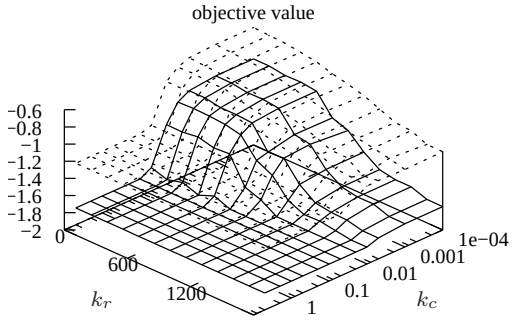


Figure 5.6: RCSP-objective for ivu08 with rounding factors k_r and k_c

5.7.2 RCSP

We calculated approximations of shortest paths by Algorithm 9 with various rounding factors k_c for the cost and k_r for the duty duration. Costs and the factor k_c are scaled by 1/36000 in our algorithm, therefore k_c is fractional. The factor k_r can be interpreted as time slices of length k_r (with a unit of a second in our case) in which all labels are treated as equal. Figure 5.6 shows the impact of different rounding factors on the approximate reduced cost of the shortest path, i.e., the objective value of (RCSP). In all following figures, the continuous graph shows the results of Algorithm 9 where only the duty duration is treated exactly. The dotted graph shows the same problem, but with exact treatment of duty duration and the number of pieces of work. Figure 5.6 shows that small rounding factors for the cost and the resource consumption improve the lower bounds by higher objectives. There is a steep decrease of the objective beyond $k_r = 600$ if pieces of work are not treated exactly. This is due to the fact that the graph contains many sign-on- and sign-off-elements for vehicle changes with a length of about 10 minutes that are all treated as if they have a length of zero, if the rounding factor k_r is larger than 600. If pieces of work are considered exactly, the number of vehicle changes is limited and these sign-on-/sign-off-times do not play such an important role. This effect is also apparent in Figure 5.7. There, the number of pieces of work and the duty duration in seconds of the shortest path found by Algorithm 9 are shown. If k_r is small, also the number of pieces is relatively small, because the shortest path avoids sign-on-/sign-off times. There is also a decrease of the objective value for $k_c \geq 0.01$. It has the same reason, because the objective value predominantly depends on paid time and cost of 0.01 represents 360 seconds paid time.

In Figure 5.8 the number of generated labels and the computation time

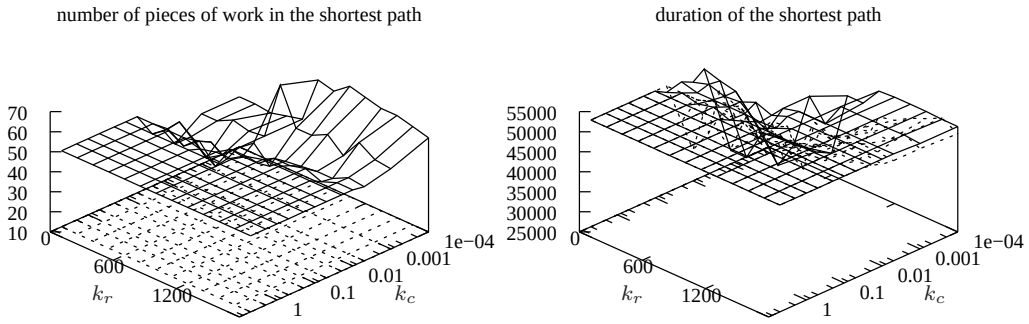


Figure 5.7: Number of pieces of work and duration for scenario ivu08

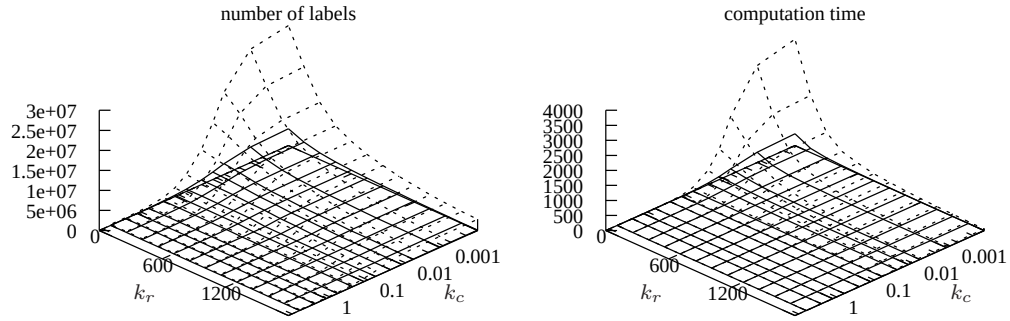


Figure 5.8: Number of labels and computation time for scenario ivu08

of Algorithm 9 are plotted. As expected the computation time is roughly proportional to the number of generated labels. The number of labels is superlinear in the rounding constants. If we consider pieces exactly we need about ten times as many labels as if we ignore the number of pieces. This is due to the maximal number of pieces of work, which is ten. The computation time decreases significantly steeper than the objective value, which is due to the fact that many similar paths with the same or nearby objective value and resource consumption exist, which can be subsumed into one label without much loss of precision.

Figures 5.9 and 5.10 show the results of the RCSP algorithm for continuous and short duties of ivu08. The results are similar to the results for the split duties. The number of labels is for these duty types a magnitude smaller, because the maximum number of pieces of work and the maximum duty time is smaller.

Figure 5.11 shows the results for split duties of a weekday scenario of an urban carrier. The results are very similar to the ones described above. A difference is that the objective value of the shortest path is not very sensitive to the rounding factors, if the number of pieces of work is treated exactly.

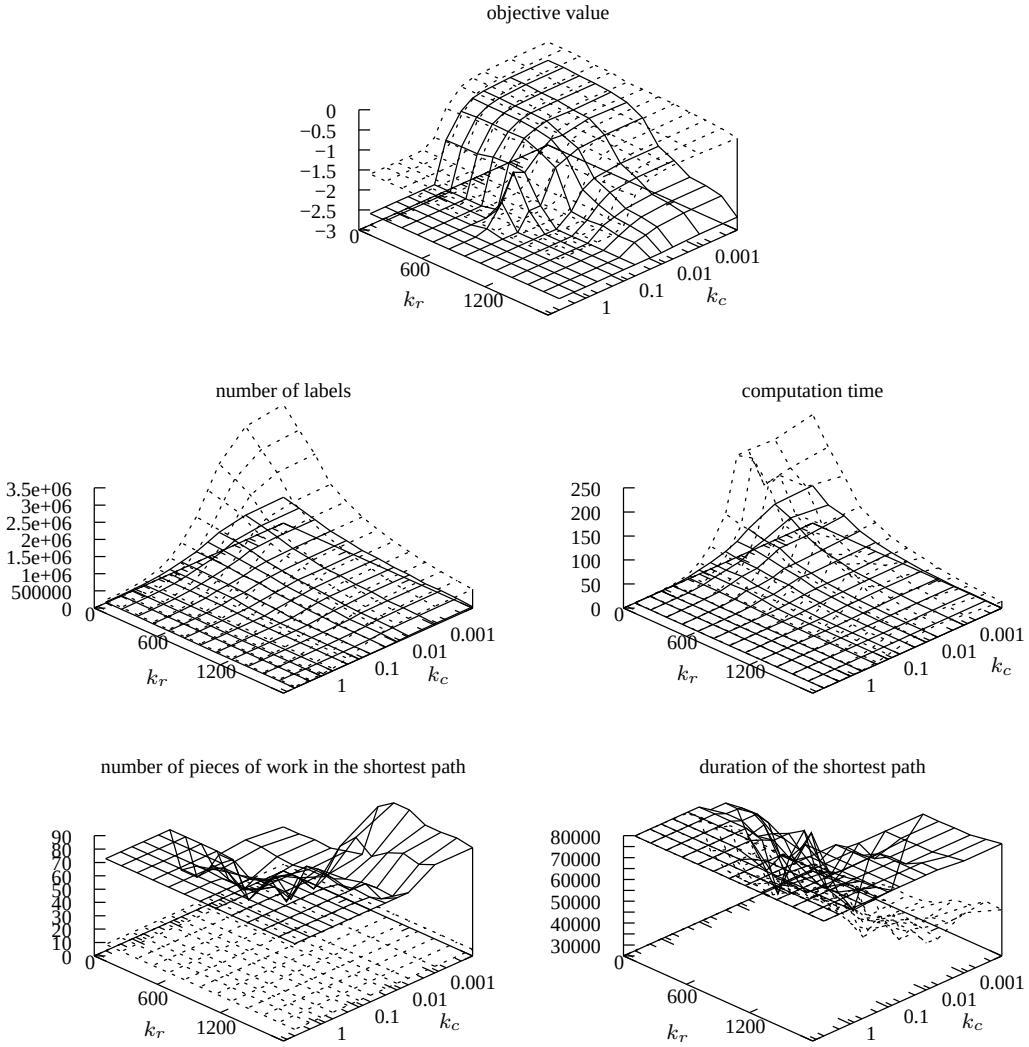


Figure 5.9: Results for continuous duties for scenario ivu08.

Probably the resource constraints of the duty durations are not tight, that is, if we restrict the number of pieces of work, we automatically also restrict the duty duration.

5.7.3 Lagrangean Relaxation

Table 5.2 presents results for solving various relaxations of (RCSP). The objective value of the shortest path in the pricing-network N_d without resource constraints is given in the first row. The second row, "RCSP best", gives the integral solution of the shortest path problem that treats the resources "duty

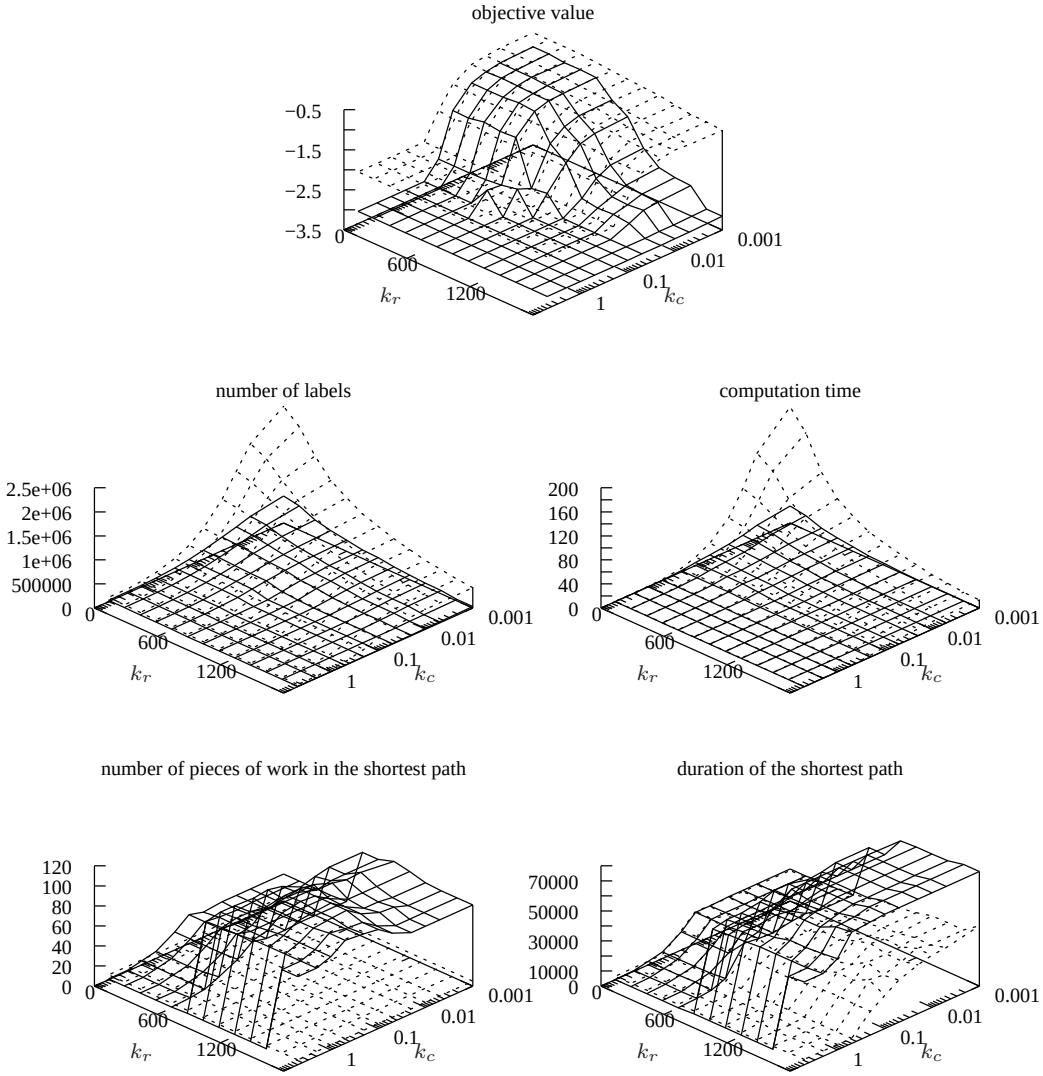


Figure 5.10: Results for short duties for scenario ivu08.

duration" and "number of pieces" exactly but ignores the other resources. Results for the Lagrangean relaxation $\max_\mu L(\mu)$ of (RCSP) proposed in Section 5.5.1 are shown in the row "without labels". Results for the Lagrangean relaxation $\max_\mu L^r(\mu)$ with scaling and rounding described in Section 5.6.4 with resource "duty duration", resource "number of pieces", or with both of them treated exactly are shown in the rows "duration", "pieces" and "both" of Table 5.2. The used scaling factors are $k_r = 600$ and $k_c = 0.01$.

In every case the simple shortest path lower bound is far away from the other lower bounds. This shows that the resource constraints are impor-

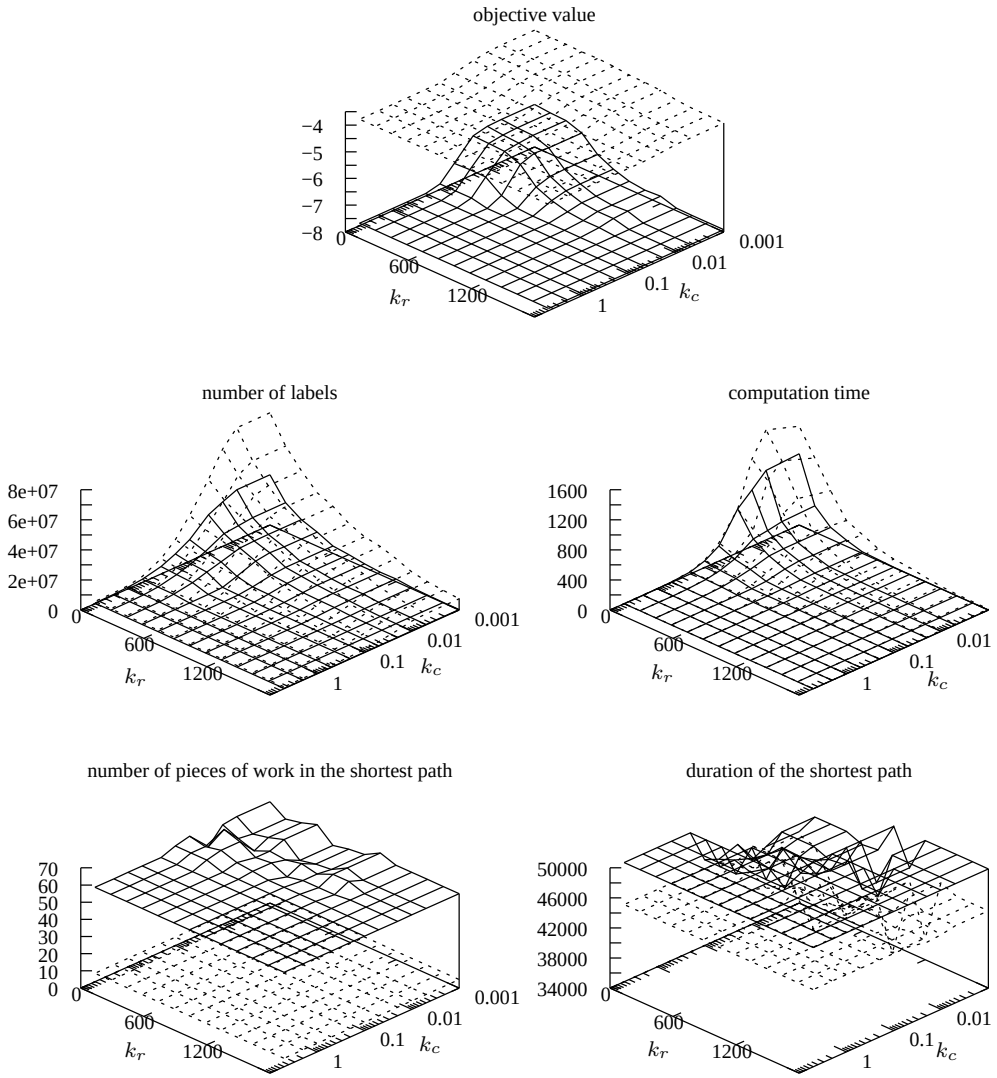


Figure 5.11: Results for split duties for scenario RVB weekday.

tant. For short and continuous duties the result of the Lagrange-relaxation is slightly worse than the solution of "int. duration+pieces". The Lagrangean-relaxation gives lower bounds for the RCSP that are in the best case equal to its LP-relaxation, and therefore a gap to the integral optimum may occur. For split duties the integral solution is slightly worse than the Lagrange-relaxation, because for "int. duration+pieces" we ignore the shift time, which is considered in the Lagrangean-relaxation. Another expected observation is, that if we include more resource constraints in the subproblem of the Lagrangean relaxation, we get better results and need more computing time.

	ivu08 – split duty		ivu08 – continuous		ivu08 – short	
	obj. value	time	obj. value	time	obj. value	time
shortest path	-1.281	< 1	-1.729	< 1	-3.317	0.12
int. duration+pieces	-0.632	3537.53	-0.429	225.78	-0.221	196.34
without labels	-0.645	2.33	-0.500	1.67	-0.305	1.91
duration	-0.628	16.49	-0.463	29.37	-0.284	26.25
pieces	-0.625	48.67	-0.498	9.10	-0.302	11.40
both	-0.620	270.86	-0.455	130.81	-0.259	207.32

Table 5.2: Computations of lower bounds for the RCSP

	best obj. value	#duties	#nodes	time
without labels	-0.26	146,758	10,000,000	119.52
duration	-0.26	161,778	7,141,051	78.90
pieces	-0.29	125,000	10,000,000	86.65
both	-0.30	162,781	6,019,942	72,12

Table 5.3: Depth first search for ivu08 – split duties

5.7.4 Results of the Enumeration Algorithm

Tables 5.3 and 5.4 show the influence of the different lower bounds on Algorithm DUTYSEARCH. The used test instance is again ivu08. We abort the search after 10,000,000 examined search tree nodes, i.e., 10,000,000 different states of S, moreover after accepting 1,000 duties with the same first arc, the next arc incident to s has to be examined. This ensures that our algorithm is searching in different regions of the graph. Therefore in some cases fewer than 10,000,000 nodes were examined.

	best obj. value	#duties	#nodes	time
without labels	-0.23	57,933	10,000,000	103.53
duration	-0.21	65,295	10,000,000	110.98
pieces	-0.23	71,694	10,000,000	102.04
both	-0.23	119,095	10,000,000	115.70

Table 5.4: Depth first search for ivu08 – continuous duties

In Table 5.3 one can see for split duties, that including more resources in the shortest path problems helps Dutysearch. With duty duration and pieces of work treated exactly in the subproblem of the Lagrangean relaxation, Dutysearch found the most duties in shortest time with smallest objective value. For continuous duties, see Table 5.4 the results are similar. However, if we take the computation times for the lower bounds into account (see Table 5.2), the best compromise between computation time and solution quality seems to be to treat only the duty durations exactly.

5.8 Lower Bounds for the Duty Scheduling Problem

Related to the pricing problem is the computation of lower bounds for DSP. We can deduce a lower bound of the master LP, if we find a solution $(\lambda, \mu) \in \mathbb{R}^I \times \mathbb{R}^{\mathcal{B}}$ of the dual problem (D-DSP^I) (see Section 3.3.3) of Lagrangean multipliers λ and μ that implies that no column i with reduced cost $c_i - \lambda^\mathsf{T} A_i x_i - \mu^\mathsf{T} R$ smaller than zero exists. This proof can be accomplished by the Algorithm 7, Dutysearch, if we set its parameter ϵ to zero and select the appropriate objective function. However, this is in in most cases too time consuming since it is equivalent to enumerating all feasible duties of a scenario.

Alternatively it is also possible to calculate a lower bound of the DSP by the lower bounds of the pricing problem. I.e., we are able to compute a lower bound of (DSP) if we have a good lower bound on the least reduced cost of the columns. So we try to improve the lower bound of (RCSP), which is a lower bound of the least reduced cost of the columns of (DSP), by a cutting-plane approach. More details about this can be found in the diploma thesis of Thomas Schlechte (Schlechte [2003]).

5.8.1 RCSP-lower-bound

In Schlechte [2003] an idea to calculate a lower bound of (DSP) is proposed that uses the RCSP-relaxation of the pricing problem.

This idea to calculate a lower bound for (DSP) is to find a solution $(\lambda, \mu) \in \mathbb{R}^I \times \mathbb{R}^{\mathcal{B}}$ of (D-DSP^I) and a real number $\delta \geq 0$ such that $\lambda^\mathsf{T} A + \mu^\mathsf{T} R \leq (1+\delta)c$.

Consider model (δ-DSP):

$$
\begin{aligned}
(\delta\text{-DSP}) \quad & \min\ (1+\delta)(c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z), \\
& \text{s. t.} \\
& \qquad Ax \qquad\ = \mathbb{1}, \\
& \qquad Rx - z \quad \leq r, \\
& \qquad z \in \mathbb{R}^{\mathcal{B}}_{+}, x \in [0,1]^{D}.
\end{aligned}
$$

(δ-DSP) is a (DSP) with a slightly modified objective function. Its dual is

$$
\begin{aligned}
& \max\ \lambda^{\mathsf{T}}\mathbb{1} + \mu^{\mathsf{T}}r, \\
& \text{s. t.} \\
& \qquad \lambda^{\mathsf{T}}A + \mu^{\mathsf{T}}R \leq (1+\delta)c, \\
& \qquad\qquad\qquad \mu \geq -(1+\delta)\gamma, \\
& \qquad\qquad\qquad \lambda \in \mathbb{R}^{V_{\mathsf{DSP}}}.
\end{aligned}
$$

Proposition 5.10. If $\lambda^{\mathsf{T}}A+\mu^{\mathsf{T}}R \leq (1+\delta)c$, then $\lambda^{\mathsf{T}}\mathbb{1}+\mu^{\mathsf{T}}r$ is a lower bound for (δ-DSP) and $(\lambda^{\mathsf{T}}\mathbb{1} + \mu^{\mathsf{T}}r)/(1+\delta)$ is a lower bound for (DSP).

Proof. Let (x^*, z^*) be an optimal solution of δ-DSP. This is automatically also an optimal solution of DSP. By LP-duality it holds that $\lambda^{\mathsf{T}}\mathbb{1} + \mu^{\mathsf{T}}r \leq (1+\delta)(c^{\mathsf{T}}x^* + \gamma^{\mathsf{T}}z^*)$ and therefore also $(\lambda^{\mathsf{T}}\mathbb{1} + \mu^{\mathsf{T}}r)/(1+\delta) \leq c^{\mathsf{T}}x^* + \gamma^{\mathsf{T}}z^*$ for every optimal solution (x^*, z^*) of (DSP). □

If we now find a δ such that the minimum of (RCSP) with cost $(1+\delta)\zeta$ becomes zero or larger, than the preconditions of Proposition 5.10 are fulfilled and we know, that the actual lower bound for the restricted (DSP) is at most $(1+\delta)$ times larger than the minimum value of the master problem.

Computational experiments on our scenarios have shown that typical values for δ of 5-25% can be achieved. In all cases split duties needed the largest δ. For details see Schlechte [2003].

5.8.2 LP lower bounds

The LP-lower bound of the previous section can be improved by including more constraints on duties, i.e. by closing the gap between the model (RCSP) and the model (PRICE) of section 5.3.3. To model non-linear conditions on duties such as the break rules or restrictions on the non-preemptive driving

time, (section 5.2.4) we can add constraints of the type (PRICE)(vii), called $\mathcal{IP}$-inequalities.

Adding $\mathcal{IP}$-inequalities and solving the problem with the MIP-Solver of CPLEX 9.1 turned out to be very time consuming. Problem ivu06 with 6,225 nodes and 134,453 arcs took about an hour on the average to solve the problem (RCSP) for a single duty type to optimality on a Intel Xeon CPU with 2.40 GHz and 3GB RAM.

To compute a lower bound for (DSP) these problems have to be solved repeatedly in a binary search for a minimal δ such that (δ-DSP) is provably optimal. This process was aborted for ivu06 after 36 hours resulting in a lower bound of 21.11% below the best found primal solution of (DSP). For the slightly smaller instance ivu05 with 4,507 nodes and 73,257 arcs we were able to compute in 30 hours and 31 minutes a lower bound of 8.21% below the best primal solution. Calculating lower bound for larger instances with this technique is at the moment too time consuming.

Our computational results have shown that break rules are important and their relaxation (i.e. simply leaving them out of the LP) leads to large gaps between the lower bound and the best found integral solution. Especially the 6-th-break-rule seems to be computational difficult.

5.9 Conclusion

We have introduced an exact model for the pricing problem of DSP and presented an approach to solve it. We have improved the lower bound on the least reduced cost of all columns of (DSP). We have also shown, how labels can be utilized to accelerate a depth-first-search algorithm to find columns with negative reduced cost. At last we have derived a lower bound for the DSP from a lower bound of (PRICE).

Chapter 6

Rapid Branching

In this chapter we present our heuristic, which we call *rapid branching*. Rapid branching can be seen as a special variant of a branch-and-bound algorithm for ILPs with many columns.

We developed rapid branching to solve integer programs arising by duty scheduling and integrated duty and vehicle scheduling problems. Namely we want to solve problem (DSP) of Section 1.10.3. The integer programs that model these problems have a huge number of variables. Therefore we are not able to generate all of them at once (see Chapter 5) and we have to use a column generation approach.

Remember (DSP) was defined as

$$
\begin{array}{lll}
(\mathsf{DSP}) & \min\ c^{\mathsf{T}}x + \gamma^{\mathsf{T}}z & \\
& \text{s. t.} & \\
\text{(i)} & Ax = \mathbb{1}, & \\
\text{(ii)} & Rx - z \leq r, & \\
\text{(iii)} & x \in \{0,1\}^n, & \\
\text{(iv)} & z \geq 0. &
\end{array}
$$

$A \in \{0,1\}^{\bar{V}_{\mathsf{DSP}} \times D}$ is a binary matrix, $R \in \mathbb{R}^{\mathcal{B} \times D}$ is a coefficient matrix of so called resource constraints. The duty-variables x_d, $d \in \{0,1\}^D$ are one if a duty d is selected and zero otherwise. The variables $z \in \mathbb{R}^{\mathcal{B}}_+$ are called slack variables of the base-constraints (ii).

The largest duty scheduling instance solved by our heuristic has about 8,000 rows and millions of columns of which about 1,000,000 are explicitly

generated throughout the algorithm. We think that rapid branching can also be useful for other ILPs with many columns arising in column generation approaches.

6.1 Overview

This chapter is organized as follows: First we explain a generic branch-and-bound algorithm in Section 6.2. Then we explain how rapid branching implements the generic steps of B&B. We present a new branching rule called perturbation branching in Section 6.3, a node selection scheme in Section 6.4 that is mainly a depth-first-search, and calculations of lower and upper bound in Sections 6.5 and 6.6. Section 6.7 compares rapid branching with other MIP-heuristics from the literature. Finally, in Section 6.8 we present computational results of our method applied to set partitioning problems with additional constraints stemming from duty scheduling problems and to set partitioning problems from the literature.

6.2 Branch-and-Bound

Branch-and-Bound, in the following denoted by B&B, is the basic method to solve combinatorial optimization problems. Descriptions of it can be found, e.g., in Papadimitriou & Steiglitz [1982] or [Schrijver 1998, chapter 24]. The idea of B&B is to partition the finite and discrete solution space of a problem into subsets ("branch") and to prune subsets which can be excluded from containing the optimum by its bounds ("bound"). This is done recursively until an optimal solution is found.

We give a generic variant of B&B in Algorithm 10 that solves a combinatorial problem (P) of the type $\min_{x \in X} c(x)$ if lower bounds for it can be calculated. Afterwards we will explain our implementation of the generic steps of B&B to solve (DSP).

The sets $S^0, S^1, \ldots$ of Algorithm 10 partition a subset of the solution set X of (P), which includes by construction an optimal solution, into disjoint solution sets denoted by Q^i_j, $j = 1, \ldots, |S^i|$. At every iteration i of B&B we select one element of S^i denoted by N^i (Step 3). Then we either delete it (Step 7) or partition it into disjoint sets (Step 12). The sets Q^i_j, $i = 0, 1, \ldots,$, $j = 1, \ldots, k^i$, are often organized as the nodes of a search tree: The children of every N^i are the sets $Q^i_j, j = 1, \ldots, k^i$. This search tree has a finite depth

Algorithm 10 Branch-and-Bound (B&B)

Input: A combinatorial problem (P) and a tolerance parameter ϵ.

Output: A solution $x_{\text{ub}} \in X$, its objective value c_{ub} and a lower bound c_{lb} for (P), with $c_{\text{lb}} \geq c_{\text{ub}} - \epsilon$.

1: Let $S^0 \leftarrow \{X\}$. Let $i \leftarrow 0$.
2: **if** $S^i = \emptyset$ **then** terminate.
3: *Node selection*: Select a set $N^i \in S^i$.
4: *Lower bounding*: Calculate a $c^i_{\text{lb}} \leq \min_{x \in N^i} c(x)$.
5: *Upper bounding*: If possible, calculate a solution $x^i_{\text{ub}} \in N^i$ with objective value $c^i_{\text{ub}} := c(x^i_{\text{ub}})$, otherwise set $c^i_{\text{ub}} \leftarrow \infty$.
6: **if** $c^i_{\text{lb}} \geq \min_{1 \leq j \leq i} c^j_{\text{ub}}$ **then**
7: *Pruning*: Set $S^{i+1} \leftarrow S^i \setminus N^i$, $i \leftarrow i + 1$, goto 2.
8: **end if**
9: **if** $\min\{c^j_{\text{lb}} \mid Q^i_j \in S^i\} + \epsilon \geq \min_{1 \leq j \leq i} c^j_{\text{ub}}$ **then**
10: Set $c_{\text{ub}} \leftarrow \min_{1 \leq j \leq i} c^j_{\text{ub}}$, let x_{ub} be the corresponding solution, and let $c_{\text{lb}} \leftarrow \min\{c^j_{\text{lb}} \mid N^j \in S^i\}$. Terminate.
11: **end if**
12: *Branching*: Partition N^i into non-empty disjoint sets Q^i_j, $1 \leq j \leq k^i$ with $k^i \geq 2$ and set $S^{i+1} \leftarrow (S^i \setminus \{N^i\}) \cup \bigcup_{j=1}^{k^i} (\{Q^i_j\})$, let $i \leftarrow i + 1$.
13: Goto 3.

because the solution space of (P) is finite and every child has fewer elements than its parent.

A global lower bound of (P) is given at each iteration i by the smallest lower bound of all subproblems defined by the elements of S^i. A global upper bound is simply the smallest known one.

B&B terminates because X has a finite number of elements and at every iteration either an element of S^i is removed or split into elements of smaller cardinality.

Algorithm 10 contains the following generic steps that we will discuss in more detail:

Node Selection. In Step 3 we have to select an element of S_i. Common strategies are depth first search, i.e., we select an N^i that was generated in the last iteration. With this strategy the cardinality of S^i, i.e., the number

of active nodes of the search tree, keeps small and also the sets N^i become small with few iterations. Thus, this strategy can often produce in short time feasible solutions of the problem.

Another common strategy is to select the node with the smallest lower bound. The lower bounds of all branches of this node together tend to improve the global lower bound. Therefore this strategy is useful to prove optimality of a solution by closing the gap between the objective value of the solution and the global lower bound.

Lower Bound Calculation. Good lower bounds are crucial for the performance of B&B algorithms because they allow to prune branches of the search tree (see Step 6 of Algorithm 10) and thus reduce the number of iterations. If (P) is given as a MIP, lower bounds can be calculated by the LP-relaxation of (P) or by Lagrangean relaxation. If it is computational expensive to calculate lower bounds, the lower bound c^i_{lb} can be set to the lower bound of its parent node.

Upper Bound Calculation. Good (i.e. small) upper bounds help to prune the search tree and therefore to accelerate the algorithm. A trivial method to calculate an upper bound is to set c^i_{ub} to ∞ (or to an $M \in \mathbb{R}$ that is "large enough"), if $|Q^i| > 1$, and to $\min_{x \in P(Q^i)} c^\mathsf{T} x$, if the current subproblem Q^i contains only one element. Furthermore, any $x^0 \in Q^i$ defines an upper bound $c^\mathsf{T} x_0$, but the problem to find a solution may already be difficult.

Branching The decision how to partition a set of solutions Q^i into two disjoint sets is called branching. The solution space of MIPs is partitioned by adding constraints to the subproblems of the new nodes. If a constraint is added that sets a single variable to a specific value, we say the variable is *fixed*. If all variables are fixed and the problem is still feasible, a solution was found.

There is an extensive literature about branching strategies in B&B for MIPs. Recent surveys and newly developed branching rules can be found in Achterberg et al. [2005] and Villeneuve et al. [2003].

6.3 Perturbation Branching Rule

We define at first a new branching rule suited for the MIP (DSP) as defined in Section 1.10.3. This branching rule selects as one branch a set of duty-variables B^* and fixes them to one. The other branches are given by fixing exactly one of the variables in B^* to zero and a subset of the variables in B^* to one.

6.3.1 Motivation

Our MIPs of type (DSP) are arising in column generation approaches. Thus, they have in general much more columns than rows. The DSP is modeled by MIPs with typical sizes of up to 4000 rows and hundreds of thousands of explicitly given columns in the restricted problem and an unknown (but magnitudes larger) number of columns in the so called master problem. This huge number of columns raises problems for B&B-algorithms: The search tree tends to become very large. Also the computations of lower and upper bounds of the subproblems at each node of the search tree are expensive because of the size of the LP-relaxations and the eventually needed generation of additional columns. Therefore we have to develop a branching rule that emphasize the necessity to keep the search tree small.

The idea of our branching rule is to find one branch, called the *main branch*, that fixes as many variables as possibles to one to quickly find a solution of (DSP). This is done by solving a series of LP-relaxations of DSP with varying cost functions $\vec{c}^i$. We perturb the cost function from one iteration to the next in such a way that the solution is likely to become more integral: The cost of variables with large primal values are reduced to move them towards an even higher value or to keep the value at one.

The other branches are unimportant unless the main branch turns out to either not include a feasible solution or to include only feasible solutions with too large costs.

6.3.2 Determining the Main Branch

In this section we show how we find the next main branch, that is, we show how to calculate a set $B^* \subset D$ of variables that will be fixed to one.

Algorithm 11 shows the calculation of the set B^* as pseudo-code. It gets as input a problem of type (DSP). Fixings of duty-variables to one or zero

results again in subproblems of type (DSP) with a smaller solution space. Therefore we state our branching rule simply for (DSP) without considering already fixed variables.

Algorithm 11 get as input a global upper bound c_{ub}, it can also be set to ∞. However, it should be set as small as possible to avoid unnecessary computations. Further the algorithm needs the integrality tolerance ϵ, if a variable d has a primal value not less than $1 - \epsilon$ the algorithm treats it as "nearly one". The integrality weight w weights the importance of the "integrality" of a solution in comparison to its objective value. The bonus weight M and the spacer step interval k_s is needed to determine when to perform a spacer step, as explained below.

Step 1 of Algorithm 11 initializes the iteration counter i, the counter of iterations without improvement k, the objective function $\bar{c}^i$, the best score c^* and the current candidate for B^*. Step 2 calculates at every iteration i an optimal primal solution of the LP-relaxation of (DSP) with the objective function $\bar{c}^i$. In practice, we only approximate this solution by the bundle method of Chapter 4. Then, in Step 3, all x-variables which are nearly one are collected as set $B(x^i)$. Step 4 decides if a *spacer step* should be done, it is performed every k_s iterations without improvement: A spacer step (Step 5) consist of selecting a variable $d \in D$ that is not already in the set B^* and that has maximal primal value x^i_d. Then we reset the objective value to the original one for all other variables that are not in B^*. For variable d and the ones in B^* we set the objective value to $-M$. If M is large enough a spacer step is equivalent to fixing the variables that are nearly one plus one additional one.

Step 7 calculates the score of $B(x^i)$. This score is given by the LP-value of (x^i, z^i) minus a merit for the number of nearly integral duty-variables, which is equivalent to the the cardinality of $B(x^i)$. If the score is better than the current best score we save $B(x^i)$ as B^*. In Step 10 the objective function for the next iteration is calculated. This step is explained in more depth in the next section.

After $k_{\max}$ iterations without improvement of the best score value we terminate the algorithm. If we did not find any suitable variable to be fixed to one, that is $B^* = \emptyset$, we select a duty-variable with the largest value and put it into B^* (Step 12–17).

Algorithm 11 Perturbation Branching Rule

Input: A problem (DSP), a global upper bound $c_{\text{ub}} \in \mathbb{R}$ on the optimal value of (DSP), an integrality tolerance $\epsilon \in [0, 0.5)$, an integrality weight $w > 0$, a perturbation factor $\alpha > 0$, a bonus weight $M > 0$, an interval for iterations with spacer steps $k_s \geq 1$, and a maximal number of iterations without improvement $k_{\max}$.

Output: A set of variables $B^* \subset I$ which can be fixed to one such that the objective value of the induced subproblem is not larger then c_{ub}, if possible, or otherwise an arbitrary column.

1: Set $i \leftarrow 0$, $k \leftarrow 1$, $\bar{c}^i \leftarrow c$, $B^* \leftarrow \emptyset$, and $c^* \leftarrow c_{\text{ub}}$.
2: Solve the LP-relaxation of (DSP) with the current objective function $\bar{c}^i$: Let (x^i, z^i) be its solution.
3: Let $B(x^i) = \{j \in I \mid x^i_j \geq 1 - \epsilon\}$.
4: **if** $(k \mod k_s) = 0$ **then**
5: Select a variable $j^* \in \arg\max_{j \in D \setminus B^*} x^i_j$ and set $\bar{c}^i_j \leftarrow \begin{cases} -M, & j \in B^* \cup \{j^*\}, \\ c_j, & \text{else.} \end{cases}$
6: **else**
7: **if** $c^\mathsf{T} x^i + \gamma^\mathsf{T} z^i - w|B(x^i)| < c^*$ **then**
8: Set $B^* \leftarrow B(x^i)$, $k \leftarrow 1$.
9: **end if**
10: Set $\bar{c}^{i+1}_j \leftarrow \bar{c}^i_j - \alpha c_j (x^i_j)^2$ for all $j \in I$.
11: **end if**
12: **if** $k = k_{\max}$ **then**
13: **if** $B^* = \emptyset$ **then**
14: Select a variable $j^* \in \arg\max_{j \in D} x_j$ and set $B^* \leftarrow \{j^*\}$.
15: **end if**
16: Terminate.
17: **end if**
18: Set $i \leftarrow i + 1$, $k \leftarrow k + 1$.
19: Goto step 2.

6.3.3 Calculation of the Perturbed Objective Function

In all steps of Algorithm 11, which are not spacer steps, the objective function for the next iteration is set to $\bar{c}^{i+1}_d := \bar{c}^i_d - c_d \alpha x^2_d$, for $d \in D$. Here $\alpha > 0$ is a coefficient that controls the rate of change of the cost function. The coefficient α can be seen as a kind of step-size of the series $(\bar{c}^i)_{i=0,1,2,\ldots}$. The idea behind this perturbation is that variables d with higher value x_d get a

higher merit than variables with smaller value. A quadratic perturbation has empirically shown to be working best for our instances in comparison with other perturbation functions, which are, e.g., linear or cubic in x.

Such a perturbation does not guarantee that any variable will get a value above $1-\epsilon$, if $\epsilon < 1/2$ because of the occurrence, e.g., of odd cycles in A (see, e.g., Borndörfer [1998] or Padberg [2001]). In most cases the spacer steps help to overcome this problem, however if they also do not help we simply fall back to a "greedy" strategy in Step 14.

In the literature perturbations of objective functions of LP-relaxations of MIPs to improve integrality can be found which can be seen as a merit function of integrality of variables. An example of such a perturbation embedded in a simplex algorithm is, e.g., Nediak & Eckstein [2001]. There, variables are penalized by their fractionality (i.e., distance to zero or one). For our class of problem this kind of modification of the objective function seems not to be applicable, since there are too many variables which potentially have a value above zero in an optimal solution of the LP-relaxation: We observed that penalizing a fractional variable only results in setting this variable to zero and another variable taking its place of being fractional.

A successful heuristic which perturbs the objective function of large set-partition-problems in a dual ascent method to find integral solutions is described in Wedelin [1995]. However, our computational results of Section 4.7 show that the PBM is better than dual ascent methods in terms of finding better lower bounds. In our experience these better lower bounds also help in the overall performance of DS-OPT and IS-OPT.

6.3.4 Calculation of the Other Branches

In the previous section we have shown how we select a set $B^* \subset D$ of variables that will be fixed to one. This restricts the solution space of DSP and thus gives us one of the new solution spaces of Step 15 of Algorithm 10. We denote this new set of integral solutions by Q_0^i. The complement of Q_0^i is the set of solutions where at least one of the variables in B^* is zero. This could be formulated by adding a constraint of the type $\sum_{i \in B^*} x_i \leq |B_i^*| - 1$ to (DSP) but this destroys the structure of the pricing problem. So we split the complement of Q_0^i into $|B^*|$ sets. We sort the variables in B^* as $d_1, d_2, \ldots, d_{|B^*|}$ by their reduced cost in the root LP. Consider the solution sets

$$Q_k^i := N^i \cap \{x \in [0,1]^D \mid x_{d_1} = 1, \ldots, x_{d_{k-1}} = 1 \text{ and } x_{i_k} = 0\},$$

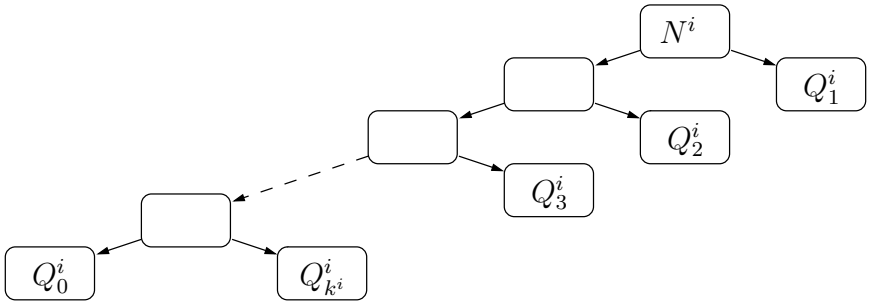


Figure 6.1: The new solution sets at iteration j

with $1 \leq k \leq |B^*|$. They produce a search tree as shown in Figure 6.1.

The sorting of the variables by their reduced cost is inspired by the scoring heuristic for set-covering-problems of Caprara et al. [1996]. If the reduced cost of a column are larger than zero the fixings to one of these column potentially leads to an increase of the objective value. Therefore we will "unfix" at first the columns with the highest reduced cost because we hope that these are the ones which caused the increase.

6.3.5 Comparison with Other Branching Rules

An alternative to branching on variables is presented in Ryan & Foster [1981] for set partitioning problems. This rule can easily be transfered to problems of type (DSP). In the rule of Ryan and Foster on one branch of the search tree all variables are fixed to zero that have a non-zero entry in both of two rows r and s of the coefficient matrix corresponding to matrix A of (DSP). On the other branch all variables are fixed to zero which support exactly one of these rows. This branching rule is a generalization of *branching on arcs* for path covering and partitioning problems (see Section 1.3). For this kind of problems every column of the matrix A of model (DSP) is a node-incidence vector of a path in a graph. We now either exclude all paths from a solution that use arc rs on one branch of the search tree or all paths that include node r or s but not arc rs on the other branch. We call the first branching decision *fixing of arc rs to zero*. The second is called *fixing of arc rs to one*, this is equivalent to combining nodes r and s to a single node. In Villeneuve et al. [2003] a generalization of the concept of branching on arcs is proposed and also a list of articles about branch-and-price can be found.

We now want to motivate why our branching rule is useful for problems of type (DSP) in comparison to other branching rules from the literature.

1. The fixing of a single variable to zero changes the problem very slightly, i.e., generally does not change dual variables or the lower bound, but makes the pricing problem harder because it has to be checked that the column associated to the fixed variable will not be generated again.

2. The fixing of an arc to zero also changes the problem only slightly depending on the total number of arcs. The number of arcs is in general magnitudes smaller than the number of columns, but may still be huge: our largest instances of (DSP) have up to 3,000,000 arcs. However, arc fixing is usually easy to consider in the pricing problem because fixing of an arc to zero translates to removing it from the underlying graph of the pricing problem.

3. Fixing of an arc to one is equivalent to fixing a set of arcs to zero. In practice we have to make too many branching decisions if we branch on arcs to achieve reasonable running times.

4. Fixing of a variable to one is equivalent to fixing all arcs of the corresponding columns to one, so it is easy to be considered in the pricing problem. This operation has the largest impact of all branching variants mentioned here to (DSP).

5. Finally, our branching rule fixes a set of variables at once to one. This can be problematic if it often leads to an increase of the objective value or to infeasible problems. In our case this is the fastest branching rule which leads to high quality solutions.

6.4 Node Selection

We examine the search tree in a depth-first-search manner. In this process we leave out some nodes to be able to consider more different branches of the search tree in the available computation time. We will now explain how we traverse the search tree by describing the node selection of our B&B-algorithm.

In the first iteration of Algorithm 10 the set S^i only has one element. This implies the largest change in the problem. In the following iterations in Step 3 at first the set Q_0^{i-1} of the previous iteration is selected if it is available. Otherwise we select the node $Q^j_{\lceil k^j/2 \rceil}$ with smallest $j < i$ that is available.

If we fix at every iteration only one column, $k^j = 2$ holds for all $j = 0, 1, \ldots$. Then our node-selection-scheme would be a complete depth-first-search. With our branching rule, however, we get in general more than two branches of which only two will be examined. Therefore our method is an heuristical one.

6.5 Lower Bounding

Rapid branching is designed to work with approximate LP-solutions as produced by the bundle method in order to reduce the computation times.

The LPs, which we use to calculate lower bounds at Step 4 of Algorithm 10, are generated by the heuristic column generation approach described in Chapter 5. Our pricing problem solver may possibly not find all columns that would be in an optimal LP-solution. Therefore the objective value c^i_{lb} of the restricted problem only approximates the lower bound. We use in rapid branching c^0_{lb} as the global lower bound because we virtually never calculate the lower bounds of all branches of a nodes. If we find a c^i_{lb} that is smaller than c^0_{lb} – this is possible because of the heuristic column generation – we use this value as global lower bound.

In the column generation process no fixed columns must be generated. If a column is fixed to one, all related nodes can be removed from the planning network. If a column is fixed to zero, we simply remove the column if it is generated again. This is possible for our instances because fixed to zero columns seldom have reduced cost that are significantly below zero.

Since the bundle method approximates a primal solution by convex combinations of solutions of a relaxed problem, this approximation tends to be more fractional then solutions of a simplex algorithm which always return a vertex of the polytope of the LP-relaxation of (DSP). Note, however, that the perturbation of the objective function in our branching rule mitigates this phenomenon.

6.6 Upper Bounding

The matrix A of (DSP) contains in our case by construction a unit-matrix. The columns of the unit matrix are called *slacks*. They have comparable

large cost coefficients c_d, such that they are not in optimal solutions if there exists one which do not has to use slacks.

We calculate the upper bounds of Step 4 by completing the fixed variables to a solution by fixing the slacks to one which correspond to constraints (i) of (DSP) that are not already covered by fixed variables. Then we set the slack variables z_b to $\max\{0, R_b x - r_b\}$. This defines a solution of (DSP) that gives us the upper bound.

6.7 MIP-heuristics in the Literature

In this section we present an overview about methods to solve large scale mixed integer programs similar to (DSP). These are $\{0,1\}$-MIPs, that is MIPs whose integral variables are all binary.

A survey article that covers the developments of MIP-algorithms until 2000 is Johnson et al. [2000].

6.7.1 Simplex Based Heuristics

An alternative to B&B are simplex methods, that try to construct sequences of pivots with all integer bases, see Nediak & Eckstein [2001] and Thompson [2002]. They add a concave merit function to the objective function, which penalizes fractional values of variables and is zero for the values 0 and 1. An example of such a function is $\psi(x) = \sum_{i \in I} x_i(1 - x_i)$ where I is the set of integral variables. They show how one can find columns to pivot into the basis that reduce the merit function or the sum of the merit function and the objective value in a simplex like algorithm. However, there are in general local minima of the merit function as well as of the merit function combined with the original function that are not integral.

Approaches with merit functions as proposed in Nediak & Eckstein [2001] were not successful in our problem instances. The penalization of fractional variables leads in our experience to many iterations without improving the integrality of the solution, because our problems are heavily degenerated. In fact most of the columns have very small or zero reduced cost. Thus, there are many potential variables which can get a value above zero without becoming integral.

6.7.2 OCTANE

The method OCTANE, see Balas et al. [2001], is another approach to get from a (fractional) solution x^* of the LP-relaxation to a solution of a binary ILP. It searches intersections of a half line beginning at x^* with a cross polytope containing x^*. The direction of the half line is selected heuristically. The first k intersections are used to yield k $\{0,1\}$-points as potential solutions of the ILP. OCTANE was tested on problems with up to about 9000 variables. For the tested problems it is shown to be able to produce in short time feasible integral solutions with a small gap to the optimum.

This method seems to be impractical for ILPs with as many variables as on our test instances.

6.7.3 Set Covering and Set Partitioning Heuristics

For some special problems such as set covering, which is in some aspects similar to the DSP, polynomial approximation schemes exist.

A simple method for an $O(\log(k))$-approximation algorithm for set covering is given by Chvátal [1979], where k is the maximum number of non-zero entries in a column. It calculates a score for each column as the quotient of its cost and the number of non-zero entries of uncovered rows. Then it selects the column with the least score and adds it to the solution. This is iterated until all rows are covered.

Similar scoring ideas have been implemented in many set partitioning and set covering heuristics, among them Balas & Ho [1980]; Beasley [1987]; Caprara et al. [1996]; Ceria et al. [1998]; Fisher & Kedia [1990]. These heuristics influenced rapid branching because you can see our branching rule as a scoring of sets of columns. However, scoring-heuristics without the possibility of selecting another branch of the search tree lead in our test-instances to large gaps between the lower bounds and the found solutions.

An exact algorithm for (static) set partitioning problems which relies on B&B and preprocessing of the LPs is described in Borndörfer [1998].

6.7.4 Branch-and-Bound Based Heuristics

Many heuristics were developed which can be seen as variants of B&B. We classify them in Table 6.1. Some of them use only special strategies for the generic steps of B&B, such as Diving, Plunging, or Rounding.

Diving	–	Branch-And-Bound with depth-first node selection and without upper bound. Aborts at first leaf or if subproblem is infeasible.
Plunging	–	Diving with Strong Branching.
Branch-And-Generate	–	Diving with column generation at certain nodes.
Rounding	–	Diving with a branching rule based on a primal LP-solution.
Branch-And-Price	–	Branch-And-Bound with column generation, and suitable branching rules, see Johnson [1989] or Barnhart et al. [1998].
Branch-And-Cut	–	Improve lower bounds by adding cuts.

Table 6.1: Various Branch-and-Bound based methods

Branch-and-Generate

The branch-and-generate heuristic (Bang) of Roy Marsten is a heuristic based on fixing of variables. It sidesteps the problem of fixing variables to zero by avoiding it. We have heard in a conference talk about this heuristic, but unfortunately no citable reference is known to us. In Bang columns are fixed to one by a scoring heuristic and after each fixation a lower bound of the new problem is calculated. If the lower bound increases above a certain threshold, new columns are generated. This is repeated until a solution is found. No column will be fixed to zero.

Bang can be seen as a B&B based heuristic with column generation which uses a depth first search node selection, stopping at the first leaf. Bang branches only by fixing variables to one. The variables are selected by a scoring heuristic. Additionally, the lower bound calculation may be heuristic by subgradient methods and by solving of the pricing problem heuristically. For static MIPs without column generation these kind of heuristics are known as diving heuristics.

Similar heuristics are used by Carresi et al. [1995] and Caprara et al. [1996] to solve set covering problems. They use the information of the Lagrangian multipliers to calculate a score per column and then fix the columns in a greedy fashion. Additionally, not all of the columns of an explicitly given set covering problem are used to calculate the new multipliers, but an subset called *active set*. This is very similar to column generation interpreting the

active set as the restricted problem.

The main drawback of these methods is that wrong decisions, i.e., fixings of columns that lead to a high increase of the objective value of the master problem, are irreversible. It leads to good results, if the problem is either a covering problem with a "nice" cost function, e.g., uniform cost for all columns, or many columns with small support and not too high cost. In the DSP this is not the case, because tripper duties, i.e. duties which violate some minimum constraints, have much larger cost then conventional duties.

Branch-and-Price

Our method can be seen as an heuristical relative of the Branch-and-Price approach described in Barnhart et al. [1998]. Our contributions are a new branching rule, a new node selection scheme, and the use of the proximal bundle method in the branching rule and for the calculation of the lower bounds.

Branch-And-Cut

A branch-and-cut algorithm is a B&B for MIPs that calculates lower bounds by LP-relaxations and improves the integrality of solutions and the quality of the lower bounds by adding cutting planes to the LP-relaxations. A survey can be found in Martin [2001]. The integration of branch-and-cut and column generation, which would be necessary to solve the DSP by branch-and-cut, is very complex. This integration is known as branch-and-cut-and-price (see, e.g., Ralphs et al. [2003]). It could be a topic of further research to include cutting planes in our approach.

6.7.5 Rounding Heuristics

Rounding heuristics are a method to calculate upper bounds of MIPs. They are often mentioned in the literature as an upper bounding tool in branch-and-bound algorithms for set covering, cutting stock and facility location problems. They are, e.g., mentioned in Linderoth & Ralphs [2004]. Also generic rounding heuristics are part of many LP-based branch-and-bound algorithms such as CPLEX.

Rounding heuristics calculate an LP-solution and round some fractional variables of its solution deterministicly or randomly to nearby integral values.

This will be iterated until either the problem is infeasible or a solution is found. Sometimes the feasibility of a problem can be maintained by adding slack variables or is inherent as in set covering problems. For set covering problems it can even be shown that fixing variables in decreasing order of their LP-relaxation values yields a p-approximation algorithm (Hochbaum [1997]), where p is the maximum number of columns which cover a row.

6.8 Computational Results

This sections attempts to do a computational evaluation of rapid branching. For this benchmark, we export all duties generated throughout a run of DS-OPT into an mps-file which can be read by CPLEX[1]. Then we solve these static problems with the MIP-solver of CPLEX 10.0 and with rapid branching without column generation as described in this chapter. The results of such a test allow to make a statement on how well rapid branching does on the set of columns that the heuristic considers explicitly.

The results that we present here all all for problems of type (DSP), that is set-partitioning-problems with additional base constraints. We also implemented rapid branching for VSP as well as for ISP, but we are not able to compare it with standard methods, because the VSP, which is also a subproblem of the ISP, is solved by a special heuristic based on network flows including column generation. At the moment we are not able to export data from VS-OPT which is readable by other solvers.

6.8.1 Testbed

Our testbed consists of instances of (DSP) arising in duty scheduling for public transit and of some instances from the OR-Library of Beasley [1990]. We have generated the DSP-instances by running the algorithm DS-OPT and writing all columns generated throughout the execution of this algorithm into an mps-file. We have selected a set of instances of different size and difficulty, some with and some without base constraints. The names of these instances have the prefix *ivu*. The three set-partitioning instances of the OR-Library of Beasley [1990] have the original names, namely aa01, aa04, and us01. These three instances were considered as hard in van Krieken et al. [2004]

[1]Mps-files of similar scenarios can be found in the contributed section of MIPLIB 2003(Achterberg et al. [2006]).

problem	columns	SPP-rows	bc	non-zeroes	nz/col
ivu01	2,549	81	–	13,404	5.26
ivu02	25,412	185	–	191,819	7.55
ivu04	37,764	346	–	291,222	7.71
ivu05	159,438	847	–	1,450,704	9.10
ivu06	980,578	1,177	–	10,565,680	10.77
ivu41	209,581	3,569	–	1,520,393	7.25
ivu52	117,466	2,110	6	1,619,649	13.79
ivu53	560,233	2,421	6	12,840,061	22.92
ivu60	1,148,050	1,979	2	21,384,769	18.63
aa01	8,904	823	–	72,965	8.19
aa04	7,195	426	–	52,121	7.24
us01	1,053,137	145	–	13,636,541	12.95

Table 6.2: Characteristics of the Testbed

and are the largest in the OR-library. They are all solvable to optimality in short time, see Borndörfer [1998]. The characteristics of all instances are shown in Table 6.2. The column "problem" contains the name of the scenario, "columns" is the number of columns of the coefficient matrices A and B, "SPP-rows" is the number of rows of matrix A, "bc" is the number of base constraints (equal to the number of rows of matrix B), "non-zeroes" contains the number of non-zero entries of matrix A. The last column "nz/col" gives the density of matrix A defined by the quotient of the the number of non-zero entries and the number of columns of A.

The results of the algorithms are shown in Table 6.3. The times are all in seconds, if not mentioned otherwise. The column LP-bound shows the result of the barrier algorithm of CPLEX 10.0. This is a lower bound for the best integral solution of the problem. In column DS-OPT the result of our duty scheduling algorithm using rapid branching is shown. In column "static" the result of the rapid branching heuristic on the static problem consisting of all columns generated in DS-OPT is shown. In the column CPLEX the result of the MIP-solver of CPLEX 10.0 on the same problem is shown. We use the barrier method as start- and subproblem LP-solver, because this is the fastest method in CPLEX to solve our type of LPs. We have set the integrality tolerance to 1% since this is also the termination criterion of DS-OPT.

We state in Table 6.3 for the algorithm "static" the running times until

problem	LP-bound	DS-OPT		static		CPLEX	
		obj.	time	obj.	time	obj.	time
ivu01	14.22	14.34	6	14.34	<1	14.32	<1
ivu02	29.87	30.29	116	30.39	13	30.06	26
ivu04	44.79	45.17	179	45.21	22	45.20	924
ivu05	96.10	96.66	448	96.18	62	96.11	276
ivu06	136.00	137.81	5,263	147.55	740	–	> 4d
ivu41	447.71	448.82	3,149	450.35	380	451.13	1,101
ivu52	490.46	494.65	2,072	496.15	300	495.83	8,000
ivu53	261.33	263.90	9,098	(265.31)	609	263.57	13,238.34
ivu60	159.99	162.34	29,982	(172.22)	1353	–	> 4d
aa01	55,535.43	–	–	57,832	56	56,364	128
aa04	25,877.61	–	–	27,202	42	26,433	34
us01	9,962.64	–	–	10,098	176	10,090	335

Table 6.3: Results of integral heuristics, all times in seconds

the solutions of this table were found. For the other algorithms the total running time is given.

6.8.2 Observations

For large problems (ivu06, ivu53, ivu60) CPLEX has problems to find good solutions in reasonable time or is significantly slower than DS-OPT using rapid branching. The MIP solver of CPLEX gave for ivu06 a solution with an integrality gap of $> 60\%$ after four days of computation. For ivu60 we got only a segmentation fault with CPLEX, we assume that this was an issue with the memory management because on a computer with 64GB RAM CPLEX began the optimization, however, this computer is too slow and not comparable with the PC used for the remainder of our computations. Problem ivu53 seems to have a nicer structure than ivu06. CPLEX was able to find a near optimal result in about 3h 40m. This is slower than the time of DS-OPT, with about 2h 30min, although DS-OPT also generates the columns of (DSP), what takes most of the computation time. This can be seen in the computation time of the static rapid branching, which only needs about 10min. The solutions found by DS-OPT have gaps from 0.1–1.44% between the LP-bounds of the restricted problems and the best solutions..

The static variant is not able to reproduce the quality of the solution

of the dynamic column generation approach of DS-OPT. The cause of this phenomenon is that DS-OPT is able to generate new columns that are missing after the fixing of some variables, whereas in the static variant only the columns that were previously generated for a certain search tree are available. Thus, if the static variant makes a wrong decision, i.e.leaves the search tree of DS-OPT, it seems that the static approach is not to able to find out what went wrong and then to select the right node of the search tree.

For the medium problems ivu41 and ivu52 CPLEX has significantly longer running times than static rapid branching. However, we have to admit that static rapid branching is not able to improve this solution further after finding the solutions shown here. Here, it becomes apparent that our algorithms lacks sophisticated preprocessing and cutting planes which are needed to find optimal solutions of problems with a restricted number of columns in reasonable time. On the other hand, rapid branching with column generation, consistently finds good solutions in short time.

The problems ivu04 and ivu05, which are solvable by CPLEX in reasonable time, static rapid branching is also significantly faster than CPLEX in finding solutions with small integrality gap. DS-OPT's solution quality is comparable to static branching, but its running time is longer because of the column generation. These problems seemingly include enough columns such that also static rapid branching works well.

For small problems such as ivu01, ivu02 CPLEX is faster or more accurate (or both) than static rapid branching. The longer running times of DS-OPT come from the time consuming column generation. However, the absolute differences of the running times and the solution qualities between the algorithms are small.

The instances aa01 and aa04 have relatively few columns, so it is not surprising that our algorithm, which is tuned for problems with many columns, had problems to improve the solutions that it found in short time. Further is the gap of the objective value of the solution of us01 found by rapid branching to the optimal solution found by CPLEX below 0.1%. However, rapid branching was not able to abort properly because it is not able to find a lower bound which is within the optimality tolerance.

6.8.3 Conclusion

We have seen that we are not able to solve large problems with a standard MIP-solver, such as CPLEX. For this problems also a static variant of rapid

branching without dynamic column generation fails. Only rapid branching combined with a dynamic column generation approach produces good solutions.

Also for medium sized instances the column generation algorithm DS-OPT produces the best results, but here also the static variant of rapid branching is able to produce solutions of the desired quality faster than CPLEX. For small problems or problems as in the OR library, where a significant gap between the LP-lower bound and the optimal solution exists, CPLEX gives the best results.

Rapid branching is applicable in a column generation framework, especially for problems with a large set of columns, where the fixing of one column probably has not so dire consequences as in integer programs with only few columns. This is also the reason why the static version of rapid branching produces worse solutions than DS-OPT for large instances. In the static case the algorithm comes to a point where many columns are fixed to one and the remaining free rows cannot be covered without increase of the lower bound. Here the dynamic algorithm is able to generate the missing columns.

Chapter 7

Computational Results

This chapter reports 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 core Intel Xeon 3.0 GHz CPU running SuSE Linux 9.0. The computation times in the following tables are in hours:minutes.

7.1 Running Time

Figure 7.1 shows a typical runtime chart of our algorithm IS-OPT. This one is from the subcontractor scenario of Section 7.6. The x-axis measures time in seconds, the y-axis gives statistics in two different scales, namely, on the right side, the number of duties generated (#columns). The scale on the left side is for the objective value of the current approximation of ISP, that is of function f of Section 4.6.4, which gives lower bounds on the objective value of the integrated scheduling problem restricted to the current column set. This scale shows also the number of deadheads fixed to one divided by 5 (#fixed deadheads) and the Euclidean norm of $\tilde{g}^i$, cf. equation (4.56), which can be seen as a measure for the quality of the LP-relaxation of (ISP).

In the first phase of the algorithm up to point A in Figure 7.1 a starting set of columns is generated with all zero Lagrangean multipliers. In principle

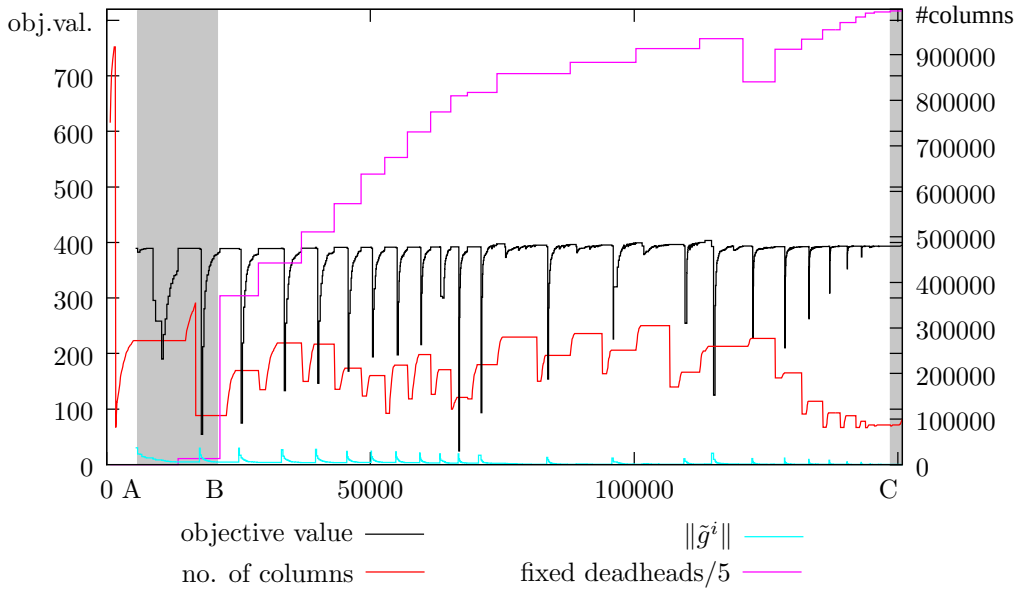


Figure 7.1: IS-OPT Runtime Chart.

the DSP objective value should be strictly decreasing, while the number of columns should grow. However, this is not the case for two reasons: in the initial phase only rough lower bounds for the restricted DSP, which are not completely accurate, are calculated. Second, columns with large reduced cost are deleted if the total number of columns exceeded 900,000 to prevent memory shortages.

Between points A and B, a series of iterations of the inexact PBM of Section 4.6 is performed, which decreases the norm of the aggregated subgradient $\tilde{g}^i$ and increases the ISP-value. That means, in this phase we compute an approximation of a primal solution and an approximated lower bound of the LP-relaxation of ISP.

Between points B and C, column generation phases alternate with PBM-steps. If no columns are found which improve the ISP-value IS-OPT uses the primal information collected by the PBM to fix or unfix deadheads until the vehicle scheduling part of the problem is completely decided. This is done by the rapid branching heuristic of Chapter 6. Only the branching rule is simplified: We do not perturb the objective value to find variables to fix but simply fix at each node of the search tree a set of variables with the largest primal values.

Between C and the end of the diagram the duty scheduling component of

the algorithm is concluded by computing a compatible duty schedule again by the rapid branching heuristic.

The peaks of the norm statistic mark serious steps of the PBM. The peaks are caused by the shift of the stability center in combination with the possible inclusion of additional columns in the column set I^i. In fact, the new stability center may lie in a region where the model $\hat{f}^i$ of the combined ISP-function (see Section 4.6.4) of the previous PBM-iteration i is less accurate; therefore, new columns in I^i change the function f^i, what worsens the model.

Figure 7.1 shows that only once the number of fixed deadheads decreases. In fact, in our computational tests the algorithm rarely ever had to reverse a fixing decision for a deadhead. In all our instances, the ISP objective value is very stable with respect to "careful" fixings of deadheads. Also, the gap between our estimated lower bound, i.e., the objective value prior to the first fixing, and the final objective value was never larger than 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 behavior is therefore only an indicator for the quality of the final solution found by IS-OPT.

7.2 Computation Times per Phase

The total running time of IS-OPT on our instances is, on average, distributed as follows:

- 50–70% for column generation in the duty scheduling subproblem,
- 10–30% for solving the LPs in the duty scheduling subproblem,
- 2–5% for solving the vehicle scheduling subproblems,
- 0.1–1% for solving quadratic problems in the PBM to calculate lower bounds on (ISP), for preprocessing, bookkeeping, etc.

One can see that most of the computation time, i.e., 50–70%, is needed to generate duties. This has many reasons:

- We have to perform a large number of column generation iterations (up to 300 for large problems).

- We have to generate duties for each combination of duty type and depot separately. It is not uncommon to have 30 different relevant combinations, which translate into 30 pricing problems per iteration.
- In some scenarios lots of possible relief points produce very small tasks. In combination with long duty durations (up to 14 hours), this leads to duties that consist of up to 50 tasks. It can be observed that the the running time to solve the pricing problem depends on the maximum number of tasks per duty.

Simplifying duty construction rules and reducing the number of relief points can alleviate these problems but leads in general to schedules with worse objective value.

7.3 Algorithms

We compare our integrated scheduling method IS-OPT 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 (Löbel [1997b]), then fixes the deadheads chosen by the vehicle schedule, and finally solves the resulting duty scheduling problem in a second step using the duty scheduling algorithm implemented in IS-OPT. The second method $d+v$ uses the combined approach of Section 2.2.2. A simplified integrated scheduling problem is set up that identifies drivers and vehicles, i.e., vehicle changes outside of the depot are forbidden. The combined vehicle and duty scheduling problem is solved using the duty scheduling algorithm in IS-OPT. The resulting driver-vehicle-rotations are concatenated into daily blocks using the vehicle scheduling algorithm of IS-OPT in a second step.

7.4 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 mostly urban. Each scenario has 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

	Sunday	workday
vehicle types	1	3
timetabled trips	794	1,414
tasks on tt	1,248	3,666
deadhead trips	47,523	57,646
duty types	3	4
break rules	4	4

Table 7.1: Statistics on the RVB Instances.

single garage the drivers can change buses 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 7.1 reports further statistics on the number of timetabled trips, tasks, and deadhead trips. The Sunday scenario is medium-sized, while the workday scenario is, one of the largest and most complex instance that has been attacked with integrated scheduling techniques.

Table 7.2 gives computational results for the Sunday scenario. The column 'reference' lists statistics for the solution that RVB planners had generated by hand. The next four columns give the results of two sequential v+d-optimizations and two integrated IS-OPT-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-OPT 2", emphasis was placed on the minimization of the number of duties, while runs "v+d 1" and "IS-OPT 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. They also produce quite reasonable results in terms of duty scheduling. "v+d 1" suffers from a slight increase in duties and paid time. Method "v+d 2" even yields substantial savings in duties; the price for this reduction, however, is an increase in average paid time. Even better are the results of the

	reference	*v+d* 1	*v+d* 2	IS-OPT 1	IS-OPT 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 7.2: Results for the RVB Sunday Scenario.

integrated optimizations. "IS-OPT 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-OPT 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-OPT 2" revived a discussion at the RVB whether drivers prefer to have fewer, but longer duties on weekends or whether they want to stay with more, but short duties.

Table 7.3 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 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 a single statistic. Doing it nevertheless, we see that both optimization approaches clearly improve the reference solution substantially. The outcome is close. In fact, $v+d$ has less paid time than IS-OPT; in the end, however, IS-OPT is better in terms of the composite objective function.

7.5 RKH Instances

The Regionalverkehrsbetrieb Kurhessen (RKH) is a regional carrier in the middle of Germany. They provided data for the subnetworks of Marburg

	reference	*v+d*	IS-OPT
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 7.3: Results for he RVB Workday Scenario.

and Fulda which is close to production data; 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 7.2 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 7.4 gives further statistics on the RKH instances. They are similar 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 continuous duties. Each duty type can have 1×30, 2×20, or 3×15 minutes block breaks or 1/6-quotient breaks.

Table 7.5 reports the results of our optimizations. We do not report results 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

	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 7.4: RKH Instances for the Cities of Marburg and Fulda

	Marburg		Fulda	
	d+v	IS-OPT	*d+v*	IS-OPT
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 7.5: Solutions on Marburg and Fulda

parentheses). These deficiencies are resolved in the IS-OPT-solutions, which also look better in terms of numbers of vehicles.

7.6 Subcontractor Planning for an Regional Carrier

This section describes the preliminary results of an optimization project with a regional public transit carrier. The public transit company, in the following called *contractor*, does not operate all of the traffic by its own vehicles and drivers but also concludes contracts to *subcontractors*. The contractor has five depots where vehicle rotations and duties begin and end. The vehicles of the subcontractors are stationed at another two depots. The available personnel and vehicles of the contractor should be used to cover as much of the trips as possible. Here not only the total numbers, but also ranges for the quantities of drivers and buses for each depot are given. The average

	Mo-Fr
depots	7
vehicle types	2
timetabled trips	3,698
tasks on tt	6,583 (3,966)
deadhead trips	121,217
duty types	6 + 1

Table 7.6: Statistics on the instance with subcontractors.

paid time per driver should not exceed 7h 40min.

The trips that are not covered by the contractor are given to two subcontractors, which are paid roughly proportional to the driving time in their rotations. The subcontractors schedule their drivers by themselves, because they may have additional tasks from other clients which are not known by the contractor. Thus, the main objective of this scenario is to minimize the driving time of trips that are operated by the subcontractors.

The data used in this section is not completely correct, because some rules are not correctly represented and some deadheads and walks may have wrong attributes or are completely lacking. However, the provisional results were convincing enough that IS-OPT has now been licensed by the respective company and is used to plan the duty and vehicle schedules.

Table 7.6 shows the characteristics of the scenario. The row "tasks on tt" shows the number of tasks on timetabled trips, these are the tasks which have to be considered in the duty-scheduling subproblem. The number in braces is the number of tasks driven by the contractor. This is also the number of tasks for which IS-OPT finally has to find a duty schedule. The six duty types stem from two sets of short, continuous, and split duties, depending on the starting depot. Additionally we have one "free" duty type without restrictions besides the compatibility of their tasks. This duty type can only be used by the subcontractors and models that we do not schedule their duties. Therefore the rotations of the contractors can also be seen as "free" duties. These free duties are not shown in the following statistics.

The correct distribution of the drivers to the depots is guaranteed by 25 base-constraints of type (ISP)(ii), the distribution of the vehicles is given by 4 capacity-constraints of type (VSP)(iv).

Table 7.6 compares the results of IS-OPT with the combined approach

	IS-OPT		*d+v*
	contractor	subcontractors	
driving time	890:29	463:22	895:47
paid time	1,023:16	–	1,032:43
number of duties	155	104	160
number of vehicles	105	104	–
average duty duration	7:10	–	6:45
computation time	49:55		3:28

Table 7.7: Solutions on subcontractor planning

d+v (see Section 7.3) on the subcontractor scenario. The times in this table are in hour:minutes. These results were obtained by a computer with a Intel Core 2 CPU with 2.93GHz and 4 GB main memory using SuSE Linux 10.2. This computer is about twice as fast as the computer used for the other computations in this chapter.

The columns "IS-OPT" shows the result of the integrated approach. The first column contains the values for the duties and rotations of the contractor, the second column shows the values of the rotations of the subcontractors.

Unfortunately we have no solution that we can compare with the one of IS-OPT. However, the duty-scheduling solution of the combined approach *d+v* is similar in its quality, it covers about 0.5% more of the driving time at the cost of more duties. These solutions are only comparable with reservations, because *d+v* completely ignores the capacity constraints for vehicles, and thus does not give a feasible vehicle schedule. It also ignores cost for subcontractors. But we think that the similar qualities of the duty schedules are an indicator for the quality of the solution of IS-OPT.

This scenario is the largest scenario known to us that was solved by an integrated approach.

7.7 ECOPT Instances

Finally, we compare IS-OPT with the approach of Huisman et al. [2003b] on the randomly generated benchmark data proposed in their article. This data consists of four sets of instances. On the one hand, these sets differ in the length of the trips. Variant A has about 25% shorter trips on average than

trips	080	100	160	200	320	400
vehicles	9.2	11.2	14.9	18.4	26.8	33.2
duties	20.9	24.4	33.0	40.0	57.3	69.3
total	30.1	35.6	47.9	58.4	84.1	102.5
reference	29.8	35.6	48.3	59.1	86.8	106.1
time	00:06	00:07	00:19	00:25	01:04	01:59

Table 7.8: Results for ECOPT-Instances with 2 Depots Variant A

trips	080	100	160	200	320	400
vehicles	9.2	11.1	14.8	18.6	27.2	33.2
duties	20.3	24.0	32.4	39.4	55.4	68.5
total	29.5	35.1	47.2	58.0	82.6	101.7
reference	29.6	36.2	49.5	60.4	—	—
time	00:07	00:10	00:22	00:33	01:34	02:57

Table 7.9: Results for ECOPT-Instances with 4 Depots Variant A

Variant B. On the other hand, the number of depots varies. Every instance was calculated using four different depots and then again using only two of these while the other data remains constant. Each set contains 10 instances of 80, 100, 160, 200, 320, and 400 trips. In Huisman et al. [2003b] a detailed description of the generation of these problems is given.

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 Huisman et al. [2003b] that in this situation one can solve the duty generation subproblem in polynomial time, i.e., exact column generation is possible.

Tables 7.8 and 7.9 report average solution values for each of the 10 instances of each problem class for the problem variant A; similar results for variant B can be found in Tables 7.10 and 7.11. 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 Huisman et al. [2003b]; for the problems with 4 depots and 320 and 400 trips, no reference is given due to excessive computation time.

trips	080	100	160	200	320	400
vehicles	11.4	13.8	19.3	24.5	36.1	44.6
duties	24.8	28.9	42.1	49.5	74.7	90.2
total	36.2	42.7	61.4	74.0	110.8	134.8
reference	36.0	42.9	61.6	75.8	113.6	139.2
time	00:05	00:06	00:17	00:24	00:58	01:39

Table 7.10: Results for ECOPT-Instances with 2 Depots Variant B

trips	080	100	160	200	320	400
vehicles	11.3	13.9	19.3	24.6	36.2	45.3
duties	24.4	29.2	41.0	49.7	72.6	88.8
total	35.7	43.1	60.3	74.3	108.8	134.1
reference	36.1	43.8	62.2	76.5	–	–
time	00:08	00:11	00:22	00:35	01:33	02:32

Table 7.11: Results for ECOPT-Instances with 4 Depots Variant B

Algorithm IS-OPT produces in most cases better results than that in Huisman et al. [2003b], only for the smallest instance with two depots it is slightly worse. IS-OPT produces comparably better results with increasing problem size and complexity; it can also solve the largest problem instances in reasonable time. IS-OPT is, contrary to the approach of Huisman et al. [2003b], able to utilize the additional depots to save some more drivers and vehicles. Another observation is that the runtime of IS-OPT increases roughly quadratic with the number of trips of the ECOPT-instances, whereas the runtime of the algorithm of Huisman et al. [2003b] increases more steeply.

7.8 Conclusion

We have shown that IS-OPT is able to solve real world large scale instances in reasonable time. Further, it helps to improve the vehicle and duty schedules of public transit companies in comparison with sequential solutions and also with manually planned solutions. For scenarios with subcontractors the traditional sequential planning is not able to produce a feasible solution which fulfills all base constraints and vehicle capacity constraints. IS-OPT, however, finds good solutions.

We have shown not only here, but also in other projects, that it is possible to improve the vehicle and duty schedules by up to 5% with respect to the objective function defined by the planners while all their requirements are met. In 2006, IS-OPT was deployed in several German public transport companies as a standard tool to plan vehicle and duty schedules.

On the artificial instances published by Huisman et al. [2003b] IS-OPT is able to compute better results than the algorithm proposed there. Especially on the larger instances the improvements are significant.

However, further work on IS-OPT should be invested to accelerate the column generation, perhaps by heuristical approaches or by identifying duty types or parts of the network, which can not be used by duties with negative reduced cost.

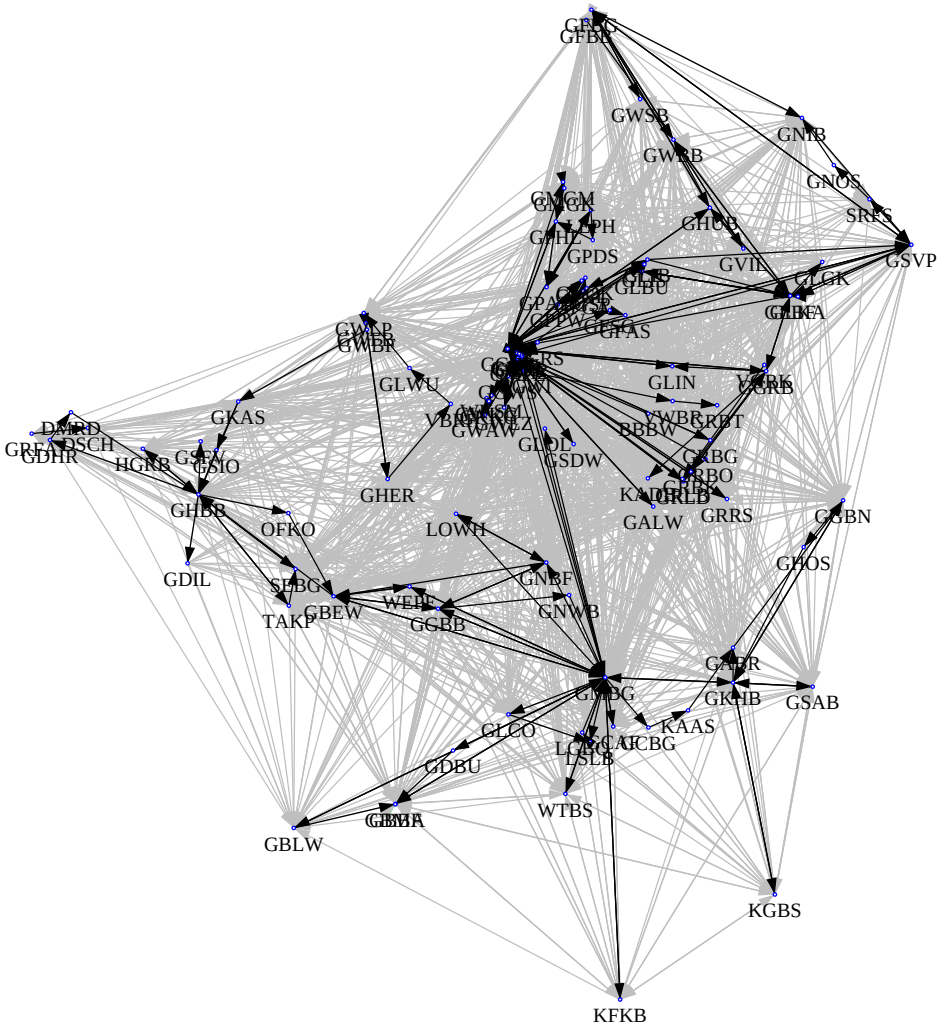


Figure 7.2: The graph of scenario *Fulda*

Glossary

block the tour of a bus starting and ending in a depot. A block is also called a **rotation**. p. 19

break a possibility for a driver in a duty to get a pause. On a break there is no work to do, nevertheless it counts as duty time. p. 115

course a course is a sequence of blocks which can be performed by a single vehicle. Also known in the literature as a vehicle schedule or a vehicle duty. p. 19

DSP duty scheduling problem. p. 27

duty a sequence of tasks, connected by links, to which personnel can be assigned to, also known as a *run*. p. 25

deadhead trip every trip without passengers. Also shortly denoted by deadhead. We also call the waiting time of a vehicle between two timetabled trips even without moving the vehicle a deadhead. p. 18

GMCF generalized minimum cost flow problem, minimum cost flow problem with additional constraints. p. 4

GPCP generalized path covering problem, path covering problem with additional constraints. p. 6

ISP integrated vehicle and duty scheduling problem. p. 43

link if two tasks can be performed subsequently a link between the tasks is established. p. 26

long arc an arc in the VSP-graph (Section 1.9) corresponding to a pull-out-pull-in-trip. p. 21

part of work a part of a duty without interruptions. A duty is in general partitioned in one (continuous duties), two (split duties), or sometimes even three parts. p. 112

piece of work an (inclusion-)maximal sequence of tasks on the same bus. p. 114

PRICE pricing problem in a column generation context. p. 121

pull-in trip a trip from the depot to a time-tabled trip. p. 19

pull-out trip a trip to the depot from a time-tabled trip. p. 19

pull-out-pull-in trip the combination of a pull-out trip, a standing time at a depot, and a subsequent pull-in trip, of a single vehicle. p. 21

RCSP resource constrained shortest path problem. p. 126

relief point is a location where drivers can be changed. p. 26

rotation the same as a block. p. 19

run the same as a duty. p. 25

split duty a duty with an interruption, sometimes also called *broken run*. p. 112

SCP set covering problem. p. 7

SPP set partitioning problem. p. 7

SSP set packing problem. p. 7

supplementary task an optional activity that is eventually needed to make a duty feasible under certain circumstances.,e.g., checking the bus and the cash box before leaving a bus. p. 25

task an activity that has to be performed uninterrupted by a driver. Also known as *d-trip*. p. 24

timetabled trip a trip carrying passengers and appearing in the timetable. p. 18

trip activity of a single bus. We differentiate between timetabled trips and deadhead trips. p. 18

VSP vehicle scheduling problem. p. 21

Symbols

$\boldsymbol{\delta^{\text{in}}(v)}$ in-degree of a node v,

$\boldsymbol{\delta^{\text{out}}(v)}$ out-degree of a node v,

$\boldsymbol{A}$ duty scheduling matrix, section 2.2.3,

$\boldsymbol{A_i}$ row i of a matrix A,

$\boldsymbol{A_{\cdot j}}$ column j of a matrix A,

$\boldsymbol{\langle \cdot, \cdot \rangle}$ scalar product,

$\boldsymbol{\mathcal{B}}$ base constraints.

$\boldsymbol{a}$ an arc in a directed graph, sometimes also a right hand side of a LP,

$\boldsymbol{b}$ right hand side of a LP,

$\boldsymbol{c}$ cost vector of a LP,

$\boldsymbol{\gamma}$ cost vector of slacks of (DSP)

$\boldsymbol{\zeta}$ cost vector of arcs of (RCSP) and (PRICE),

$\boldsymbol{d}$ cost vector of deadhead trips,

$\boldsymbol{D_v}$ set of duties covering task v.

$\boldsymbol{e_i}$ vector with an entry of one at position i and zeroes else,

$\boldsymbol{f}$ a function,

$\boldsymbol{\bar{f}}$ a linearization of f,

$\boldsymbol{\hat{f}}$ a cutting plane model of f,

$\tilde{f}$ a quadratic model of f,

g a subgradient,

$\tilde{g}$ an aggregated subgradient,

κ^u, κ^ℓ costs of slacks in (RCSP) and (PRICE),

Q resource matrix of arcs in (RCSP) and (PRICE),

R resource matrix of duties in (DSP),

r a resource,

s source of a network.

t sink of a network,

u, v, w nodes of a network,

x, y, z variables in an LP or ILP,

$x(Y)$ sum over the elements in the set Y of a vector x,

x_I vector that consists of the elements of x that are elements of the index set I,

N network adjacency matrix,

α multipliers of a convex combination of vectors,

λ, μ, ν Lagrangean multipliers or dual variables,

V_{VSP} set of timetabled trips in the VSP,

A_{VSP} set of deadhead trips in the VSP,

V_{DSP} set of tasks for the DSP

V_{DSP}^T set of tasks corresponding to timetabled trips (V_{VSP}) of the VSP for the DSP

V_{DSP}^D set of tasks corresponding to deadheads (A_{VSP}) of the VSP for the DSP

A_{DSP} set of possible concatenations of tasks in $V_{\mathsf{DSP}} \cup \{s, t\}$

$V_{\mathsf{DSP}}(t)$ set of tasks on the trip t.

Bibliography

Achterberg, Koch & Martin (2005). Branching rules revisited. *Operations Research Letters 33, 42 – 54.*

Achterberg, Koch & Martin (2006). MIPLIB 2003. *Operations Research Letters 34(4), 1–12.* See `http://miplib.zib.de`.

Ahuja, Magnanti & Orlin (1993). *Network flows.* Prentice Hall, Englewood Cliffs.

Andersson, Housos, Kohl & Wedelin (1998). Crew Pairing Optimization. In Yu [1997], pp. 228–258.

Balas, Ceria, Dawande, Margot & Pataki (2001). Octane: A New Heuristic for Pure 0-1 Programs. *Operations Research 49(2), 207–225.*

Balas & Ho (1980). Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization: A Computational Study. *Mathematical Programming 12, 37–60.*

Ball, Bodin & Dial (1983). A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles. *Transportation Science 17, 4–31.*

Barahona & Anbil (1998). The Volume Algorithm: producing primal solutions with a subgradient method. Technical report, IBM Research Division, T.J. Watson Research Center.

Barnhart, Johnson, Nemhauser, Savelsbergh & Vance (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research 46(3), 316–329.*

Beasley (1987). An Algorithm for the Set Covering Problem. *European Journal of Operational Research 19, 379–394.*

—— (1990). OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society 41(11), 1069–1072.*

Beasley & Christofides (1989). An Algorithm for the Resource Constrained Shortest Path Problem. *Networks 19, 379–394.*

Blais & Rousseau (1982). HASTUS: A model for the economic evaluation of drivers' collective agreements in transit companies. *INFOR 20(3), 3 – 15.*

Bönisch (2006). Sensitivitätsanalyse in der Fahrzeugumlaufplanung. Master's thesis, TU Berlin.

Borndörfer (1998). *Aspects of Set Packing, Partitioning, and Covering.* PhD thesis, TU Berlin.

—— (2006). A tutorial on railway optimization. Talk held at ATMOS 2006 and personal communication.

Borndörfer, Grötschel & Löbel (2003). Duty Scheduling in Public Transit. In W. Jäger & H.-J. Krebs (Eds.), *MATHEMATICS — Key Technology for the Future* pp. 653–674. Berlin: Springer Verlag.

Borndörfer, Grötschel & Pfetsch (2004). Models for Line Planning in Public Transport. Technical report, Zuse Institute Berlin.

Borndörfer, Löbel & Weider (2002). Integrierte Umlauf- und Dienstplanung im Öffentlichen Nahverkehr. In Boltze (Ed.), *Heureka '02*, pp. 77–98. VDV, Forschungsgesellschaft für Straßen- und Verkehrswesen, Köln.

Borndörfer, Löbel & Weider (2004). A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit. Technical report, Zuse Institute Berlin. talk held at CASPT 2004, proceedings not published yet.

Borndörfer, Grötschel & Pfetsch (2006). Public Transport to the fORe! *OR/MS Today 33(2), 30–40.*

Bouma & Oltrogge (1994). Linienplanung und Simulation für öffentliche Verkehrswege in Praxis und Theorie. *Eisenbahntechnische Rundschau 43(6), 369–378.*

Bussieck, Kreuzer & Zimmermann (1997). Optimal lines for railway systems. *Eur. J. Oper. Res. 96(1), 54–63.*

Bussieck, Lindner & Lübbecke (2003). A Fast Algorithm for Near Cost Optimal Line Plans. Technical report, TU Berlin.

Bussieck, Winter & Zimmermann (1997). Discrete optimization in public rail transport. *Mathematical Programming 1–3(79B), 415–444.*

Byun (2001). Lower Bounds for Large-Scale Set Partitioning Problems. Technical report, Zuse Institute Berlin.

Caprara, Fischetti & Guida (1997). Solution of Large-Scale Railway Crew Planning Problems: the Italian Experience. In Wilson [1997], pp. 1–18.

Caprara, Fischetti & Toth (1996). A Heuristic Algorithm for the Set Covering Problem. In *IPCO.*

Caprara, Fischetti, Toth & Vigo (1998). Modeling and Solving the Crew Rostering Problem. *Operations Research 46, 820–830.*

Caprara, Monaci & Toth (2001). A Global Method for Crew Planning in Railway Applications. In Voß & Daduna [2001], pp. 17–36. Springer Verlag.

Carresi, Girardi & Nonato (1995). Network models, Lagrangean relaxation and subgradients bundle approach in crew scheduling problems. In Daduna et al. [1995], pp. 188–212.

Ceder & Tal (1997). Timetable Synchronization for Buses. In Wilson [1997], pp. 245–258.

Ceria, Nobili & Sassano (1998). A Lagrangian-based Heuristic for Large-scale Set Covering Problems. *Mathematical Programming 81, 215–228.*

Chien, Qin & Liu (2003). Optimal Bus Stop Locations for Improving Transit Accessibility. Preprint cd-rom, Transportation Research Board.

Chvátal (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research 4(3), 233–235.*

Claessens, van Dijk & Zwaneveld (1998). Cost optimal allocation of rail passenger lines. *European Journal of Operational Research 110, 474–4889.*

Daduna (2001). Impacts of Deregulation on Planning Processes and Information Management Design in Public Transit. In Voß & Daduna [2001], pp. 429–441.

Daduna, Branco & Paixão (Eds.) (1995). *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Daduna & Wren (Eds.) (1988). *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Dantzig & Wolfe (1960). Decomposition principle for linear programs. *Operations Research 8, 101–111.*

Darby-Dowman, J. K. Jachnik & Mitra (1988). Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience. In Daduna & Wren [1988], pp. 226–239.

de Groot & Huisman (2004). Vehicle and Crew Scheduling: Solving Large Real-World Instances with an Integrated Approach. Technical report, Econometric Institute, Erasmus University Rotterdam.

Desaulniers, Desrosiers & Solomon (Eds.) (2005). *Column Generation.* GERAD 25th Anniversary Series. Springer.

Desaulniers, Lavigne & Soumis (1998). Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research 34(3), 479–494.*

Desrochers (1986). A New Algorithm for the Shortest Path Problem with Resource Constraints. Technical Report 421A, Centre de Recherche sur les Transports, University of Montréal.

Desrochers, Gilbert, Sauvé & Soumis (1992). Crew-Opt: Subproblem Modeling in a Column Generation Approach to Urban Crew Scheduling. In Desrochers & Rousseau [1992].

Desrochers & Rousseau (Eds.) (1992). *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Desrochers & Soumis (1988). A Generalized Permanent Labeling Algorithm for the Shortest Path Problem with Time Windows. *INFOR 26, 191–212.*

Desrochers & Soumis (1989). A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science 23(1), 1–13.*

Desrosiers & Lübbecke (2005). A primer in column generation. In GERAD [2005], pp. 1–32.

Desrosiers, Soumis & Desrochers (1982). Routes sur un réseau espace-temps. Technical Report 236, Centre de recherche sur les transports, Université de Montréal.

Dorn (1960). Duality in Quadratic Programming. *Quarterly of Applied Mathematics 18(2), 155–162.*

Dumitrescu (2002). *Constrained Path and Cycle Problems.* PhD thesis, Department of Mathematics and Statistics, The University of Melbourne.

Emden-Weinert, Kotas & Speer (2001). DISSY – A Driver Rostering System for Public Transport. Technical report, HU Berlin. presented at the CASPT 2000.

Ernst, Jiang, Krishnamoorthy & Nott (2001). Rail Crew Scheduling and Rostering Optimization Algorithms. In Voß & Daduna [2001], pp. 53–72.

Ernst, Jiang, Krishnamoorthy & Sier (2004). An Annotated Bibliography of Personnel Scheduling and Rostering. *Annals of Operations Research 127, 21–144.*

Falkner & Ryan (1992). Express: Set partitioning for bus crew scheduling in Christchurch. In Desrochers & Rousseau [1992], pp. 359–378.

Fischer, Gruber, Rendl & Sotirov (2003). The Bundle Method in Combinatorial Optimization. Technical report, University of Klagenfurt, Austria.

Fischetti, Lodi, Martello & Toth (2001). A Polyhedral Approach to Simplified Crew Scheduling and Vehicle Scheduling Problems. *Management Science 47, 833–850.*

Fisher & Kedia (1990). Optimal Solution of Set Covering/Partitioning Problems Using Dual Heuristics. *Management Science 39, 674–688.*

Frangioni (2002). Generalized Bundle Methods. *SIAM Journal on Optimization 13(1), 117–156.*

Freling (1997). *Models and Techniques for Integrating Vehicle and Crew Scheduling.* PhD thesis, Erasmus University Rotterdam, Amsterdam.

Freling, Huisman & Wagelmans (2000). Models and Algorithms for Integration of Vehicle and Crew Scheduling. Technical Report EI2000-10/A, Econometric Institute, Erasmus University Rotterdam.

Freling, Huisman & Wagelmans (2001). Applying an Integral Approach to Vehicle and Crew Scheduling in Practice. In Voß & Daduna [2001], pp. 73–90.

Freling, Huisman & Wagelmans (2003). Models and Algorithms for Integration of Vehicle and Crew Scheduling. *Journal of Scheduling 6, 63–85.*

Friberg & Haase (1997). An Exact Algorithm for the Vehicle and Crew Scheduling Problem. In Wilson [1997], pp. 63–80.

Gaffi & Nonato (1997). An Integrated Approach to Ex-Urban Crew and Vehicle Scheduling. In Wilson [1997], pp. 103–128.

Garey & Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman and Company.

Gintner, Kliewer & Suhl (2004). Line Change Considerations Within a Time-Space Network Based Multi-Depot Bus Scheduling Model. DSOR Arbeitspapiere/DSOR working papers, Universität Paderborn.

Goossens, van Hoesel & Kroon (2004). A Branch-and-Cut Approach for Solving Railway Line-Planning Problems. *Transportation Science 38(3), 379–393.*

Grötschel, Lovász & Schrijver (1993). *Geometric Algorithms and Combinatorial Optimization* (Second corrected edition ed.)., volume 2 of *Algorithms and Combinatorics.* Springer.

Haase, Desaulniers & Desrosiers (2001). Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. *Transportation Science 35(3), 286–303.*

Hadjar, Marcotte & Soumis (2001). A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem. Technical report, Les Cahiers du GERAD.

Handler & Zang (1980). A Dual Algorithm for the Constrained Shortest Path Problem. *Networks 10, 293–310.*

Hanisch (1990). Die Regionalverkehr Köln GmbH und HASTUS. `http://www.giro.ca/Deutsch/Publications/publications.htm`.

Held & Karp (1970). The traveling-salesman problem and minimum spanning trees. *Operations Research 18, 1138–1162.*

—— (1971). The traveling-salesman problem and minimum spanning trees: part II. *Mathematical Programming 1, 6–25.*

Helmberg (2000). Semidefinite Programming for Combinatorial Optimization. Technical report, Zuse Institute Berlin. also habilitation thesis.

Helmberg & Kiwiel (2002). A Spectral Bundle Method with Bounds. *Mathematical Programming 2(93), 173–194.*

Hintermüller (2001). A Proximal Bundle Method Based on Approximate Subgradients. *Computational Optimization and Applications 3(20), 245–266.*

Hiriart-Urruty & Lemaréchal (1993a). *Convex Analysis and Minimization Algorithms I*, volume 305 of *A Series of Comprehensive Studies in Mathematics*. Springer-Verlag.

—— (1993b). *Convex Analysis and Minimization Algorithms II*, volume 306 of *A Series of Comprehensive Studies in Mathematics*. Springer-Verlag.

Hochbaum (1997). Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In D. S. Hochbaum (Ed.), *Approximation Algorithms for NP-hard Problems* pp. 94–143. PWS Publishing Company.

Hooghiemstra, Kroon, Odijk, S. & Zwaneveld (1999). Decision Support Systems Support the Search for Win-Win Solutions in Railway Network Design. *Interfaces 29(2), 15–32.*

Huisman (2004). *Integrated and Dynamic Vehicle and Crew Scheduling.* PhD thesis, Erasmus University Rotterdam.

Huisman, Freling & Wagelmans (2003a). Multiple-Depot Integrated Vehicle and Crew Scheduling. Technical report, Econometric Institute, Erasmus University Rotterdam.

—— (2003b). Multiple-Depot Integrated Vehicle and Crew Scheduling. Technical report, Econometric Institute, Erasmus University Rotterdam. Report EI2003-02.

Johnson (1989). Modeling and strong linear programs for mixed integer programming. In Wallace (Ed.), *Algorithms and model formulations in mathematical programming*, 51, pp. 1–41. NATO ASI.

Johnson, Nemheuser & Savelsbergh (2000). Progress in Linear-Programming based Algorithms for Integer Programming: An exposition. *INFORMS Journal on Computing 12, 2–23*.

Kim & Barnhart (1997). Transportation Service Network Design: Models and Algorithms. In Wilson [1997], pp. 259–283.

Kiwiel (1990). Proximal bundle methods. *Mathematical Programming 46(123), 105–122*.

—— (1994). A Cholesky Dual Method for proximal piecewise linear programming. *Numerische Mathematik 68, 325–340*.

—— (1995). Approximation in Proximal Bundle Methods and Decomposition of Convex Programs. *Journal of Optimization Theory and applications 84(3), 529–548*.

—— (2006). A proximal bundle method with approximate subgradient linearizations. *SIAM Journal on Optimization 16(4), 1007–1023*.

Kliewer & Mellouli (2002). Umlaufplanung im öffentlichen Verkehr mit mehreren Depots und Fahrzeugtypen. In Boltze (Ed.), *Heureka '02*, pp. 63–76. VDV, Forschungsgesellschaft für Straßen- und Verkehrswesen, Köln.

Kliewer, Mellouli & Suhl (2004). A time-space network based exact optimization model for multi-depot bus scheduling. DSOR Arbeitspapiere/DSOR working papers, Universität Paderborn.

Kohl & Karisch (2004). Airline Crew Rostering: Problem Types, Modeling, and Optimization. *Annals of Operations Research 127, 223–257*.

Kroon & Fischetti (2000). Scheduling Train Drivers and Guards: The Dutch "Noord-Oost" Case. In *HICSS*.

Kroon & Peeters (2003). A Variable Trip Time Model For Cyclic Railway Timetabling. *Transportation Science 37(2), 198–212*.

Lamatsch (1992). An Approach to Vehicle Scheduling with Depot Capacity Constraints. In Desrochers & Rousseau [1992], pp. 181–195.

Lemaréchal (2001). Lagrangian Relaxation. In Jünger & Naddef (Eds.), *Computational Combinatorial Optimization*, volume 2241 of *Lecture Notes in Computer Science*, pp. 112–156. Springer.

Leuthardt (1998). Kostenstrukturen von Stadt-, Oberland- und Reisebussen. *Der Nahverkehr (6), 19–23.*

Liebchen & Möhring (2002). A Case Study in Periodic Timetabling. *Electronic Notes in Theoretical Computer Science 66(6), 14.*

—— (2004). The Modeling Power of the Periodic Event Scheduling Problem. Technical report, TU Berlin. presented at the CASPT 2004.

Linderoth & Ralphs (2004). Noncommercial Software for Mixed-Integer Linear Programming. Optimization Online.

Lindner (2000). *Train schedule optimization in public rail transport.* PhD thesis, TU Braunschweig.

Löbel (1996). Solving Large-Scale Real-World Minimum-Cost Flow Problems by a Network Simplex Method. Technical report, Zuse Institute Berlin.

—— (1997a). *Optimal Vehicle Scheduling in Public Transit.* PhD thesis, TU Berlin.

—— (1997b). Solving Large-Scale Multi-Depot Vehicle Scheduling Problems. In Wilson [1997], pp. 195–222.

Lougee-Heimer (2003). The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development 47(1), 57–66.*

Magnanti & Wong (1984). Network Design and Transportation Planning: Models and Algorithms. *Transportation Science 18(1), 1–55.*

Martin (2001). Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions, chapter General Mixed Integer Programming: Computational Issues for Branch-and-Cut Algorithms, pp. 1–25. Springer.

Mehlhorn & Ziegelmann (2000). Resource Constrained Shortest Paths. In *Proc. 8th European Symposium on Algorithms*, pp. 326–337. Springer.

Nediak & Eckstein (2001). Pivot, Cut, and Dive: A Heuristic for 0-1 Mixed Integer Programming. Technical Report RRR 53-2001, Rutgers University, 640 Bartholomew Road, Piscataway, NJ, 08854 USA.

Nering & Tucker (1993). *Linear Programs and Related Problems.* Computer Sciece and Scientific Computing. Academic Press, London.

Odoni, Rousseau & Wilson (1994). Handbooks in Operations Research and Management Science, volume 6, chapter 5, pp. 107–150. Elsevier.

Padberg (2001). Almost Perfect Matrices and Graphs. *Math. Oper. Res. 26(1), 1–18.*

Papadimitriou & Steiglitz (1982). *Combinatorial Optimization: Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice-Hall.

Patrikalakis & Xerocostas (1992). A new decomposition scheme of the urban public transport scheduling problem. In Desrochers & Rousseau [1992], pp. 407–425.

Ralphs, Ladányi & Saltzman (2003). Parallel branch, cut, and price for large-scale discrete optimization. *Mathematical Programming 98(1–3), 253–280.*

Rang (2006). *Lenk- und Ruhezeiten im Straßenverkehr.* Verlag Heinrich Vogel.

Resch, Neth & Budäus (2006). ÖPNV zwischen Ausschreibungswettbewerb und Direktvergabe. Technical report, Hans-Böckler-Stiftung. in German.

Ribeiro & Minoux (1986). Solving Hard Constrained Shortest Path Problems by Lagrangean Relaxation and Branch-and-Bound Algorithms. *Mathematics of Operations Research 53, 303–316.*

Ryan & Foster (1981). An integer programming approach to scheduling. In A. Wren (Ed.), *Computer Scheduling of Public Transport.* North-Holland Publishing Company.

Schlechte (2003). Das Resource-Constrained-Shortest-Path-Problem und seine Anwendung in der ÖPNV-Dienstplanung. Master's thesis, TU Berlin.

Schöbel & Scholl (2006). Line Planning with Minimal Traveling Time. In Kroon & Möhring (Eds.), *5th Workshop on Algorithmic Methods and Models for Optimization of Railways.* Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.

Schrijver (1998). *Theory of Linear and Integer Programming.* Interscience series in discrete mathematics and optimization. Wiley.

Schrijver (2003). *Combinatorial Optimization.* Algorithms and Combinatorics. Springer.

Scott (2001). *An Inexact Bundle Method for Solving Large Structured Linear Matrix Inequalities*. PhD thesis, University of California.

Scott (1985). A Large Scale Linear Programming Approach to the Public Transport Scheduling and Costing Problem. In Rousseau (Ed.), *Computer Scheduling of Public Transport 2*. Elsevier.

Serafini & Ukovich (1989). A methematical model for periodic event scheduling problems. *SIAM Journal of Discrete Mathematics 4(2), 550–581.*

Thompson (2002). An integral simplex algorithm for solving combinatorial optimization problems. *Computational Optimization and Applications 22(3), 351–367.*

Tosini & Vercellis (1988). An interactive system for extra-urban vehicle and crew scheduling problems. In Daduna & Wren [1988], pp. 41–53.

van Krieken, Fleuren & Peeters (2004). A Lagrangean Relaxation Based Algorithm for Solving Set Partitioning Problems. Technical Report 2004-44, Tilburg University.

van Nes & Bovy (2000). The importance of objectives in urban transit network design. Preprint cd-rom, Transportation Research Board.

Villeneuve, Desrosiers, Lübbecke & Soumis (2003). On Compact Formulations for Integer Programs Solved by Column Generation. Technical Report No. 2003/25, TU Berlin.

Völker & Schütze (1995). Recent Developments of HOT II. In Daduna et al. [1995], pp. 334–348.

Voß & Daduna (Eds.) (2001). *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Warburton (1987). Approximation of Pareto Optima in Multiple-Objective Shortest Path Problems. *Op. Res. 35, 70–79.*

Wedelin (1995). An algorithm for a large scale 0-1 integer programming with application to airline crew scheduling. *Annals of Operations Research 57, 283–301.*

Wilson (Ed.) (1997). *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Yu (Ed.) (1997). *Operations Research in the Airline Industry.* Boston, MA: Kluwer Academic Publishers.