Susanne Scholl

Customer-Oriented Line Planning

Vom Fachbereich Mathematik
der Technischen Universität Kaiserslautern
genehmigte

Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)

Erstgutachterin: Prof. Dr. Anita Schöbel
Zweitgutachter: Prof. Dr. Leo G. Kroon

Datum der Disputation: 15. Juli 2005

D 386

Scholl, Susanne:
Customer-Oriented Line Planning / Susanne Scholl. –
Als Ms. gedr.. – Berlin : dissertation.de – Verlag im Internet GmbH, 2006
Zugl.: Kaiserslautern, Techn. Univ., Diss., 2005
ISBN 3-86624-084-8

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Es wird ausschließlich chlorfrei gebleichtes
Papier (TCF) nach DIN-ISO 9706 verwendet.
Printed in Germany.

dissertation.de - Verlag im Internet GmbH
Pestalozzistraße 9
10625 Berlin

URL: http://www.dissertation.de

Wir stolzen Menschenkinder
sind eitel, arme Sünder
und wissen gar nicht viel.
Wir spinnen Hirngespinnste
und wissen viele Künste
und kommen weiter von dem Ziel.

(Matthias Claudius, 1778)

Meiner lieben Großtante
Schwester Therese,
geborene Else Molitor
gewidmet.

Acknowledgements

I would like to take the opportunity to thank all the people who supported my work in various ways.

First of all, I thank Prof. Dr. Anita Schöbel and Prof. Dr. Horst Hamacher of the Technical University of Kaiserslautern, who gave me the opportunity to get deeper into the field of integer programming and traffic planning. For me it was a great experience to have all the freedom I wanted to decide about my scientific work and at the same time knowing that I get the support I need.

The thesis was prepared and written during my three years scholarship of the Graduate College *Mathematics and Practice* of the *German Research Foundation* (DFG) at the Technical University of Kaiserslautern. I thank my project partners Dr. Frank Wagner from Die Bahn (German Railway) and Dr. Norbert Ascheuer from Intranetz for the support and real-world data.

I thank all my colleagues in the department *Optimization* for the pleasant times there. In particular, I would like to mention Olena Gavriliouk and Mangalika Jayasundara. Furthermore Martin Pieper was a patient proofreader and a first friend at the Georg-August University of Göttingen, where I finished this work.

I am thankful to my grandaunts Hanna and Else Molitor for paving the way for women at German universities.

Besides all these, I am very grateful to Ludger, my brother Peter, and more than all, to my parents Dagmar and Hanspeter, and my husband Alexander for their love and imperturbable belief in me and the completion of this work.

Contents

Chapter 1

Introduction

The rail transportation industry is very rich in terms of problems that can be modeled and solved using mathematical optimization techniques. Public transportation planning is based on the anticipatory determination of vehicle runs from a start to an end point and the assignment of an operating resource and employees to these runs. In 2003, German Rail had 243700 employees that carried 1.7 billion travelers on a total of 70 billion kilometers. The German rail network consists of 35000 kilometers of rail track and 5665 stations ([Rai]). Facing these dimensions it is clear that is is not possible to plan such a complex system in just one step. Thus, the typical planning process is divided in three planning levels: the *strategical level* treating long-term decisions such as the estimation of passengers demand, the stop location problem or the line planning problem. These decision have a time horizon of 10 to 20 years. The second level is the *tactical level* where the time schedule and the duty roster is made. These problems have a time horizon of about one year. The last level is the *dispositive level*, where short-term decisions like delay management are covered. *Delay management* is the problem: which vehicle should wait if another vehicle has a delay. [Goo04] gives a nice introduction into the different planning problems for passenger railways.

This thesis deals with the line planning problem which is part of the strategic, the long-term level. It is concerned with the question of the routes the trains will run in a given public transportation network. Even though the real world data our approaches are developed for comes from rail transport, the ideas mentioned in this thesis can easily be adapted to the line planning problem for other transportation systems, e.g. buses.

A comprehensive discussion of the line planning problem including its modeling and solution applying mathematical programming methods, constitutes

the core of this thesis. Therefore we concentrate on structural properties of the problem. We show that it belongs to the class of hardest optimization problems and present numerous solution approaches. In opposite to other line planning models presented in literature so far, which aim to minimize the operational costs or to maximize the number of direct travelers, we minimize the travel times over all customers including penalties for the transfers needed while keeping the operational costs in mind. Penalties on transfers are important since customers associate inconvenience and the risk of a delay due to a missed connection with transfers. If a customer has to be at a destination in time and has a short transfer time on his way, he will either take one train earlier and so its total travel time will increase a lot or, in the worst case, he will not use public transportation at all. Psychologically, the annoyance about a missed connection is much higher than about a delay of a train the customer is sitting in. This is called the "'red light phenomenon"' because the (running) traveler just sees the red back-lights of the leaving train.
If we are able to find a solution of the line planning problem such that the travelers can travel with a low number of transfers, we also simplify the delay management problem. This means we somehow combine a problem of the strategical level with one of the dispositive level.

The thesis is organized as follows. In the next chapter we introduce the line planning problem and give a short literature survey. In Chapter 3 we introduce a new customer-oriented line planning model. Various integer programming models are proposed and discussed. In Chapter 4 we present different heuristic approaches. In Chapter 5 we dwell on the structure of the integer programs to solve its LP-relaxation by using a Dantzig-Wolfe approach. All these techniques are combined finally to an exact solution approach presented in Chapter 6. In the last chapter, we draw some conclusion and give a prospect of future research on the line planning problem.

Part I

Modeling Line Planning Problems

Chapter 2

Survey on line planning literature

In this chapter we will give a short survey on the work that has been done in the field of line planning in the last century. It does not claim to be complete but tries to classify the - with respect to this thesis - most important publications.
First we will show how passengers data is estimated. Then, after a formal problem description, we will explain the two major ways how to find a feasible line concept. In Section 2.4 we will explain in more detail the main types of line planning approaches, namely the cost- and the customer-oriented approaches. We will close the chapter with some information about the real-world data that is used in this thesis.

2.1 Passenger demand

The volume of traffic or passenger demand must be given to establish a customer-oriented transportation service. Given a set of stations S, the conventional form is an $S \times S$ *origin-destination matrix*, where the entry in the ith row and the jth column denotes the number of passengers that want to travel from station i to station j within a given time period. The origin-destination matrix is difficult, and often costly to obtain by direct measurements or interviews, but by using traffic counts and other available information one may obtain a "reasonable" estimate. Various approaches to estimate the origin-destination matrix using traffic counts have been developed and tested. [Abr98] gives a detailed survey on these approaches.
The origin-destination matrix, in general, refers to the number of trips between two geographic locations which are not classified regarding their travel

mode, such as car, bus, train or airplane. The splitting of all trips over the available travel modes is called *modal split*.
Train travelers often have a number of train lines and connections available to travel from their origin to their destination. These connections may even involve geographically different routes. The passengers choice depends on criteria like travel time, comfort, or ticket price. The problem of estimating how customers travel through the network specific for railway systems is described by [Olt94].
The symmetry depends on the length of the time period as we can assume that most of the customers travel from their origin to their destination and back again since they mainly travel to work or to holidays and back home again. So, an origin-destination matrix for a whole year will be very symmetric whereas the matrix for the time between 6 and 9 o'clock in the morning will not be symmetric at all because it will reflect for example the traffic from the suburbs to the city centers.
In this thesis the origin-destination matrix is assumed to be given and symmetric. Estimating the origin-destination matrix is outside the scope of this thesis.

2.2 Line planning: Problem description and notations

Definition 2.1. A *public transportation network* is a finite, undirected graph $PTN = (S, E)$ with a node set S representing stops or stations, and an edge set E, where each edge $\{u, v\}$ indicates that there exists a direct ride from station u to station v (i.e., a ride that does not pass any other station in between).

Definition 2.2. A *line* is a path in a public transportation network. It is specified by a sequence of stations, or, equivalently a set of edges.

If we consider periodic transportation planning, each line is connected with a number of trips it runs within a given time interval, called the *frequency* f_l

Definition 2.3. A *line concept* L is a set of lines in a public transportation network. In periodic transportation planning, it is a set of lines together with their frequencies (L, f).

Definition 2.4. The *frequency of an edge* e in a line concept (L, f) is given as

$$f(e) = \sum_{l \in L : e \in l} f_l.$$

f_e^{min} and f_e^{max} denote the minimal and maximal allowed frequency on edge $e \in E$.

For example, f_e^{min} can be chosen as the minimal number of vehicles on edge e that is needed to transport all customers. The upper frequency f_e^{max} is often defined due to safety reasons.

We can generalize all line planning problems treated so long by a *basic line planning problem*:

Definition 2.5. Given a public transportation network and an origin-destination matrix, the *basic line planning problem* is to find a line concept, such that all customers are served.

2.3 Finding a feasible line concept

There are two ways to find a set of lines.

1. We can choose lines out of a given set of possible lines, the so called *line pool* $\mathcal{L}$. In this case, the basic line planning problem can be formulated as

   Definition 2.6. Given a public transportation network, a line pool $\mathcal{L}$, and lower and upper frequencies $f_e^{min} \leq f_e^{max}$ for all $e \in E$, the *basic line planning problem* (LP0-pool) is to find a line concept (L, f) with $L \subseteq \mathcal{L}$ and $f_l \in \mathbb{N}$ for all $l \in L$ such that the *line frequency requirement*

   $$f_e^{min} \leq \sum_{l \in L : e \in l} f_l \leq f_e^{max} \tag{2.1}$$

   holds for all $e \in E$.

   This problem is known to be NP-complete, even if $f_e^{min} = f_e^{max} = 1$ for all $e \in E$ (see [Bus98]).

   If we neglect the upper frequencies, the problem is polynomial solvable by the following algorithm.

   Algorithm 2.7. ([Sch01a])
   Given: PTN, line pool $\mathcal{L}$, lower frequencies f_e^{min} for all $e \in E$

   (a) Set $L = \emptyset$, $f_l = 0$ for all $l \in \mathcal{L}$

(b) If for all $e \in E : \sum_{l \in L: e \in l} f_l \geq f_e^{min}$ Stop: (L, f) is a feasible line concept.
Otherwise take some $e \in E$ with $\sum_{l \in L: e \in l} f_l < f_e^{min}$

(c) If there is a line $l \in \mathcal{L}$ with $e \in l$ define $L := L \cup \{l\}, f_l := f_e^{min}$ and goto Step 2.
Otherwise stop: no feasible line concept exists.

2. We can construct lines from scratch. The resulting basic line planning problem can be formulated as follows.

Definition 2.8. Given a public transportation network, and lower and upper frequencies $f_e^{min} \leq f_e^{max}$ for all $e \in E$, the *basic line planning problem* (LP0-construct) is to construct a line concept (L, f) with $f_l \in \mathbb{N}$ for all $l \in L$ such that the *line frequency requirement*

$$f_e^{min} \leq \sum_{l \in L: e \in l} f_l \leq f_e^{max} \qquad (2.1)$$

holds for all $e \in E$.

This problem is easy to solve, if no additional constraints have to be satisfied since we just have to define a line for each edge $e \in E$ with frequency f_e^{min}.

2.4 Objectives

If we solve the basic line planning problem (LP0-pool) or (LP0-construct), we get a feasible line concept. But we do not know how "good" it is. There are three (conflicting) criteria that need to be considered:

1. Serve all customers.
2. Maximize passengers convenience.
3. Minimize the costs of the public transportation company.

The first criterion is achieved by the line frequency requirement (2.1). As we have already mentioned, the lower edge frequencies are chosen such that all customers are transported. Therefore, we have to estimate or assume how passengers will traverse the network. One possibility is to assume an a priori

assignment of passengers to geographical routes through the network. From the passengers perspective, this is comparable to the restrictions enforced by the ticket regulations. The freedom of the line planning model is now restricted to assigning the passengers to train lines along the prescribed route. This assumption has two important implications. Firstly, it can be used to exclude unrealistic traffic assignments in which some travelers have to travel a large detour. Secondly, this assumption can be used to simplify the line planning models considerably, for example in the multi-type line planning problem in [Goo04], where different types of trains, such as high speed and regional trains are considered. The prescribed routes could be the shortest paths in the PTN or the observed routes taken by the customers traveling to the current line concept and timetable.
Another possibility is to let the line planning model determine the traffic assignment, that is ideal for minimizing the operating cost of the line concept. In this case the model dictates the routes of the passengers. This solution can be far from the true traffic assignment made by the passengers themselves. It could force a passenger on a large detour if this would have a positive effect on the chosen objective.
It is also possible to estimate the time needed to change from one train to the next to predict the shortest path of passengers with respect to travel time. Different to the first traffic assignment, this assignment depends on the chosen line concept. It prevents unnecessarily large detours and will be used in the model developed in this thesis.

The second criterion, we have to consider, is the maximization of the passengers convenience. This can be short travel times or a small number of transfers. Customer-oriented approaches are described in more detail in Section 2.4.2.

Contrarily the third criterion, the minimization of the operational costs, asks for an efficient use of its resources such as rolling stock. Note that rolling stock planning is a self-contained field in traffic planning (see e.g. [Sch93] or [PK03]). Simplifying, we can assign each line some cost, depending on the length of the route, the number of coaches, the type of the engine, the number of crew members, etc.. Cost-oriented line planning approaches have been investigated extensively in the last ten years. A literature survey on cost-oriented approaches is given in Section 2.4.1.

2.4.1 Cost-oriented approaches

In the *cost-oriented approaches* next to the lines and their frequencies also the number of coaches per train are determined. The number of coaches is assumed to be identical for each train serving line l. These identical trains are called *compositions*.
All cost-orientated models in the literature have the common style:

(LPP)

$$\min \sum_{l \in \mathcal{L}} f_l \cdot cost_l$$

s.t.

$$\begin{aligned} f_e^{min} \leq \sum_{\substack{l \in \mathcal{L} \\ e \in l}} f_l \leq f_e^{max} \quad & \forall\ e \in \mathcal{E} \\ f_l \geq 0 \quad & \forall\ l \in \mathcal{L} \\ f_l \in \mathbb{N}_0 \quad & \forall\ l \in \mathcal{L} \end{aligned}$$

where f_l is the frequency of line l, f_e^{min}, f_e^{max} the edge restrictions explained in Section 2.2. Since we minimize f_l in the objective function, the lower frequencies f_e^{min} are important to make sure that all passengers are transported (see Section 2.4, 1st criterion). $cost_l$ is a cost factor for each line which might be divided into various parts.

In the master thesis of Claessens ([Cla94]), (see also [CvDZ98], [ZCvD96]) the costs are divided into:

- *fixed cost* per coach ($cost^{c_{fix}}$) and motor unit ($cost^{t_{fix}}$) given by depreciation cost, capital cost, fixed maintenance cost, cost of overnight parking
- *variable cost* per coach ($cost^{c_{var}}$) and motor unit ($cost^{t_{var}}$) given by energy consumption and maintenance cost

The number of coaches in an operating line is bounded from below and above by NC^{min} and NC^{max}, while SC denotes the capacity of a single coach. Also a turn-around time for cleaning, maintenance and changing the crew is considered. The running time plus the needed cleaning time is divided by the length of the time interval to obtain the number of compositions T_l needed for operating a line once per basic time interval. The multiplication of T_l

with the calculated frequency of line l rounded up to the next smallest integer gives us the total number of compositions needed to serve line l. With variables f_l giving the frequency for each line l and y_l counting the coaches needed we get the following nonlinear model:

(COSTNLP)

$$\min \sum_{l \in \mathcal{L}} \lceil T_l \cdot f_l \rceil (cost^{t_{fix}} + (y_l + NC^{min}) cost^{c_{fix}})$$

$$+ d_l \cdot x_l \cdot (cost^{t_{var}} + (y_l + NC^{min}) \cdot cost^{c_{var}})$$

s.t.

$$\begin{aligned}
& f_e^{min} \leq \sum_{e \in l \in \mathcal{L}} f_l \leq f_e^{max} && \forall\ e \in \mathcal{E} \\
& \sum_{e \in l \in \mathcal{L}} SC \cdot f_l (y_l + NC^{min}) \geq ld(e) && \forall\ e \in \mathcal{E} \\
& NC^{min} \leq y_l \leq NC^{max} && \forall\ l \in \mathcal{L} \\
& f_l \in \{0, 1, \ldots, f^{max}\}, y_l \in \mathbb{N}_0 && \forall\ l \in \mathcal{L}
\end{aligned}$$

The model has integer variables, discontinuous and quadratic terms. As solving it by Lagrangian relaxation gives only poor results, a heuristic has been proposed which is solvable in reasonable time. The heuristic is based on a relaxation combined with iterative rounding of the f_l variables. If we fix f_l-variables we get an integer multi-commodity flow problem.

Two linearizations to this nonlinear model are proposed in [CvDZ98] and [Bus98].

1. In the (COSTNLP) model the quadratic terms always have the form $f_l \cdot y_l$ and are substituted in [CvDZ98] by introducing a new class of variables

$$z_{l,\phi,\gamma} = \begin{cases} 1 & \text{if the line concept contains a line } l \\ & \quad \text{with frequency } \phi \text{ and } \gamma \text{ coaches} \\ 0 & \text{otherwise} \end{cases}$$

   Now, the quadratic term can be substituted by

$$f_l y_l := \sum_{\phi \in \{1, \ldots, f^{max}\}} \sum_{\gamma = NC^{min}}^{NC^{max}} \phi \gamma z_{l,\phi,\gamma}$$

and f_l by

$$f_l := \sum_{\gamma=NC^{min}}^{NC^{max}} \phi z_{l,\phi,\gamma}.$$

The discontinuous $\lceil T_l f_l \rceil$ term is substituted by

$$\lceil T_l f_l \rceil := \sum_{\phi \in \{1,\dots,f^{max}\}} \sum_{\gamma=NC^{min}}^{NC^{max}} \lceil T_l \phi \rceil \gamma z_{l,\phi,\gamma}.$$

The resulting formulation (COSTBLP) is a binary linear program with an unchanged number of constraints but much higher number of variables. In real-world instances the number of variables grows by a factor of 10. Therefore some preprocessing must be done before the model can be solved using a general linear programming-based branch-and-bound algorithm.

2. The second linearization of (COSTNLP) is proposed in [Bus98]. In this approach the quadratic terms are avoided by introducing binary variables for the combination of a particular line and frequency:

$$x_{l,\phi} = \begin{cases} 1 & \text{if the line concept contains a line } l \text{ with frequency } \phi \\ 0 & \text{otherwise} \end{cases}$$

and integer variables $y_{l,\phi} \in \mathbb{N}_0$ representing the number of coaches of the line l with frequency ϕ.

The quadratic term in the (COSTNLP) can now be substituted by

$$f_l y_l := \sum_{\phi \in \{1,\dots,f^{max}\}} \phi y_{l,\phi}$$

if we guarantee that

$$y_{l,\phi} \geq NC^{min} \Leftrightarrow x_{r,\phi} = 1.$$

f_l can be replaced by

$$f_l = \sum_{\phi \in \{1,\dots,f^{max}\}} \phi x_{l,\phi}$$

and the discontinuous term $\lceil T_l f_l \rceil$ by

$$\lceil T_l f_l \rceil = \sum_{\phi \in \{1,\dots,f^{max}\}} \lceil \phi T_l \rceil y_{l,\phi}.$$

In the resulting integer linear program (COSTILP) the number of variables grows by a factor of f^{max} and the number of constraints increases by $|\mathcal{L}| \cdot f^{max}$ compared to (COSTNLP). In comparison to (COSTBLP), the size of the model significantly reduces whereas the quality of the initial linear programming relaxation keeps unchanged. Again, preprocessing is done to reduce the size.

Comparing (COSTILP) and (COSTBLP) using a cutting plane algorithm, we get that the much more compact (COSTILP) formulation is preferable to generate good feasible solutions. On the other hand the lower bounds provided by the (COSTBLP) are superior to the lower bounds of the (COSTILP). Therefore the (COSTBLP) is preferable for proofing optimality of a feasible solution.

In [GvHK04] it is shown that the (COSTBLP) formulation *can* be used to solve large instances of the problem using branch-and-cut.
In [Goo04] and [GvHK02] the authors get rid of the assumption that the passengers are assigned a priori for example by modal split to different types of trains. This is done by assigning every node in the PTN a certain *type*, representing for example the size of the station.Then the type of a line determines the stations they pass. For example a line of type 1 stops at every station it passes, a line of type 2 will not halt at a station of type 1 but at every station of type 2 or higher. Several models, correctness and equivalence proofs are presented.
Recently, a fast heuristic variable fixing procedure which combines nonlinear techniques with integer programming is proposed in [BLL04].
In [Goo04] a model that reconsiders the stations at which the trains stop for a given line plan. This model is used to determine the halting stations in such a way that the total travel time of passengers is minimized. Lagrangian relaxation is used to find lower bounds for this problem. Preprocessing and tree search techniques augment the efficiency of the branch&bound framework, the bounds are used for.

2.4.2 Customer-oriented approaches

From the customers point of view a good public transportation system is cheap, fast, reliable, and serves directly with a high frequency from origin to destination. Of course not all of these objectives can be pleased at once. Customer-oriented line planning tries to find a line plan that offers a good comfort to the passenger, such as fast connections with a small number of transfers. This can be done by maximizing the number of direct travelers

which is quite well studied in literature.
In the *direct travelers approach* the objective is to maximize the number of direct customers (i.e. customers that need not change the line to reach their destination). In this case, an upper bound to the number of vehicles is important. To this end, we either have to make sure that not more vehicles are established than it is possible due to safety reasons:

$$\sum_{l \in L: e \in L} f_l \leq f_e^{max} \ \forall \ e \in E \tag{2.2}$$

or more than it is affordable for the transportation company

$$\sum_{l \in L} cost_l f_l \leq B$$

where $cost_l$ is the fixed cost the company has to spend to run a vehicle on line l and B is the budget the company is willing to spend.
This *budget constraint* is used in [Sim80], [Sim81b] and [Sim81a], where Simonis presents a solution approach that starts with an empty line concept and adds successively lines on shortest paths with a maximum number of direct travelers. The algorithm stops if all passengers find an appropriate travel path or the budget constraint is violated.

The problem of maximizing the number of direct travelers with respect to upper line frequency requirements (2.2) has been well studied. Starting in 1925, where Patz presents a first model for the line optimization problem that determines a line plan with small penalty. The penalty is calculated with respect to the number of empty seats and the number of passengers changing to another line to reach their destination. The algorithm starts with a line plan containing a line for each non-zero entry of the origin-destination matrix, the so called *origin-destiantion pairs*. Lines will be eliminated in a greedy method with respect to the penalty.
Such greedy heuristics that either add lines to an empty set or delete lines from a line pool are also presented in [Son77], [Son79], [RR92], [PRE95], and [Völ01]. Other heuristics construct lines out of small pieces, see [LS67], [Weg74], [Son77] and [Son79].
In [Sch01b] the passengers demand is not assumed to be fixed but depends on the level of service.
A recent work by Quak [Qua03] treats line planning for buses instead of trains. He develops a two phase algorithm with the construction of the lines in the first and setting of frequencies and departure times in the second phase. In contrary to the other models he is not taking lines out of a given line pool

but constructs them from the scratch, which is the main part of his work. The two main objectives of this model are "minimizing the total drive time of the vehicles" to keep the costs for the company low and "minimizing the mean detour time of the passengers requests" to keep the passengers comfort high since short travel times are requested by the passengers. As he sets up also a timetable in the second phase, he tries to keep the changing times low to couple the second objective. But if the changing times are low, the risk of loosing a connection in case of a small delay in the network is very high. We also have to mention that a transfer is a bigger discomfort than a slightly longer travel time.

In the last ten years, exact solution methods have been proposed. The two main approaches will be explained in the following in more detail.

Bussieck, [Bus98]

In his PhD-thesis as well as in [ZBKW97], [BKZ96], and [BZ97], Bussieck et al. decompose the network into a short-distance, a medium-distance and a long-distance network and compute them independently because a customer who wants to travel a long distance from a small town A to a small town B will in general not find a direct connection that does not stop at each small village in between. So this customer will travel first from the small town A to the next big town C, change there to the long-distance network to travel to a big town D, which is near the small town B and change back to the short-distance network.
Bussieck assumes the customers to travel on shortest path P_{ij} respective travel time which is reasonable on long-distance networks but not in local bus-networks of bigger towns. With this assumption the *travel load* $ld(e)$ on each edge $e \in E$ can easily be calculated by

$$ld(e) := \sum_{(s,t)\in\mathcal{R}:e\in P_{st}} w_{st}$$

with $\mathcal{R}$ be the set of origin-destination pairs and w_{st} their weights (i.e. the number of customers traveling from node s to node t). With fixed vehicle capacity VC, we now get

$$f_e^{min} \geq \left\lceil \frac{ld(e)}{VC} \right\rceil$$

The upper frequency bound f_e^{max} is motivated by safety reasons and set to 20%:

$$f_e^{max} \geq \left\lceil \frac{1.2 \cdot ld(e)}{VC} \right\rceil$$

With a given line pool, i.e. a set of possible new lines, $\mathcal{L}$, which is reduced to combinations of shortest paths and a class of variables d_{ijl}, counting the direct travelers from node i to node j using line l, we get a mixed integer programming model:

(LOP)

$$\max \sum_{l \in \mathcal{L}} \sum_{\substack{(i,j) \in \mathcal{R} \\ P_{ij} \subseteq l}} d_{ijl}$$

s.t.

$$\sum_{\substack{l \in \mathcal{L} \\ P_{ij} \subseteq l}} d_{ijl} \leq w_{ij} \qquad \forall\ (i,j) \in \mathcal{R} \tag{2.3}$$

$$\sum_{\substack{i,j \in \mathcal{R} \\ e \in P_{ij} \subseteq l}} d_{ijl} \leq VC \cdot f_l \qquad \forall\ e \in E, l \in \mathcal{L} \tag{2.4}$$

$$f_e^{min} \leq \sum_{\substack{l \in \mathcal{L} \\ e \in l}} f_l \leq f_e^{max} \qquad \forall\ e \in E \tag{2.5}$$

$$d_{ijl}, f_l \in \mathbb{N}_0 \qquad \forall\ (i,j) \in \mathcal{R}, l \in \mathcal{L} \tag{2.6}$$

Constraint (2.3) restricts the number of direct travelers on an origin destination pair to the total number of passengers on this relation. Constraints (2.4) prevents that there are more passengers using a line than can be served. Constraint (2.5) are the frequency requirement constraints explained above and constraint (2.6) is the integrality constraint.
The problem is shown to be NP-hard and since the line pool is very big, it is not possible to solve this problem for a real-world instance within reasonable time. Relaxations of the problem are related to relaxations methods of multi-commodity flow problems, including Lagrangian relaxation. Furthermore, constraint (2.4) can be relaxed to

$$d_{ijl} \leq VC \cdot f_l \qquad \forall\ (i,j) \in \mathcal{R}, l \in \mathcal{L} : P_{ij} \subseteq l \tag{2.7}$$

and furthermore all these constraints can be aggregated to one for each origin-destination pair:

$$\sum_{l \in \mathcal{L}: P_{ij} \subseteq l} d_{ijl} \leq VC \sum_{l \in \mathcal{L}: P_{ij} \subseteq l} f_l \qquad \forall\ (i,j) \in \mathcal{R} \tag{2.8}$$

In the resulting model the d_{ijl} variable always occur in the form $\sum_{l \in \mathcal{L}: P_{ij} \subseteq l} d_{ijl}$ and can be substituted to a new variable $D_{i,j}$:

(lop)

$$\max \sum_{(i,j)\in\mathcal{R}} D_{ij}$$

s.t.

$$\begin{aligned}
&D_{ij} \le w_{ij} && \forall\ (i,j) \in \mathcal{R} (1)\\
&D_{ij} \le VC \sum_{l\in\mathcal{L}:P_{ij}\subseteq l} f_l && \forall\ (i,j) \in \mathcal{R} (2^{**})\\
&f_e^{min} \le \sum_{\substack{l\in\mathcal{L}\\ e\in l}} f_l \le f_e^{max} && \forall\ e \in E (3)\\
&D_{ij}, f_l \in \mathbb{N}_0 && \forall\ (i,j) \in \mathcal{R}, l \in \mathcal{L} (4)
\end{aligned}$$

The integrality of f_l, w_{ij} and VC implies the integrality of D_{ij}, such that the integrality constraint of the D_{ij} variables can be relaxed to $D_{ij} \ge 0$ without any changes on the solution but with significant reduction of the problem size. A similar relaxation can be done in the (LOP) but here the integrality of the d_{ijl} variables is not implied by the f_l variables.
The solution of the relaxed model (lop) provides a feasible line plan, but the objective value of an optimal solution of (lop) gives an upper bound for the number of direct travelers, only.
The size of the (lop) is substantially reduced and the linear relaxation is quite fast even for larger instances. Nevertheless more computation time can be saved by preprocessing and constraint generation.

Dienst, [Die78]

In 1978, Dienst proposes a branch-and-bound algorithm for the line planning problem with respect to the number of direct travelers based on the following simplification of the problem, described in section 2.4.2. Dienst assumes an infinite train capacity and sets $f_e^{max} = 1$ for all $e \in E$. f_e^{min} is used to overcome infeasibility. He tries to get a line cover by adding successively lines. After adding a line, the data is updated:

1. All origin-destination pairs $(i,j) \in \mathcal{R}$ with $P_{ij} \subseteq l$ are deleted from $\mathcal{R}$, because these customers are served by line l.

2. If for any edge $e \in E$ the lower frequency bound $f_e^{min} = 1$, no other line will use this edge (since $f_e^{max} = 1$ by assumption). Therefore all origin-destination pairs $(i,j) \in \mathcal{R}$ with $e \in P_{ij}$ and $P_{ij} \not\subseteq l$ are deleted from $\mathcal{R}$. These customers will not travel directly to their destination.

3. Reduce f_e^{min} along the new line l.

Dienst searches within a node of the branch-and-bound tree by a Greedy method for the line l^* with the maximal number of direct travelers and branches on it.

- Add line l^* to the partial line cover: The upper bound of the optimal value reduces by the number of not direct travelers (see step (2) in the data correction). The value of the line cover increases by the amount of direct travelers served by line l^*. Correct data.

- Do not add line l^* to the partial line cover: The upper bound of the optimal value reduces by the number of travelers who only can use line l^*. Delete these origin-destination relations from $\mathcal{R}$.

The rest of the algorithm are standard branch-and-bound methods. The algorithm works quite good, but is very slow, such that it will be stopped after a given time or number of steps. The resulting problems are that the actually best line cover might be infeasible if it cannot be completed by the remaining lines and if the algorithm is stopped by time or iteration restriction, the relation of the calculated solution to the optimal solution is unknown.

2.5 Real-world applications

The solution approaches presented in this thesis are tested on real-world data of German Railway (DB) and Intranetz. The instances are based on the german long-distance train network, shown in Figure 2.1. It consists of 233 stations and 319 edges. The origin-destination matrix has 35322 non-zero entries. The given line pools sometimes do not cover all customers, i.e. even if all lines would be used some customers would not reach their destination. In this case the models presented in this thesis would be infeasible. Therefore we deleted these origin-destination pairs in a preprocessing step. Table 2.1 shows the instances used in this thesis. Instance No.0 is the instance of Example 3.11, Figure 3.5. $|\mathcal{L}|$ denotes the number of lines in the line pool and $|\mathcal{R}|$ the number of non-zero origin-destination pairs.
The line pools are generated by the line pool generator of DB. It uses different methods to produce a huge set of lines, such as

- *enumeration* of (straight) paths or sequences of stations which produces good but "'main-stream"'-lines

- *constructive graph algorithms* such as

No.	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{R}\rvert$
0	3	2
1	10	2602
2	50	4766
3	100	11219
4	132	18238
5	200	10126
6	250	13246
7	275	14071
8	300	17507
9	330	18433
10	350	17095
11	375	18350
12	400	22191
13	423	22756

Table 2.1: Instances for different line pool sizes

- *global spanning tree*: solve a Max-Spanning-Tree problem of a set of important stations. A path in this tree is a line. Choose the "best" line and reduce arc weights. Recalculate Max-Spanning-Tree.
- *local spanning tree*: for each start or end station distribute the demand on the network. Solve a Max-Spanning-Tree problem and determine the x "best" lines.

This huge set of possible lines is then reduced by using ad-hoc-filters, quality-functions or tools avoiding detours to get a line pool of potential lines.

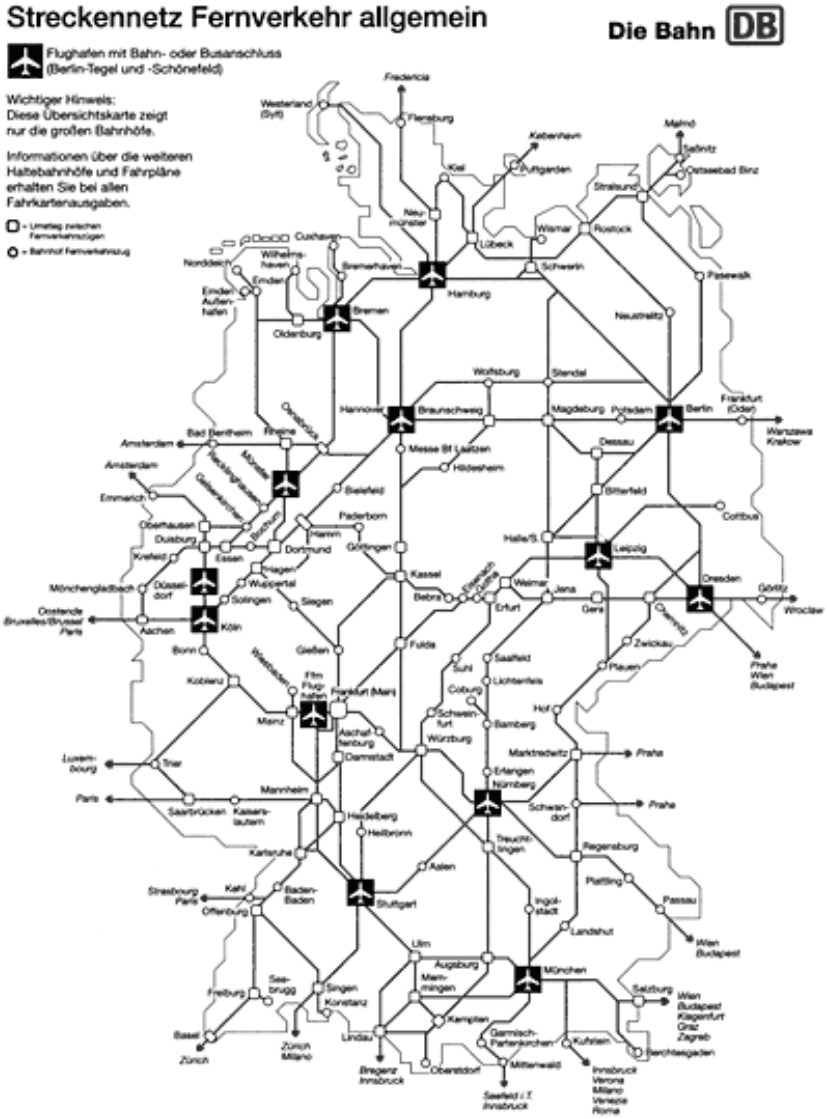

Figure 2.1: The network with the main stations of long-distance trains in Germany. [Rai]

Chapter 3

A new Model

In this chapter we present a new model to find a line concept minimizing the travel time over all customers while taking into account the number of transfers needed on their ways. This approach maximizes the comfort of the passengers. Since it is possible to put a penalty on transfers, the resulting timetable will be more reliable. In Section 3.1 we introduce some basic definitions and motivate why there is need for a new model. We show the difference between the direct travelers approach of Bussieck and our approach in an example. In Section 3.2 we present some complexity results. In Section 3.3 we introduce an extended network and various integer formulations which are discussed in Section 3.4.

3.1 Motivation and basic definitions

In this section we introduce some basic definitions we need for our model and show that the direct traveler approach of Bussieck (see Section 2.4.2) is not equivalent to our approach.
We first define the network which we assume to be given and fixed.

Definition 3.1. A *public transportation network* is a finite, undirected graph $PTN = (S, E)$ with a node set S representing stops or stations, and an edge set E, where each edge $\{u, v\}$ indicates that there exists a direct ride from station u to station v (i.e. a ride that does not pass any other station in between). For each edge $\{u, v\}$ we assume that the driving time t_{uv} is known.

Definition 3.2. The *line pool* $\mathcal{L}$ is a set of paths in the PTN. Each *line* $l \in \mathcal{L}$ is specified by a sequence of stations, or, equivalently, by a sequence of edges. Let $E(l)$ be the set of edges belonging to line l. Given a station $u \in S$ we furthermore define

$$\mathcal{L}(u) = \{l \in \mathcal{L} : u \in l\}$$

as the set of all lines passing through u.

The next definition corresponds to the passengers demand.

Definition 3.3. $\mathcal{R} \subseteq S \times S$ denotes the set of all *origin-destination pairs* (s,t) where w_{st} is the number of customers wishing to travel from station s to station t.

As we want to keep the number of transfers small in our approach, we first have to define formally, what a transfer is.

Definition 3.4. Given a set of lines $L \subseteq \mathcal{L}$, a customer can travel from his origin s to his destination t, if there exists an s-t-path P in the PTN only using edges $\{E(l) : l \in L\}$. The minimum number of lines of L needed to cover these edges minus one is the number of *transfers* needed by this customer.

The line planning problem then is to choose a subset of lines $L \subseteq \mathcal{L}$, the so called *line concept*, which

- allows each customer to travel from its origin to its destination,
- is not too costly, and
- minimizes the "inconvenience" for the customers.

In the literature, the common customer-oriented approach dealing with the inconvenience of the customers is the approach of [Bus98] (see also [BKZ96]) in which the number of direct travelers is maximized. In this work, however, we deal with the sum of *all* transfers over all customers. On a first glance, the problem to minimize the number of transfers seems to be similar to maximizing the number of direct travelers. That is in general not the case, as the following example demonstrates.

Example 3.5. Given a PTN with 9 nodes and 8 edges as shown in Figure 3.1 and a line pool $\mathcal{L}$ containing 11 lines $\mathcal{L} = \{l_1, \ldots, l_{11}\}$ shown in Table 3.1. In Figure 3.1 for simplicity only lines l_1, l_2 and l_3 are named. The remaining lines correspond to the single edges of the PTN. Let the set of origin-destination pairs be $\mathcal{R} := \{(1,3),(2,8),(7,9)\}$ with customer demand $w_{1,3} = w_{2,8} = w_{7,9} = 1$. Assume that due to safety rules not more than one vehicle per edge is allowed within our planning period of e.g. 30 minutes.
Then the optimal solutions for the two objectives are the following line concepts:

line	stations
l_1	1,2,3
l_2	7,8,9
l_3	2,3,4,5,6,7,8
l_4	1,2
l_5	2,3
l_6	3,4
l_7	4,5
l_8	5,6
l_9	6,7
l_{10}	7,8
l_{11}	8,9

Table 3.1: The line pool of example 3.5.

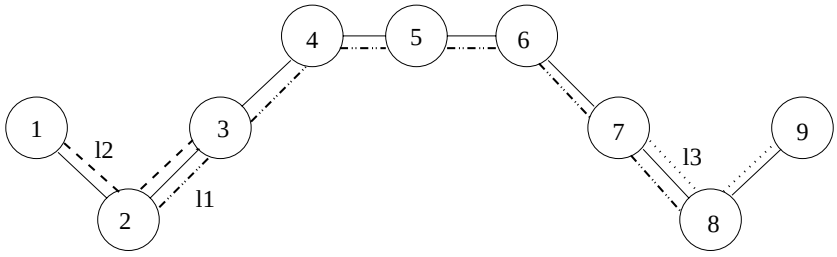

Figure 3.1: Difference between the objectives "maximize direct travelers" and "minimize transfers".

- "maximize number of direct travelers": $L = \{l_2, l_3, l_6, l_7, l_8, l_9\}$, see Figure 3.2
  In this case the two passengers $(1, 3)$ and $(7, 9)$ can travel directly, but passenger $(2, 8)$ has 5 transfers.
- "minimize number of transfers": $L = \{l_1, l_4, l_{11}\}$, see Table 3.3
  In this case only one passenger, namely passenger $(2, 8)$ travels directly, but the total number of transfers is only two because passengers $(1, 3)$ and $(7, 9)$ have to change the vehicle once each.

Note that considering the number of transfers only would lead to solutions with very long lines, serving all origin-destination pairs directly but having large detours for the customers. To avoid this we determine not only a line concept, but also a path for each origin-destination pair and count the number of transfers *and* the length of the paths in the objective function. This

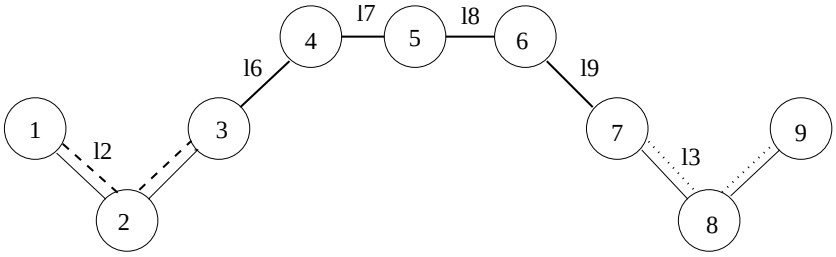

Figure 3.2: Solution "maximize direct traveler".

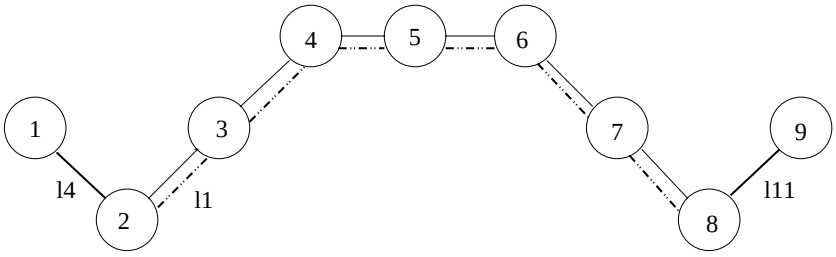

Figure 3.3: Solution "minimize transfers".

is specified next.

Given a set of lines $L \subseteq \mathcal{L}$, a customer can travel from his origin s to his destination t, if there exists an s-t-path P in the PTN only using edges in $\{E(l) : l \in L\}$. The "inconvenience" of such a path is then approximated by the weighted sum of the traveling time Time_P along the path and the number of transfers Transfers_P, i.e.

$$\text{inconvenience}(P) = k_1 \text{Time}_P + k_2 \text{Transfers}_P. \tag{3.1}$$

On the other hand, the cost of a line concept $L \subseteq \mathcal{L}$ is calculated by adding the costs C_l for each line $l \in L$, assuming that such costs C_l are known beforehand.

Definition 3.6. If each origin-destination pair can be served, the line concept is called *feasible*.

We can now give a formal definition of the line planning problem.

Definition 3.7. The *line planning problem* hence is to find a feasible set of lines $L \subseteq \mathcal{L}$ together with a path P for each origin-destination pair, such that

the costs of the line concept do not exceed a given *budget* B and such that the sum of all inconveniences as defined in (3.1) over all paths is minimized.

Since the capacity of a vehicle is not arbitrarily large, we have to extend the basic problem to include frequencies of the lines. This makes sure that there are enough vehicles along each edge to transport all passengers. We remark that often, the number of vehicles running along the same edge is also bounded from above, e.g., for safety reasons.

3.2 Complexity

3.2.1 An introduction to computational complexity

The theory of computational complexity tries to categorize the computational requirement of algorithms and important classes of problems. Although, this theory and the corresponding notation can be found in literature (e.g. [Wol98], [NW88]) we repeat the basic concepts here to make the thesis more self-contained.

The *time complexity* of a problem is the number of steps that it takes to solve an instance of the problem, as a function of the size of the input, (usually measured in bits) using the most efficient algorithm. This function is called the *time complexity function*. To understand this intuitively, consider the example of an instance that is n bits long that can be solved in n^2 steps. In this example we say the problem has a time complexity of n^2. Of course, the exact number of steps will depend on exactly which machine or language is being used. To avoid that problem, we generally use *Big O notation*. An algorithm is said to run in $O(f(n))$ time if for some numbers c and n_0, the time complexity function is at most $c \cdot f(n)$ for all $n \geq n_0$. If a problem has time complexity $O(n^2)$ on one typical computer, then it will also have complexity $O(n^2)$ on most other computers, so this notation allows us to generalize away from the details of a particular computer.

An algorithm is said to be a *polynomial-time algorithm* or *efficient algorithm* if it runs in $O(f(n))$ time, where f is bounded by a polynomial of fixed degree, e.g. $O(n^2)$ and $O(n \log n)$. An algorithm is said to be an *exponential-time algorithm* if its complexity function cannot be polynomially bounded by the input size n, e.g. $O(2^n)$ and $O(n!)$.

Obviously, polynomial-time algorithms are "good" algorithms. Nevertheless, we might not succeed in developing a polynomial-time algorithm for a particular problem. The theory of *NP-completeness* provides us a way to prove that the problem is inherently hard in the sense that if we develop an efficient algorithm for this problem, we would be able to develop an efficient algorithm

for a huge class of intractable problems, including famous problems like the traveling salesman problem (TSP) or graph coloring.
The theory of NP-completeness helps us to classify a given problem into broad classes:

1. easy problems that can be solved by polynomial-time algorithms, and
2. hard problems that are not likely to be solved in polynomial-time and for which all known algorithms require exponential time.

Much of complexity theory deals with *decision problems*. A decision problem is a problem where the answer is always YES or NO. The problems discussed in this thesis are *optimization problems*. It is easy to see that the optimization and the decision version of a problem are equivalent in terms of whether or not they can be solved in polynomial time. We refer to an instance of the decision problem as a YES instance if the answer to this problem instance is yes, and a NO instance otherwise. We say that a problem P_1 is *polynomially reducible* to another problem P_2 if for every instance I_1 of P_1 we can construct in polynomial-time in terms of the size of I_1 an instance I_2 of P_2 such that I_1 is a yes instance of P_1 if and only if I_2 is a yes instance of P_2. If problem P_2 is polynomially reducible to problem P_1, P_2 is at least as hard as P_1: Given an algorithm for problem P_2 we can always use it to solve problem P_1 with comparable (i.e. polynomial or not) running time.

We can now specify the most common classes of computational complexity theory:

- P: The class of decision problems that can be solved in polynomial time.
- NP: The class of problems that can be solved in *non-deterministic polynomial* time. For a decision problem P to be in NP we require that if I is a YES instance of P, then there exists a polynomial-time algorithm that verifies the solution.
- NP-complete: A decision problem P is said to be (in the class) NP-complete if
  1. it is in NP, and
  2. All problems in NP are polynomially reducible to P.
- NP-hard: When a decision version of a combinatorial optimization problem is proved to belong to the class of NP-complete problems, then the optimization version is NP-hard.

If we do succeed in showing that a problem is NP-complete, we have sufficient reasons to believe that the problem is hard and no efficient algorithm can ever be developed to solve it. We should concentrate our efforts on developing efficient heuristics and at developing various types of enumeration algorithms.

3.2.2 Complexity of (LPMT)

In this section we first show that the line planning problem as defined in Definition 3.7 is NP-hard, even in a very simple case, corresponding to $k_1 = 0$.

Theorem 3.8. ([SS04a]) The line planning problem is NP-complete, even if

- we only count the number of transfers in the objective function,
- the PTN is a linear graph, and
- the line costs are equal for all lines

Proof. In the decision version, the line planning problem in the above case can be written as follows:
Given a graph PTN=(S, E), the set of all origin-destination pairs $\mathcal{R}$, and a budget B, does there exist a feasible set of B lines with less than K transfers?

We show that the set covering problem is polynomially reducible to our given problem: Given the set covering problem in its integer programming problem formulation

$$\min\{\underline{1}x : Ax \geq 1, x \in \{0,1\}^m\}$$

with an 0-1 $n \times m$-matrix A, we construct a line planning problem as follows: We define the PTN as a linear graph with $2n$ nodes $S = \{s_1, t_1, s_2, t_2 \ldots, s_n, t_n\}$ and edges $E = \{(s_1, t_1), (t_1, s_2), (s_2, t_2), (t_2, s_3), \ldots, (s_n, t_n)\}$. We define an origin-destination pair for each row of A,

$$\mathcal{R} = \{(s_i, t_i) : i = 1, \ldots, n\}.$$

For column j of A we construct a line l_j passing through nodes s_i and t_i if $a_{ij} = 1$. As an example, Figure 3.4 shows the line planning problem obtained from a set covering problem with

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

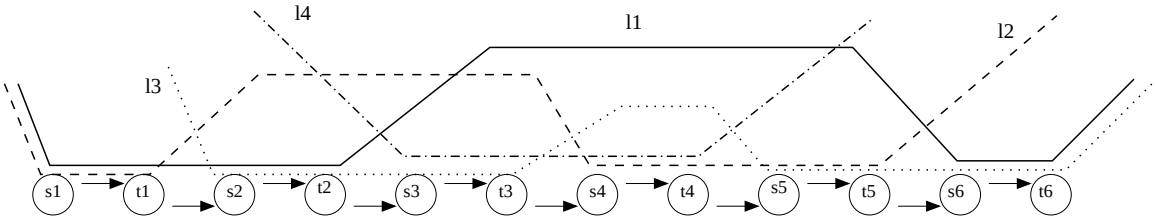

Figure 3.4: Construction of the line planning problem in the proof of Theorem 3.8.

Setting $K = 0$ we hence have to show that a cover with less than B elements exists if and only if the line planning problem has a solution in which all passengers can travel without changing lines.

"$\Rightarrow$": Given a cover with less than B elements. In the integer programming formulation this means, that we have a solution vector $\bar{x} \in \{0,1\}^m$ that satisfies the following

- for all $i \in \{1,\dots,n\}$, there is a $j \in \{1,\dots,m\}$ with $a_{ij} = 1$ and $\bar{x}_j = 1$.
- $\sum_{j=1}^{m} x_j \leq B$

Thus, by construction of the line planning problem, the corresponding solution, the so called line concept consists of less than B lines, no transfers are needed (since $K = 0$) and for all $(s_i, t_i) \in \mathcal{R}$ there exists a line l_j in the line concept that serves this customer directly.

"$\Leftarrow$": Given a solution of the line planning problem in which all passengers can travel without changing lines. This means that each origin-destination pair is served directly by at most B lines. Thus, $\bar{x}_j = 1$ if and only if line j is in the line concept, solves the set covering problem.

■

A question that might arise in this context is what happens if the lines need not be chosen of a given line pool, but can be constructed as any path. Some of the basic cost models become very easy in this case, but unfortunately, the complexity status of the line planning problem treated in this paper does not change.

Theorem 3.9. ([SS04a]) The line planning problem in which *all* possible simple paths are allowed is NP-complete, even if we only count the number of transfers in the objective function.

Proof. We show that the Hamiltonian path problem is polynomially reducible to our problem.
Let $G = (\mathcal{V}, \mathcal{E})$ be the graph in which we want to check the existence of a Hamiltonian path from a given node s to a given node t.

We construct the line planning problem as follows:
We define the PTN as the given graph G and construct

$$\mathcal{R} = \{(s, v), (v, t) : v \in \mathcal{V}\}$$

as the set of origin-destination pairs. Furthermore, we set the budget $B = 1$. The line planning problem with $K = 0$ hence is to find *one* line serving all origin-destination pairs. Such a line must start in s, pass through all nodes and end in t (otherwise at least one element of $\mathcal{R}$ would have to change for its trip), and hence constitutes a Hamiltonian path. Vice versa, any Hamiltonian path is a solution of the line planning problem with a total of zero transfers. ■

3.3 Model formulations

3.3.1 Change&go-network

For line planning we extend the PTN to the undirected *change&go-network* $G_{CG} = (\mathcal{V}, \mathcal{E})$ as follows:
Given a line pool $\mathcal{L}$ and a PTN, we extend the set S of stations to a set $\mathcal{V}$ of nodes consisting of nodes representing station-line-pairs (change&go nodes $\mathcal{V}_{CG}$) and nodes representing the start and end points of the customer paths (origin-destination nodes $\mathcal{V}_{OD}$).
The new set of edges $\mathcal{E}$ consists of edges between nodes of the same station (representing getting in or out of a vehicle, $\mathcal{E}_{OD}$ or changing a line, $\mathcal{E}_{change}$) and edges between nodes of the same line (representing driving on a line, $\mathcal{E}_{go}$).

We now give a formal definition of the new extended graph.

Definition 3.10. Given a public transportation network $PTN = (S, E)$ and a line pool $\mathcal{L}$, the corresponding undirected *change&go-graph* $G_{CG} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes

$$\mathcal{V} := \mathcal{V}_{CG} \cup \mathcal{V}_{OD}$$

with

- $\mathcal{V}_{CG} := \{(s, l) \in S \times \mathcal{L} : l \in \mathcal{L}(s)\}$ (set of all station-line-pairs)

No.	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$
0	3	2	10	16
1	10	2602	419	606
2	50	4766	1015	5576
3	100	11219	1716	16040
4	132	18238	2487	24394
5	200	10126	3590	69308
6	250	13246	4716	111517
7	275	14071	5303	134348
8	300	17507	5931	158372
9	330	18433	6706	191453
10	350	17095	6503	233750
11	375	18350	7101	272414
12	400	22191	7682	300174
13	423	22756	8268	339691

Table 3.2: Instances for different line pool sizes

- $\mathcal{V}_{OD} := \{(s,0) : (s,t) \in \mathcal{R} \text{ or } (t,s) \in \mathcal{R}\}$ (in/out-nodes)

and edges

$$\mathcal{E} := \mathcal{E}_{change} \cup \mathcal{E}_{go} \cup \mathcal{E}_{OD}$$

with

- $\mathcal{E}_{change} := \{\{(s,l_1),(s,l_2)\} \in \mathcal{V}_{CG} \times \mathcal{V}_{CG} : l_1 \neq l_2\}$ (changing edges)
- $\mathcal{E}^l := \{\{(s,l),(s',l)\} \in \mathcal{V}_{CG} \times \mathcal{V}_{CG} : \{s,s'\} \in E\}$ (driving edges of line $l \in \mathcal{L}$)
- $\mathcal{E}_{go} := \bigcup_{l \in \mathcal{L}} \mathcal{E}^l$ (driving edges)
- $\mathcal{E}_{OD} := \{\{(s,0),(s,l)\} \in \mathcal{V}_{OD} \times \mathcal{V}_{CG} \text{ and } \{(t,l),(t,0)\} \in \mathcal{V}_{CG} \times \mathcal{V}_{OD} : (s,t) \in \mathcal{R}\}$ (in/out-edges)

The following example shows the extension of a simple public transportation network with a small line pool. Table 3.2 shows some instance sizes of real-world instances.

Example 3.11. We consider the public transportation network with three nodes and two edges of Figure 3.5 together with the line pool

$$\mathcal{L} = \{l1, l2, l3\} = \{(1,2),(2,3),(1,2,3)\}$$

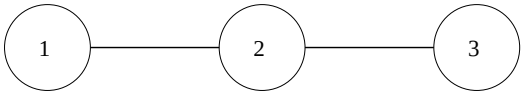

Figure 3.5: The public transportation network of Example 3.11.

We get the change&go-graph with the node sets

$$\mathcal{V}_{OD} = \{(1,0),(2,0),(3,0)\}$$

$$\mathcal{V}_{CG} = \{(1,l1),(2,l1),(2,l2),(3,l2),(1,l3),(2,l3),(3,l3)\}$$

and the edge sets shown in Figure 3.6.

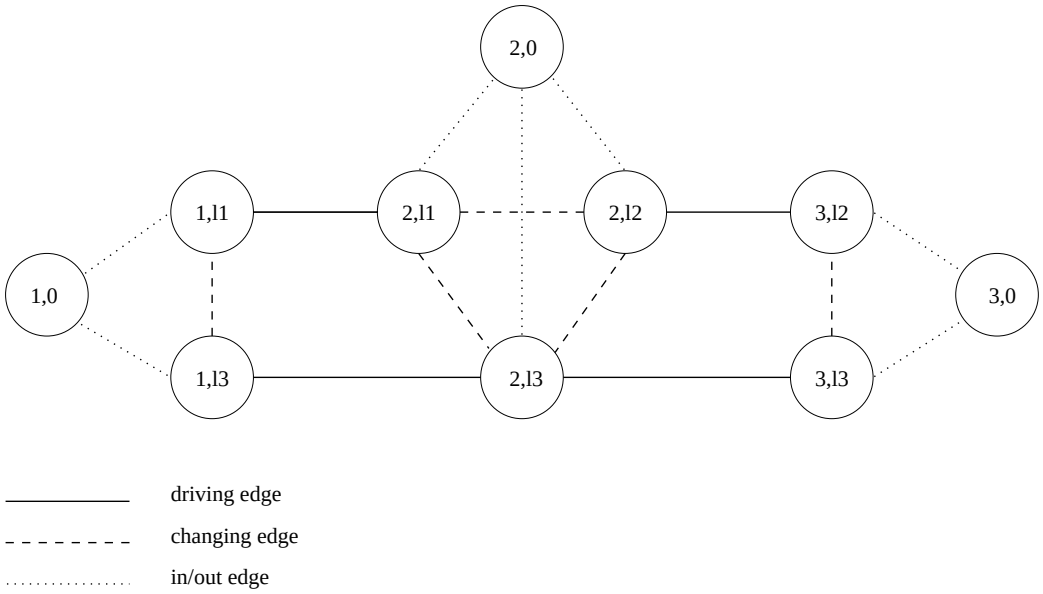

Figure 3.6: The change&go graph of Example 3.11.

We now define weights on all edges $e \in \mathcal{E}$ of the change&go-graph representing the inconvenience customers have by using the edge. Then, for a single origin-destination pair we can determine the lines the customer is likely to use by calculating a shortest path in the change&go-graph. Therefore the choice of the *edge costs* c_e is very important.

Some examples (Remark 3.12 gives an explanation of the choice of the in/out-edges $\mathcal{E}_{OD}$):

1. Customers only count transfers:

$$c_e = \begin{cases} 1 & e \in \mathcal{E}_{change} \\ 1 & e \in \mathcal{E}_{OD} \\ 0 & \text{else} \end{cases}$$

Note that in this case, it is possible to shrink the change&go-network to a *line-change-network* with only $|\mathcal{L}| + |S|$ nodes and $|\mathcal{E}_{change}| + |\mathcal{E}_{OD}|$ edges (see Figure 3.7 as an example.).

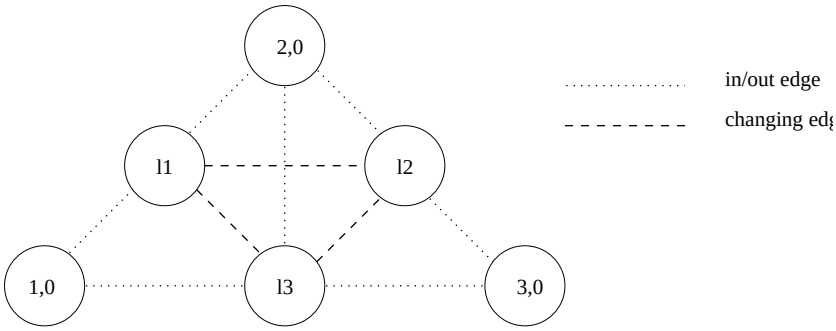

Figure 3.7: The shrunken line-change-network of Figure 3.6 in the special case: "Customers only count transfers".

2. Real travel time:

$$c_e = \begin{cases} c_{e_{OD}} & e \in \mathcal{E}_{OD} \\ t_e & e \in \mathcal{E}_{go} \\ \text{time needed for changing platform} & e \in \mathcal{E}_{change} \end{cases}$$

Remark 3.12. The weights c_e for $e \in \mathcal{E}_{OD}$ must be set to a fixed number at least bigger than $\frac{1}{2} \cdot \max\{c_e : e \in \mathcal{E}_{change}\}$ to avoid that it is cheaper to use two in/out-edges instead of a changing-edge. Remember that in/out-edges represent the start- and end-point of a customers path. So, for each customers path exactly two in/out-edges are used. If we choose these weights $c_{e_{OD}}$ arbitrarily (large enough) and equal for all $e \in \mathcal{E}_{OD}$, we just have to subtract $2 \cdot c_{e_{OD}}$ of the length of any origin-destination path to get the real length.

Remark 3.13. It is often reasonable to make transfers more inconvenient by increasing c_e for all $e \in \mathcal{E}_{change}$ in the real travel time model. It is also

possible to prefer special stations for transfers because of their infrastructure by decreasing the costs of the changing edges corresponding to this station.

Other combinations and variations are possible.

Since we assume that customers prefer shortest paths according to the weights c_e we need an implicit calculation of shortest paths within our model. This is obtained by doubling the edges to get a directed graph and solving the following network flow problem for each origin-destination pair $(s,t) \in \mathcal{R}$.

$$\theta x_{st} = b_{st},$$

where $\theta \in \mathbb{Z}^{|\mathcal{V}|\times|\mathcal{E}|}$ is the node-arc-incidence matrix of the (directed) G_{CG},

$$\theta_{ie} = \begin{cases} 1 & \text{if } e = (i,j) \in \mathcal{E} \\ -1 & \text{if } e = (j,i) \in \mathcal{E} \\ 0 & \text{else} \end{cases}$$

and $b_{st} \in \mathbb{Z}^{|\mathcal{V}|}$ is defined by

$$b^i_{st} = \begin{cases} 1 & \text{if } i = (s,0) \\ -1 & \text{if } i = (t,0) \\ 0 & \text{else} \end{cases},$$

The variables are $x \in \{0,1\}^{|\mathcal{R}|\times|\mathcal{E}|}$ with $x^e_{st} = 1$ if and only if edge e is used on a shortest dipath from node $(s,0)$ to $(t,0)$ in G_{CG}.

To specify the lines in the line concept we introduce a variable $y_l \in \mathbb{B}$ for each line $l \in \mathcal{L}$ which is set to 1 if and only if line l is chosen to be in the line concept.

3.3.2 Integer programming formulations

Our model, *Line Planning with Minimal Transfers* can now be presented in four different binary programming formulations.

The *objective function* we use is customer-oriented. We allow to specify some edge cost c_e for each edge in the change&go-network depending on the objective the decision maker has. There are various possibilities to choose c_e, some of them have been mentioned in Section 3.3.1. Note that the variety of possibilities of edge cost choices is a big advantage of this model. In the

objective function we sum up these costs to the length of a shortest path from s to t for each origin-destination pair $(s,t) \in \mathcal{R}$. Adding over all $(s,t) \in \mathcal{R}$ means that we minimize the average costs of the customers.

$$\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x^e_{st} \tag{3.2}$$

As we want the customer to travel on shortest paths according to the edge costs c_e we introduce *shortest path constraints* for each origin-destination pair as explained in the previous section.

$$\theta x_{st} = b_{st} \quad \forall\ (s,t) \in \mathcal{R} \tag{3.3}$$

Note that so far the best line concept from a customer-oriented point of view would be to introduce all lines of the line pool. This is certainly no option for a public transportation company, since running a line is costly. Let C_l be an estimation of the costs which occur if line l is chosen and let B be the budget the public transportation company is willing to spend. Then the *budget constraint* takes into account the economic aspects.

$$\sum_{l\in\mathcal{L}} C_l y_l \leq B \tag{3.4}$$

Now we need a link between the x- and the y-variables. The *coupling constraints* make sure that a line must be included in the line concept if the line is used by some origin-destination pair. This can be done in a couple of ways. We will represent four of them and discuss their strength and equivalence in Section 3.4.1.

$$\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x^e_{st} \leq |\mathcal{R}||\mathcal{E}^l| y_l \qquad \forall\ l \in \mathcal{L} \tag{3.5}$$

$$\sum_{(s,t)\in\mathcal{R}} x^e_{st} \leq |\mathcal{R}| y_l \qquad \forall\ l \in \mathcal{L}, e \in \mathcal{E}^l \tag{3.6}$$

$$\sum_{e\in\mathcal{E}^l} x^e_{st} \leq |\mathcal{E}^l| y_l \qquad \forall\ l \in \mathcal{L}, (s,t) \in \mathcal{R} \tag{3.7}$$

$$x^e_{st} \leq y_l \qquad \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}^l : l \in \mathcal{L} \tag{3.8}$$

Restricting the variables to $\mathbb{B} := \{0,1\}$, we can now present our four formulations of the *Line Planning with Minimal Transfers* (LPMT).

The first formulation consists of $|\mathcal{L}| + |\mathcal{V}||\mathcal{R}| + 1$ constraints.

(LPMT1)

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x^e_{st} \tag{3.9}$$

$$\begin{aligned} s.t. \quad & \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x^e_{st} \leq |\mathcal{R}||\mathcal{E}^l| y_l && \forall\ l \in \mathcal{L} \\ & \theta x_{st} = b_{st} && \forall\ (s,t) \in \mathcal{R} \\ & \sum_{l\in\mathcal{L}} C_l y_l \leq B \\ & x^e_{st}, y_l \in \mathbb{B} && \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L} \end{aligned}$$

The second formulation consists of $|\mathcal{L}||\mathcal{E}_{go}| + |\mathcal{V}||\mathcal{R}| + 1$ constraints.
(LPMT2)

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x^e_{st} \tag{3.10}$$

$$\begin{aligned} s.t. \quad & \sum_{(s,t)\in\mathcal{R}} x^e_{st} \leq |\mathcal{R}| y_l && \forall\ e \in \mathcal{E}^l : l \in \mathcal{L} \\ & \theta x_{st} = b_{st} && \forall\ (s,t) \in \mathcal{R} \\ & \sum_{l\in\mathcal{L}} C_l y_l \leq B \\ & x^e_{st}, y_l \in \mathbb{B} && \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L} \end{aligned}$$

The third formulation consists of $(|\mathcal{L}| + |\mathcal{V}|)|\mathcal{R}| + 1$ constraints.
(LPMT3)

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x^e_{st} \tag{3.11}$$

$$\begin{aligned} s.t. \quad & \sum_{e\in\mathcal{E}^l} x^e_{st} \leq |\mathcal{E}^l| y_l && \forall\ l \in \mathcal{L}, (s,t) \in \mathcal{R} \\ & \theta x_{st} = b_{st} && \forall\ (s,t) \in \mathcal{R} \\ & \sum_{l\in\mathcal{L}} C_l y_l \leq B \\ & x^e_{st}, y_l \in \mathbb{B} && \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L} \end{aligned}$$

The fourth formulation consists of $(|\mathcal{L}||\mathcal{E}_{go}| + |\mathcal{V}|)|\mathcal{R}| + 1$ constraints.
(LPMT4)

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x^e_{st} \tag{3.12}$$

$$\begin{aligned} s.t.\quad & x^e_{st} \le y_l && \forall\ (s,t)\in\mathcal{R}, e\in\mathcal{E}^l : l\in\mathcal{L} \\ & \theta x_{st} = b_{st} && \forall\ (s,t)\in\mathcal{R} \\ & \sum_{l\in\mathcal{L}} C_l y_l \le B \\ & x^e_{st}, y_l \in \mathbb{B} && \forall\ (s,t)\in\mathcal{R}, e\in\mathcal{E}, l\in\mathcal{L} \end{aligned}$$

3.3.3 Bicriteria formulation

As we have mentioned above we have to restrict the number of lines allowed to be chosen out of the line pool. Otherwise all lines of the line pool would be chosen which is not affordable by any company. In Section 3.3.2 we have treated this problem by introducing the budget constraint (3.4). Of course it would also be possible to minimize the number of lines while minimizing the inconvenience of the passengers at the same time. This can be done by formulating bicriteria programs which can be solved using methods of multiobjective optimization.

Basics of bicriteria optimization (e.g. [Ehr00])

The *bicriteria optimization problem* we consider in this section is given by a discrete set of feasible points $X \subseteq \mathbb{Z}^n$ and two objective functions $f_1, f_2 : X \to \mathbb{R}$.

(BP)

$$\min_{x\in X} \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix}$$

Definition 3.14. (e.g. [Ehr00]) Let $x_1, x_2 \in X$

- x_1 *dominates* x_2 if

$$f_1(x_1) \le f_1(x_2)$$

and

$$f_2(x_1) \leq f_2(x_2)$$

where at least one of the inequalities is strict.

- $x \in X$ is a *Pareto solution*, if there does not exist any $y \in X$ that dominates x.

The goal in bicriteria optimization is to determine the Pareto solutions, i.e., the set of all $x \in X$ which are non-dominated. However, it often is enough to know the objective values of the Pareto solutions. To this end, let

$$f(X) = \left\{ \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} : x \in X \right\}$$

denote the *objective space* of (BP). Then a point $\begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} \in f(X)$ is called *efficient*, if $x \in X$ is a Pareto solution.

For an illustration, see Figure 3.8. In this example let us assume that the set of objective values for all points $x \in X$ is given by the points depicted in the figure. Then the five filled points $p1, \ldots, p5$ are not dominated by any other point, i.e. exactly these points are efficient.

For finding Pareto solutions we can solve a one-criteria optimization problem. This method is called *weighted sum scalarization*.

Theorem 3.15. (e.g. [Ehr00]) If x is an optimal solution of

(BP(λ))

$$\min_{x \in X} \lambda f_1(x) + (1 - \lambda) f_2(x)$$

for some $0 < \lambda < 1$ then x is a Pareto solution of (BP).

Unfortunately, not all Pareto solutions can be found by weighted sum scalarization, if the set $X \subseteq \mathbb{Z}^n$ consists of a discrete set of points. In Figure 3.8, the efficient points $p1, p2$ and $p5$ can be found by solving a weighted sum problem, while no λ exists such that $p3$ and $p4$ are optimal solutions of (BP(λ)).

Definition 3.16. (e.g. [Ehr00]) A Pareto solution x is called *supported* if there exists a λ with $0 < \lambda < 1$ such that x is the optimal solution of (BP(λ)).

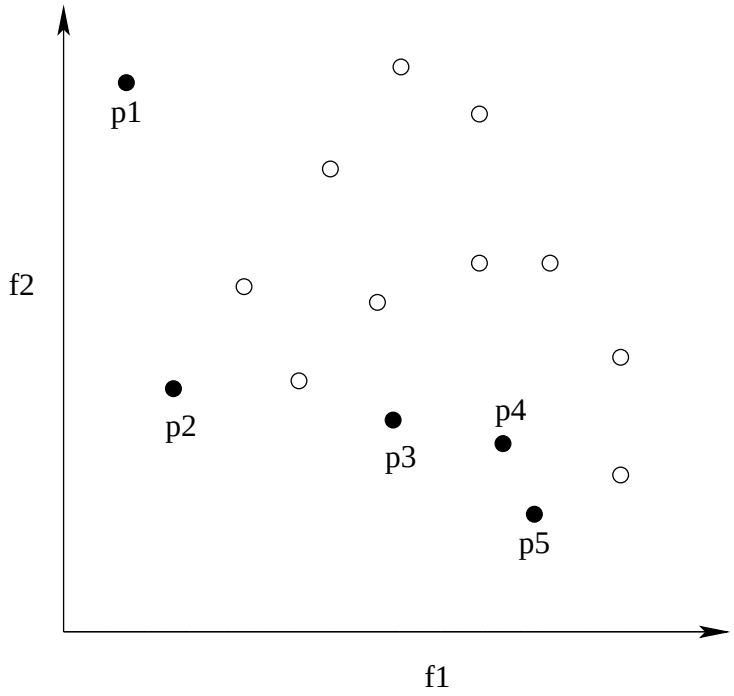

Figure 3.8: Efficient solutions of bicriteria optimization problem.

Note that the name *supported* is due to the following fact: If x is a supported Pareto solution, then $f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix}$ lies on the boundary of the convex hull of $f(X)$. Hence there exists a supporting line of $f(X)$ passing through $f(x)$.

By weighted sum scalarization, we find the set of supported Pareto solutions. With the following result we find also non-supported Pareto solutions. It uses the constraint version of (BP).

Lemma 3.17. ([HC83]) Let $\{i, j\} = \{1, 2\}$ and let x be a unique optimal solution of

$$\min\{f_i(x) : x \in X \text{ and } f_j(x) \leq y_j\}$$

for some y_j. Then x is a Pareto solution of (BP).

Using Lemma 3.17 to find Pareto solutions is known as the *ϵ-constraint method*, see e.g. [Ehr00].

Bicriteria line planning problem

We now reformulate our line planning problem as a bicriteria problem, minimizing both

$$f_{time} := \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x^e_{st}$$

and

$$f_{cost} := \sum_{l\in\mathcal{L}} C_l\ y_l$$

We will present it for one formulation, corresponding to the single criteria formulation (LPMT1) in Section 3.3.2. Obviously we can formulate three alternative bicriteria formulations by changing the coupling constraints (3.5) to (3.6), (3.7) or (3.8).

The bicriteria line planning model with minimal transfers:

(BLPMT1)

$$\min \begin{pmatrix} f_{time} \\ f_{cost} \end{pmatrix} \tag{3.13}$$

$$\begin{aligned} s.t. \quad & \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x^e_{st} \le |\mathcal{R}||\mathcal{E}^l| y_l && \forall\ l \in \mathcal{L} \\ & \theta x_{st} = b_{st} && \forall\ (s,t) \in \mathcal{R} \\ & x^e_{st}, y_l \in \mathbb{B} && \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L} \end{aligned}$$

If we apply the weighted sum scalarization method of theorem 3.15, we get as single objective function

$$\min W_1 f_{time} + W_2 f_{cost}$$

with $0 < W_1 < 1$ and $W_2 = 1 - W_1$. But this means comparing apples and oranges. The problem to find an appropriate weights W_1 and W_2 such that the obtained solution represents the wishes of the decision maker might be very difficult.

Applying the ϵ-constrained method of Lemma 3.17, and setting y_j to the budget B, we get the following one-criteria ϵ-constraint problem resulting from (BLPMT1):

(BLPMT-time)

$$\min f_{time} \tag{3.14}$$

$$\begin{aligned} s.t. \quad & \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l && \forall\ l \in \mathcal{L} \\ & \theta x_{st} = b_{st} && \forall\ (s,t) \in \mathcal{R} \\ & f_{cost} \leq y_{cost} \\ & x_{st}^e, y_l \in \mathbb{B} && \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L} \end{aligned}$$

Due to Lemma 3.17 we have the following result.

Lemma 3.18. Let $(\bar{x}, \bar{y})$ be a unique optimal solution of (BLPMT-time). Then $(\bar{x}, \bar{y})$ is a Pareto solution of (BLPMT). If more than one optimal solution of (BLPMT-time) exists, the solutions that additionally minimize f_{cost} are Pareto solutions.

Unfortunately, (BLPMT-time) is hard to solve.

Corollary 3.19. (BLPMT-time) is NP-hard.

Proof. Setting y_{cost} to the budget B, this model is equal to (LPMT1) which is already shown to be NP-hard. ■

This approach reflects exactly what a decision maker would do intuitively in real world if we would give him a set of Pareto optimal solutions and leave him to choose one of them. It is obvious that the more lines one can establish, the better the solution is for the passengers but on the other hand, the more costly it is for the company. So the decision maker would check how much money the company is willing to spend and would set an upper bound on the second objective. Note that this is exactly, what we do in our single objective formulation (LPMT1).

Since there exist much more powerful solution methods for single objective programming than for multiobjective programming we will treat only the single-objective formulations of (LPMT) in the following. The relaxation of the bicriteria programs by using the budget constraint (3.4) is acceptable since in real world applications a financial budget that must not be exceeded is always given.

3.3.4 Formulation including frequencies

In (LPMT) we implicitly assume that all customers traveling from station s to station t choose the same path in the change&go network, i.e., the same set of lines. This can be done if edge capacities are neglected in (LPMT). In practice, this is usually not the case, since each vehicle only can transport a limited number of customers and usually there is only a limited number of vehicles possible along each line (e.g. due to safety rules). In the following, we therefore present an extension of (LPMT) taking into account the number of vehicles on each line in a given time period. Consequently, this formulation allows to split customers along different paths from s to t in the change&go-network G_{CG}.
Let N denote the capacity of a vehicle and let the new variables $f_l \in \mathbb{N}$ contain the frequency of line l, i.e., the number of vehicles running along line l within a given time period. Furthermore we choose variables $x^e_{st} \in \mathbb{N}$ and change the vector b_{st} to

$$b^i_{st} = \begin{cases} w_{st} & \text{if } i = (s,0) \\ -w_{st} & \text{if } i = (t,0) \\ 0 & \text{else} \end{cases}$$

Then the *Line Planning Model with minimal transfers and frequencies* (LPMTF) is the following:

(LPMTF)

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} c_e \, x^e_{st} \tag{3.15}$$

s.t.

$$\frac{1}{N} \sum_{(s,t)\in\mathcal{R}} x^e_{st} \leq f_l \qquad \forall \, l \in \mathcal{L}, e \in \mathcal{E}_l \tag{3.16}$$

$$\theta x_{st} = b_{st} \qquad \forall \, (s,t) \in \mathcal{R} \tag{3.17}$$

$$\sum_{l\in\mathcal{L}} C_l f_l \leq B \tag{3.18}$$

$$\sum_{l\in\mathcal{L}:k\in\mathcal{E}^l} f_l \leq f^{max}_k \qquad \forall \, k \in E \tag{3.19}$$

$$x^e_{st}, f_l \in \mathbb{N} \qquad \forall \, (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L} \tag{3.20}$$

Constraints (3.16) make sure that the frequency of a line is high enough to transport the passengers. If $f_l = 0$, line l is not chosen in the line concept. Constraints (3.17) are flow conservation constraints routing the passengers through the network. Note that the x^e_{st} variables can take integer values, such that passengers may choose different paths for the same origin-destination pair. Constraint (3.18) is again the budget constraint but with costs for each vehicle of a line (which are multiplied by the frequency to get the costs of the line). The capacity constraint (3.19) may be included if upper bounds for the frequencies are present.

3.4 Discussion of the formulations

3.4.1 Equivalence and strength

In this section we will show that the four formulations of the (LPMT) presented in section 3.3.2 are equivalent and thus will yield the same set of feasible integer solutions. Therefore we first have to repeat some polyhedral theory. For more details the reader is referred to [Wol98] and [NW88].

First we make precise what we mean by a formulation.

Definition 3.20. ([NW88]) Given an integer program

$$\min\{cx : x \in X \subset \mathbb{Z}^n\}$$

where X represents the set of feasible points in $\mathbb{Z}^n$. We say that

$$\min\{cx : Ax \leq b, x \in \mathbb{Z}^n\}$$

is a *valid IP formulation* if $X = \{x \in \mathbb{Z}^n : Ax \leq b\}$.

In general there are many choices of (A, b) and it is usually easy to find some (A, b) that yields one. But an obvious choice may not be a good one when it comes to solving the problem.

Definition 3.21. ([Wol98]) A subset of $\mathbb{R}^n$ described by a finite set of linear constraints $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is a *polyhedron*.

Definition 3.22. ([Wol98]) A polyhedron $P \subseteq \mathbb{R}^n$ is a *formulation* for a set $X \subseteq \mathbb{Z}^n$ if and only if $X = P \cap (\mathbb{Z}^n)$.

Example 3.23. In Figure 3.9 we show two different formulations for the set:

$$X = \{(2,2), (2,3), (3,2), (3,3), (3,4), (4,2)\}$$

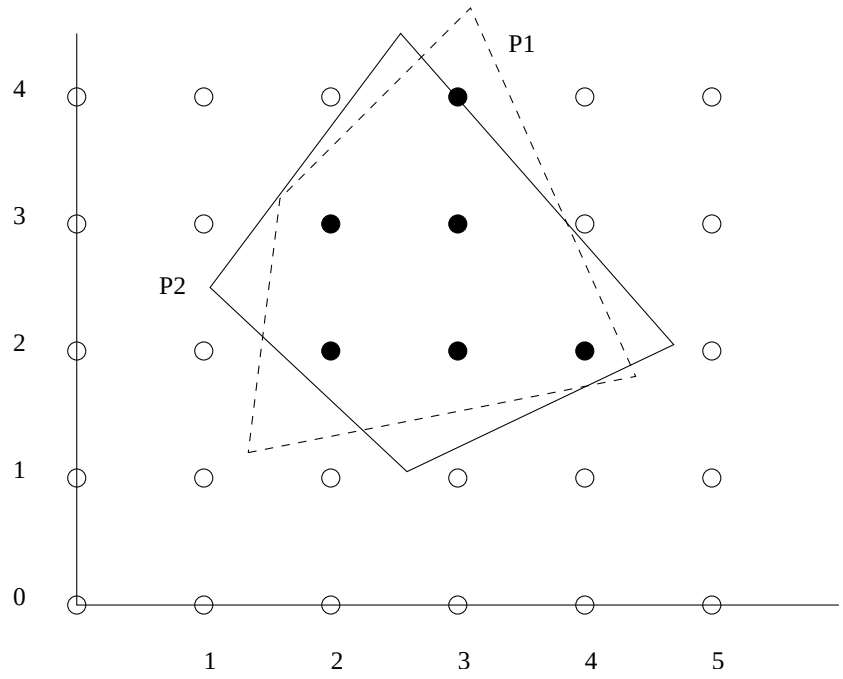

Figure 3.9: Two alternative formulations for the same integer set proposed in Example 3.11.

We will now show that the integer formulations of the (LPMT) presented in Section 3.3.2 are valid IP formulations for the same integer set. Therefore we recall that the integer formulations (3.9), (3.10), (3.11), and (3.12) only differ in the coupling constraints (3.5), (3.6), (3.7), and (3.8).

$$\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x^e_{st} \leq |\mathcal{R}||\mathcal{E}^l| y_l \qquad \forall\, l \in \mathcal{L} \quad (3.5)$$

$$\sum_{(s,t)\in\mathcal{R}} x^e_{st} \leq |\mathcal{R}| y_l \qquad \forall\, l \in \mathcal{L}, e \in \mathcal{E}^l \quad (3.6)$$

$$\sum_{e\in\mathcal{E}^l} x^e_{st} \leq |\mathcal{E}^l| y_l \qquad \forall\, l \in \mathcal{L}, (s,t) \in \mathcal{R} \quad (3.7)$$

$$x^e_{st} \leq y_l \qquad \forall\, e \in cE^l : l \in \mathcal{L}, (s,t) \in \mathcal{R} \quad (3.8)$$

with

$$x^e_{st}, y_l \in \{0,1\}$$

Theorem 3.24. The feasible set of (LPMT4), denoted by X_4, is included in the feasible set of the other formulations: $X_4 \subseteq X_1$, $X_4 \subseteq X_2$, and $X_4 \subseteq X_3$.

Proof. If a point (x, y) satisfies the constraints

$$x_{st}^e \leq y_l \quad \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}^l$$

for all $l \in \mathcal{L}$, then summing up

- over all $(s,t) \in \mathcal{R}$ and $e \in \mathcal{E}^l$ shows that it also satisfies the constraints

$$\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l.$$

  Thus $X_4 \subseteq X_1$.

- over all $(s,t) \in \mathcal{R}$ shows that it also satisfies the constraints

$$\sum_{(s,t)\in\mathcal{R}} x_{st}^e \leq |\mathcal{R}| y_l \quad \forall\ e \in \mathcal{E}^l$$

  and thus $X_4 \subseteq X_2$.

- over all $e \in \mathcal{E}^l$ shows that it also satisfies the constraints

$$\sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{E}^l| y_l \quad \forall\ (s,t) \in \mathcal{R}$$

  and thus $X_4 \subseteq X_3$.

■

Theorem 3.25. (LPMT1), (LPMT2), (LPMT3), and (LPMT4) given in Section 3.3.2 are valid IP formulations of the same integer set X.

Proof. We will show that the integer sets X_1, X_2, and X_3 described by (LPMT1), (LPMT2), and (LPMT3), respectively are equal to the integer set X_4 described by (LPMT4). Given a feasible solution $(\bar{x}, \bar{y})$ of

- (LPMT1), if $\bar{y}_l = 0$ then $\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} \bar{x}_{st}^e = 0$ and thus $\bar{x}_{st}^e = 0$ for all $(s,t) \in \mathcal{R}$, $e \in \mathcal{E}^l : l \in \mathcal{L}$.

- (LPMT2), if $\bar{y}_l = 0$ then $\sum_{(s,t)\in\mathcal{R}} \bar{x}_{st}^e = 0$ for all $e \in \mathcal{E}^l$ and thus $\bar{x}_{st}^e = 0$ for all $(s,t) \in \mathcal{R}$, $e \in \mathcal{E}^l : l \in \mathcal{L}$.

- (LPMT3), if $\bar{y}_l = 0$ then $\sum_{e \in \mathcal{E}^l} \bar{x}^e_{st} = 0$ for all $(s,t) \in \mathcal{R}$ and thus $\bar{x}^e_{st} = 0$ for all $(s,t) \in \mathcal{R}$, $e \in \mathcal{E}^l : l \in \mathcal{L}$.

So, we see that the coupling constraints of (LPMT4) are satisfied in all cases and together with Theorem 3.24, we get that the formulations are equivalent, i.e. $X = X_1 = X_2 = X_3 = X_4$. ■

It is easy to see that there is an infinite number of formulations for each integer problem. But there are some formulations that are "better" than others and there exists also always an "ideal" one, which is in general difficult to find. In the following we will give definitions for "better" and "ideal" and will show, that some of our formulations of the (LPMT) are better than others.

Definition 3.26. ([NW88]) Given a set $P \subseteq \mathbb{R}^n$, a point $x \in \mathbb{R}^n$ is a *convex combination* of points of P if there exists a finite set of points $\{x^1, \ldots, x^t\}$ in P and a $\lambda \in \mathbb{R}^t_+$, with

$$\sum_{i=1}^{t} \lambda_i = 1$$

and

$$x = \sum_{i=1}^{t} \lambda_i x^i$$

.
The *convex hull of* P, denoted $conv(P)$, is the set of all points, that are convex combinations of points in P.

Proposition 3.27. ([Wol98]) $conv(X)$ is a polyhedron.

Definition 3.28. ([NW88]) $x \in Q = \{x \in \mathbb{R}^n : Ax \leq b\}$ is an *extreme point* of Q if there do not exist $x^1, x^2 \in Q$, $x^1 \neq x^2$, such that $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$.

Proposition 3.29. ([Wol98]) The extreme points of $conv(X)$ all lie in X.

Because of these two results, we can replace the IP : $\{\min cx : x \in X\}$ by the equivalent linear program: $\{\min cx : x \in conv(X)\}$. This ideal reduction to a linear program also holds for unbounded integer and mixed integer sets with rational coefficients. The ideal formulation of Example 3.23 is shown in Figure 3.11. As mentioned above this is in general only a theoretical solution, because in most cases there is such an enormous number of inequalities needed to describe $conv(X)$, and there is no simple characterization for them. But how can we compare different formulations? Most integer programming algorithms require a lower bound on the value of the objective function, and

the efficiency of the algorithm is very dependent on the strength of the bound (see Chapter 6). A lower bound is determined by solving the *LP-relaxation*

$$z_{LP} = \{\min cx : Ax \leq b, x \in \mathbb{R}^n\}$$

since $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq X = \{x \in \mathbb{Z}^n : Ax \leq b\}$. Now, given two formulations of X, defined by (A^i, b^i) for $i = 0, 1$, let $P^i = \{x \in \mathbb{R}^n : A^i x \leq b^i\}$ be their formulations and $z^i_{LP} = \{\min cx : A^i x \leq b^i, x \in \mathbb{R}^n\}$ the lower bound provided by their LP-relaxations. Note that if $P^0 \subseteq P^1$, then $z^0_{LP} \geq z^1_{LP}$. Hence, we get the better bound from the formulation based on (A^0, b^0) and we can say that it is the *better* or *stronger* formulation.

Definition 3.30. ([Wol98]) Given a set $X \subseteq \mathbb{R}^n$, and two formulations P_A and P_B for X, P_A is a *stronger formulation* than P_B if $P_A \subset P_B$.

Lemma 3.31. ([NW88]) Given two formulations P_A and P_B for the same integer set X with P_A stronger than P_B, then

$$\min_{x \in X} cx \geq \min_{x \in P_A} cx \geq \min_{x \in P_B} cx.$$

Example 3.32. In Figure 3.10, the formulation $P3$ is better than the formulations $P1$ and $P2$ which are not comparable, but it is still worse than the ideal formulation shown in Figure 3.11.

Notation 3.33. In this section we will denote the feasible set described by the LP-relaxation of (LPMT1) by P_1. We get it by restricting the variables to $0 \leq x^e_{st} \leq 1$ for all $(s,t) \in \mathcal{R}, e \in \mathcal{E}$ and $0 \leq y_l \leq 1$ for all $l \in \mathcal{L}$ instead to $\{0, 1\}$.
The corresponding polyhedra described by (LPMT2), (LPMT3), and (LPMT4) are denoted by P_2, P_3, and P_4, respectively.

Lemma 3.34. Using Notation 3.33, we get $P_4 \subseteq P_1$, $P_4 \subseteq P_2$, and $P_4 \subseteq P_3$.

Proof. If a point (x, y) satisfies the constraints

$$x^e_{st} \leq y_l \quad \forall\ (s,t) \in \mathcal{R}, e \in \mathcal{E}^l$$

for all $l \in \mathcal{L}$, we get that

$$y_l \geq \max_{e \in \mathcal{E}^l} \max_{(s,t) \in \mathcal{R}} x^e_{st}$$

for all $l \in \mathcal{L}$. Summing up

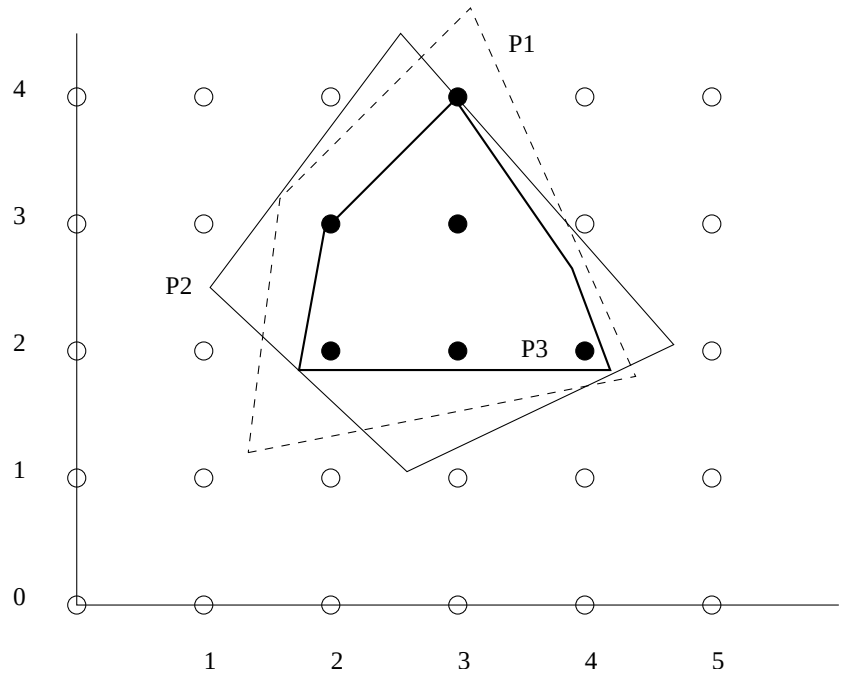

Figure 3.10: P3 is a better formulation of the integer set of Example 3.11 than P1 and P2, but not ideal.

- over all $(s,t) \in \mathcal{R}$ and $e \in \mathcal{E}^l$ shows that it also satisfies the constraints

$$\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l.$$

  Thus $P_4 \subseteq P_1$.

- over all $(s,t) \in \mathcal{R}$ shows that it also satisfies the constraints

$$\sum_{(s,t)\in\mathcal{R}} x_{st}^e \leq |\mathcal{R}| y_l \quad \forall\ e \in \mathcal{E}^l$$

  and thus $P_4 \subseteq P_2$.

- over all $e \in \mathcal{E}^l$ shows that it also satisfies the constraints

$$\sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{E}^l| y_l \quad \forall\ (s,t) \in \mathcal{R}$$

  and thus $P_4 \subseteq P_3$.

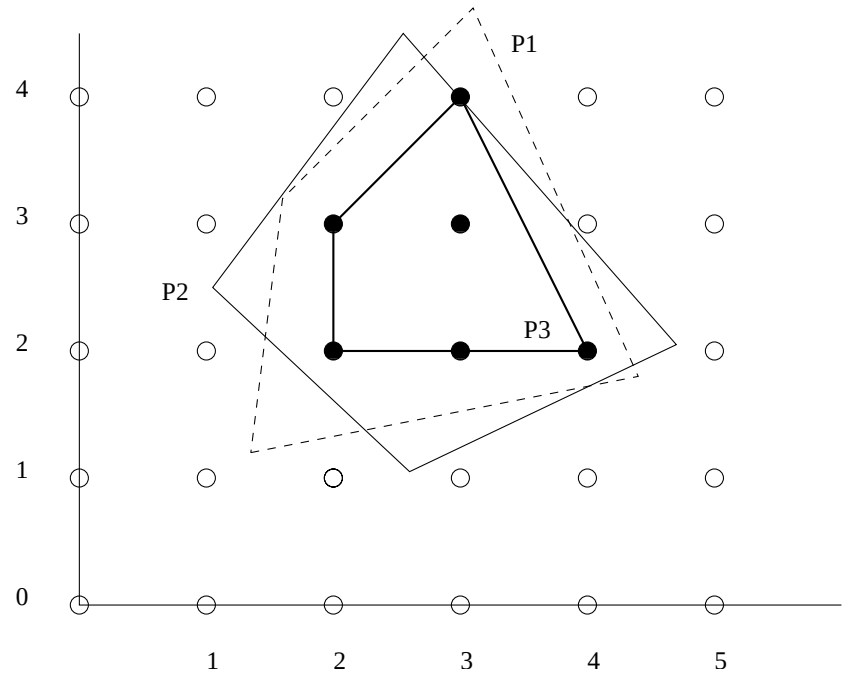

Figure 3.11: P3 is the ideal formulation of the integer set of Example 3.11.

■

We will now discuss the strength of our formulations. We will demonstrate the results on Example 3.11. Figure 3.12 recalls the corresponding change&go-network and names the driving edges $e_1, \ldots, e_4$. Let the set of origin-destination pairs be $\mathcal{R} = \{(1,2),(1,3)\}$. Figure 3.13 and 3.14 show two feasible path-combinations for the two customers. We will call them *alternative A* and *alternative B*. The corresponding feasible point reads like follows: if edge e_i is depicted in the figure for customer j, then $x_j^{e_i} = 1$ and zero otherwise. The y_l will be explained in the proofs.

Theorem 3.35. P_4 is stronger than P_1.

Proof. In Lemma 3.34 we have shown that $P_4 \subseteq P_1$.
To show that $P_4 \subset P_1$, we need to find a point in P_1 that is not in P_4.
In our example, the solution $(\bar{x}, \bar{y})$ with the $\bar{x}$-values of alternative A (Figure 3.13) and $\bar{y} = (\frac{1}{2}, 0, \frac{1}{2})$ is a feasible point in P_1 but not in P_4 since e.g. constraint $\bar{x}_1^{e_1} \leq \bar{y}_1$ is violated.
More general, a solution in which not all driving edges of a line are used by all origin-destination pairs, i.e. $\exists e^* \in \mathcal{E}^{l^*}$ such that $\bar{x}_{ab}^{e^*} = 0$ for some $(a,b) \in \mathcal{R}$,

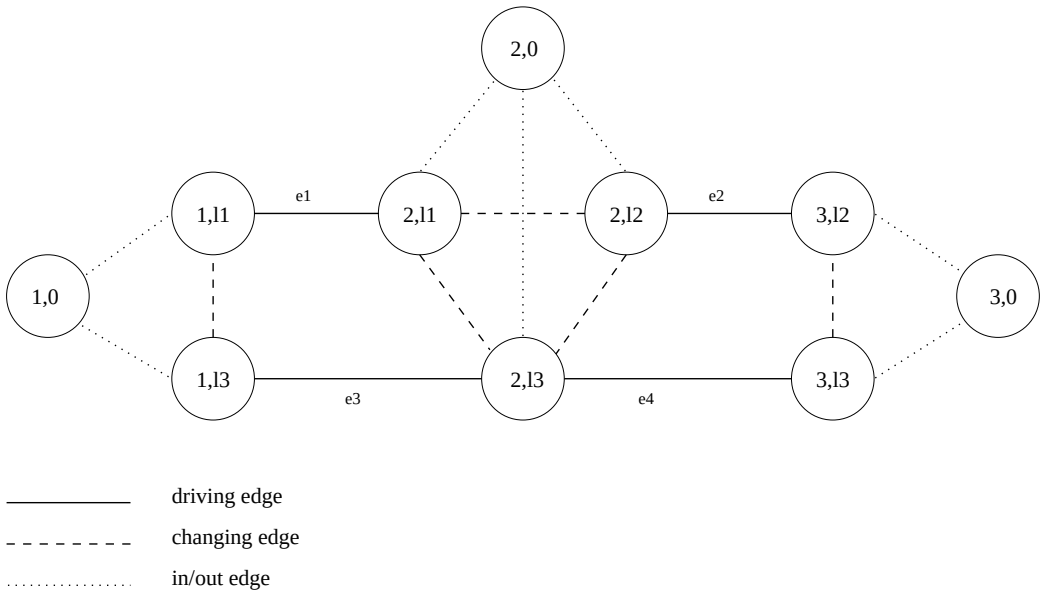

Figure 3.12: The change&go graph of Example 3.11.

and

$$\bar{y}_l := \frac{\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} \bar{x}^e_{st}}{|\mathcal{R}||\mathcal{E}^l|}$$

for all $l \in \mathcal{L}$ lies in $P_1 \setminus P_4$. ■

Theorem 3.36. P_4 is stronger than P_2.

Proof. In Lemma 3.34 we have shown that $P_4 \subseteq P_2$.
To show that $P_4 \subset P_2$, we need to find a point in P_2 that is not in P_4.
In our example, the solution $(\bar{x}, \bar{y})$ with the $\bar{x}$-values of alternative A (Figure 3.13) and $\bar{y} = (\frac{1}{2}, 0, \frac{1}{2})$ is a feasible point in P_2 but not in P_4 since e.g. constraint $\bar{x}^{e_1}_1 \leq \bar{y}_1$ is violated.
More general, a solution $(\bar{x}, \bar{y})$ in which one line is not used at all by one customer, i.e. $\exists (a,b) \in \mathcal{R}, l^* \in \mathcal{L}$ such that $\bar{x}^e_{ab} = 0$ for all $e \in \mathcal{E}^{l^*}$ but there is another edge of this line used by some customer, i.e. $\exists (c,d) \in \mathcal{R}, (c,d) \neq (a,b)$ such that $\bar{x}^e_{cd} = 1$ for some $e \in \mathcal{E}^{l^*}$ and

$$\bar{y}_l = \max_{e\in\mathcal{E}^l} \frac{\sum_{(s,t)\in\mathcal{R}} \bar{x}^e_{st}}{|\mathcal{R}|}$$

for all $l \in \mathcal{L}$ lies in $P_2 \setminus P_4$. ■

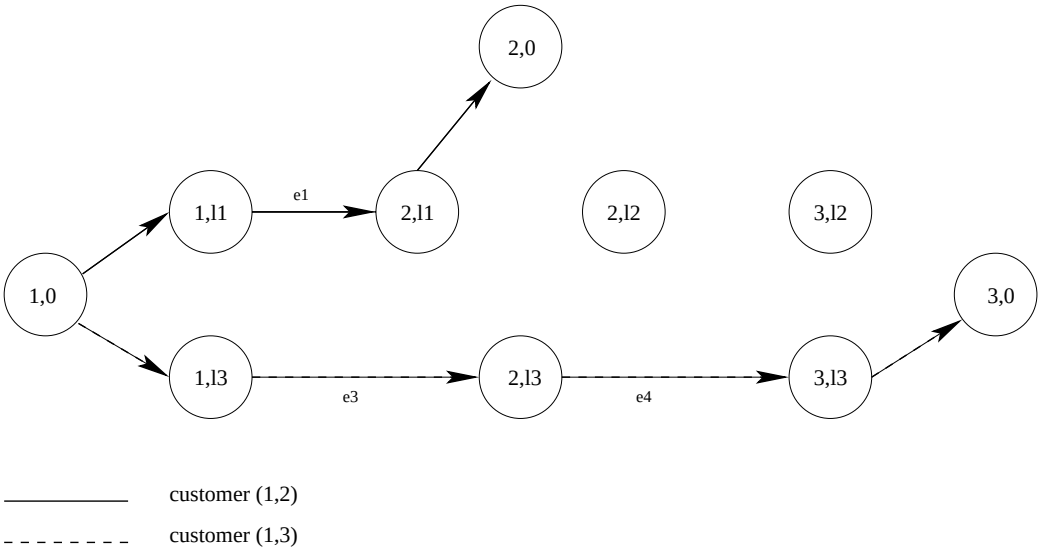

Figure 3.13: path combination: alternative A.

Theorem 3.37. P_4 is stronger than P_3.

Proof. In Lemma 3.34 we have shown that $P_4 \subseteq P_3$.
To show that $P_4 \subset P_3$, we need to find a point in P_3 that is not in P_4.
In our example, the solution $(\bar{x}, \bar{y})$ with the $\bar{x}$-values of alternative B (Figure 3.14) and $\bar{y} = (0, 1, \frac{1}{2})$ is a feasible point in P_3 but not in P_4 since e.g. constraint $\bar{x}_1^{e_3} \leq \bar{y}_3$ is violated.
More general, a solution $(\bar{x}, \bar{y})$ in which there is one edge of a line that is not used by any customer, i.e. $\exists\, e^* \in l^*$ such that $\bar{x}_{st}^{e^*} = 0$ for all $(s,t) \in \mathcal{R}$ but there is another edge of this line used by some customer, i.e. $\exists\, \hat{e} \in \mathcal{E}^{l^*}, \hat{e} \neq e^*$ such that $\bar{x}_{st}^{\hat{e}} = 1$ for some $(s,t) \in \mathcal{R}$, and

$$\bar{y}_l = \max_{(s,t)\in\mathcal{R}} \frac{\sum_{e\in\mathcal{E}^l} \bar{x}_{st}^e}{|\mathcal{E}^l|}$$

for all $l \in \mathcal{L}$ lies in $P_3 \setminus P_4$. ■

Theorem 3.38. P_2 is stronger than P_1.

Proof. If a point (x, y) satisfies the constraints

$$\sum_{(s,t)\in\mathcal{R}} x_{st}^e \leq |\mathcal{R}| y_l \quad \forall\, e \in \mathcal{E}^l,$$

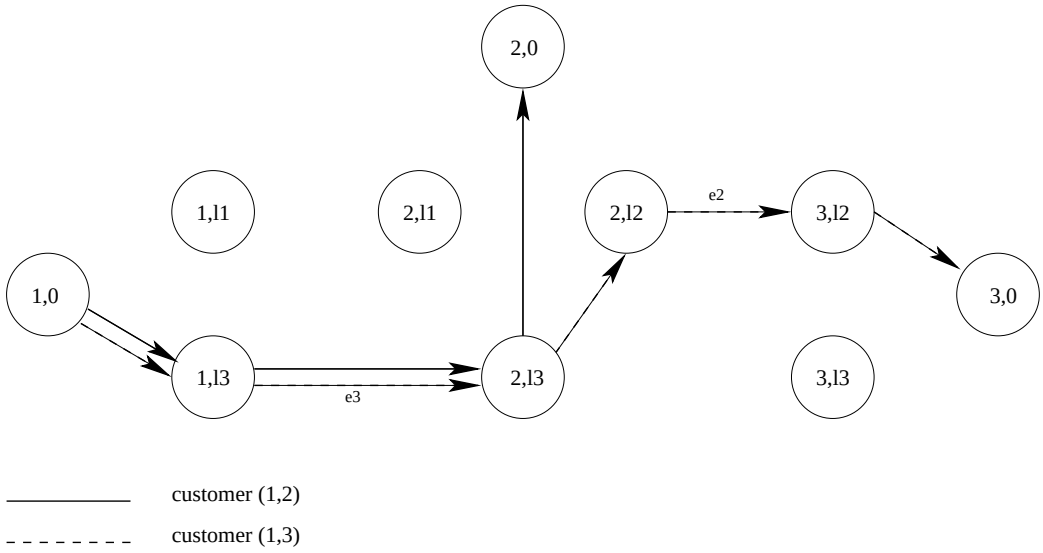

Figure 3.14: path combination: alternative B.

then summing up over $e \in \mathcal{E}^l$ shows that it also satisfies the constraints

$$\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l$$

and thus $P_2 \subseteq P_1$.
To show that $P_2 \subset P_1$, we need to find a point in P_1 that is not in P_2.
In our example, the solution $(\bar{x}, \bar{y})$ with the $\bar{x}$-values of alternative B (Figure 3.14) and $\bar{y} = (0, \frac{1}{2}, \frac{1}{2})$ is a feasible point in P_1 but not in P_2 since e.g. constraint $\sum_{(s,t)\in\mathcal{R}} \bar{x}_{st}^{e_3} \leq 2 \cdot \bar{y}_3$ is violated.
More general, a solution $(\bar{x}, \bar{y})$ in which there is an edge that is not used by all customers, i.e. $\exists e^* \in l^*$ such that $\bar{x}_{st}^{e^*} = 0$ for some $(s,t) \in \mathcal{R}$ but there is an edge of this line that is used by all customers, i.e. $\exists \hat{e} \in \mathcal{E}^{l^*}, \hat{e} \neq e^*$ such that $\bar{x}_{st}^{\hat{e}} = 1$ for all $(s,t) \in \mathcal{R}$, and

$$\bar{y}_l := \frac{\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} \bar{x}_{st}^e}{|\mathcal{R}||\mathcal{E}^l|}$$

for all $l \in \mathcal{L}$ lies in $P_1 \setminus P_2$. ■

Theorem 3.39. P_3 is stronger than P_1.

Proof. If a point (x, y) satisfies the constraints

$$\sum_{e \in \mathcal{E}^l} x_{st}^e \leq |\mathcal{E}^l| y_l \quad \forall\ (s,t) \in \mathcal{R},$$

then summing over up $(s,t) \in \mathcal{R}$ shows that it also satisfies the constraints

$$\sum_{(s,t) \in \mathcal{R}} \sum_{e \in \mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l$$

and thus $P_3 \subseteq P_1$.
To show that $P_3 \subset P_1$, we need to find a point in P_1 that is not in P_3.
In our example, the solution $(\bar{x}, \bar{y})$ with the $\bar{x}$-values of alternative A (Figure 3.13) and $\bar{y} = (\frac{1}{2}, 0, \frac{1}{2})$ is a feasible point in P_1 but not in P_3 since e.g. constraint $\sum_{e \in \mathcal{E}^1} \bar{x}_1^e \leq 1 \cdot \bar{y}_1$ is violated.
More general, a solution $(\bar{x}, \bar{y})$ in which there is an edge that is not used by all customers, i.e. $\exists e \in \mathcal{E}^{l^*}$ such that $\bar{x}_{st}^e = 0$ for some $(s,t) \in \mathcal{R}$ but there is a customer that uses all edges of a line, i.e. $\exists (a,b) \in \mathcal{R}$ such that $\bar{x}_{ab}^e = 1$ for all $e \in \mathcal{E}^{l^*}$, and

$$\bar{y}_l = \frac{\sum_{(s,t) \in \mathcal{R}} \sum_{e \in \mathcal{E}^l} \bar{x}_{st}^e}{|\mathcal{R}||\mathcal{E}^l|}$$

for all $l \in \mathcal{L}$ lies in $P_1 \setminus P_3$. ■

Theorem 3.40. Comparing P_2 and P_3, none of them is stronger than the other.

Proof. We can find a point, that is in P_2 but not in P_3 and vice versa.
In our example, the solution $(\bar{x}, \bar{y})$ with the $\bar{x}$-values of alternative B (Figure 3.14) and $\bar{y} = (0, \frac{1}{2}, 1)$ is a feasible point in P_2 but not in P_3 since e.g. constraint $\sum_{e \in \mathcal{E}^2} \bar{x}_3^e \leq 1 \cdot \bar{y}_2$ is violated.
More general, a solution $(\bar{x}, \bar{y})$ in which there is a customer that uses all edges of a line, i.e. $\exists (a,b) \in \mathcal{R}$ such that $\bar{x}_{ab}^e = 1$ for all $e \in l^*$ but there is no edge of this line that is used by all customers, i.e. $\bar{x}_{st}^e = 0$ for some $(s,t) \in \mathcal{R}$ for all $e \in \mathcal{E}^{l^*}$, and

$$\sum_{(s,t) \in \mathcal{R}} \bar{x}_{st}^e = k < |\mathcal{R}|$$

and

$$\bar{y}_l = \max_{e \in \mathcal{E}^l} \frac{\sum_{(s,t) \in \mathcal{R}} \bar{x}_{st}^e}{|\mathcal{R}|}$$

for all $l \in \mathcal{L}$ lies in $P_2 \setminus P_3$.

On the other hand, in our example, the solution $(\bar{x}, \bar{y})$ with the $\bar{x}$-values of alternative B (Figure 3.14) and $\bar{y} = (0, 1, \frac{1}{2})$ is a feasible point in P_3 but not in P_2 since e.g. constraint $\sum_{(s,t)\in\mathcal{R}} \bar{x}_{st}^{e_3} \leq 2 \cdot \bar{y}_3$ is violated.
More general, a solution $(\bar{x}, \bar{y})$ in which there is an edge that is used by all customers, i.e. $\exists e^* \in \mathcal{E}^{l^*}$ such that $\bar{x}_{st}^{e^*} = 1$ for all $(s,t) \in \mathcal{R}$ but there is a customer that does not use all edges of this line, i.e. $\exists (a,b) \in \mathcal{R}$ such that $\bar{x}_{ab}^{e} = 0$ for some $e \in \mathcal{E}^{l^*}, e \neq e^*$, and

$$\bar{y}_l := \max_{(s,t)\in\mathcal{R}} \frac{\sum_{e\in\mathcal{E}^l} \bar{x}_{st}^{e}}{|\mathcal{E}^l|}$$

for all $l \in \mathcal{L}$ lies in $P_3 \setminus P_2$. ■

Remark 3.41. Note that in real world instances (LPMT3) comes out to be much better than (LPMT2) (see Table 3.5). Unfortunately, due to its huge amount of constraints it is only computable for smaller instances (see Table 3.4).

Figure 3.15 shows schematically the results of this section. All formulations describe the same integer set but the (LPMT4) formulation is better than all others. (LPMT1) is the worst and (LPMT2) and (LPMT3) are not comparable but better than (LPMT1) and worse than (LPMT4). Later we will see that in real world instances (LPMT3) figures out to be better than (LPMT2) in most cases (see Section 3.4.3).

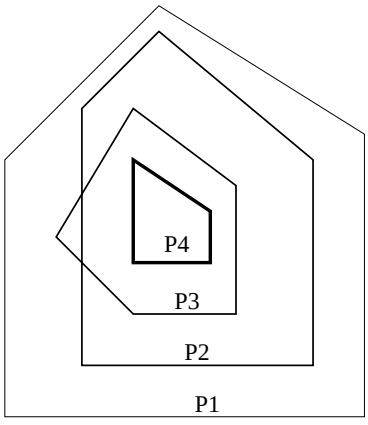

Figure 3.15: Schematical summary of the results of Section 3.4.

3.4.2 Model structure

Integer programs and their associated linear relaxations encountered in applications almost always exhibit a large deal of structure. Constraint matrices are typically very *sparse*, having non zero elements in the order of magnitude of one percent, or less. This well known phenomenon is due to the fact that activities associated with variables are subject to only a few of the conditions represented by the constraints, respectively. On top of that, there may exist a hierarchical, geographical or logical segmentation of the underlying problem, which is reflected in the model formulation. Thus, it is likely that the non zeros are grouped in such a way that *independent subsystems* of variables and constraints result, possibly *linked* by a distinct set of constraints and/or variables. Figure 3.16 shows schematically the distribution of non zeros for different angular block diagonal matrices, occurring in practice most frequently ([Min86]).

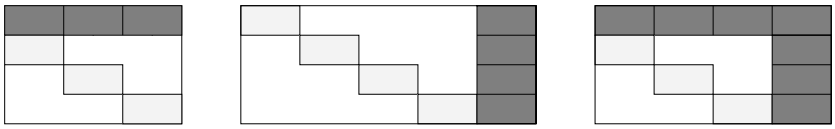

Figure 3.16: Block diagonal matrix structure, (a) with coupling constraints, (b) with coupling variables, and (c) with coupling constraints and variables. Only the shaded regions may contain non-zero elements.

All our model formulations presented in Section 3.3.2 have block diagonal structure with only a few coupling constraints and all blocks (except the single budget constraint) are totally unimodular.

The block structure of the model (LPMT1) is shown in the following scheme.

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st} c_e x^e_{st}$$

$$\boxed{\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x^e_{st} \le |\mathcal{R}||\mathcal{E}^l| y_l} \qquad \text{coupling constraints} \qquad (3.5)$$

$$\boxed{\theta x_{s_1 t_1} = b_{s_1 t_1}}$$

$$\ddots \qquad (3.3)$$

$$\boxed{\theta x_{s_{|\mathcal{R}|}, t_{|\mathcal{R}|}} = b_{s_{|\mathcal{R}|}, t_{|\mathcal{R}|}}}$$

$$\boxed{\sum_{l\in\mathcal{L}} C_l y_l \le B} \qquad (3.4)$$

Note that the other formulations (LPMT2), (LPMT3), and (LPMT4) only differ in the coupling constraints (3.6), (3.7), and (3.8) and thus the structure is equivalent.
The shortest path constraint set (3.3) decomposes in $|\mathcal{R}|$ many independent blocks: one for each origin-destination pair $(s,t) \in \mathcal{R}$. These shortest-path-blocks are identical on the left-hand side and differ only in the right-hand side b_{st}. We get one more independent block for the budget constraint (3.4). The constraints (3.5) are *coupling constraints*.

Since the change&go-network and hence the corresponding problem formulations get huge, it is not possible to compute the optimal solution of the LP-relaxation of the different formulations using standard optimization software. Table 3.4 shows the CPU times for some real world instances explained in Table 3.2 for the different formulations. The results are based on a 3.06 GHz Intel4 processor with 2 GB RAM using Xpress IVE Version 1.15.01. The size of (LPMT) depends on two criteria: the size of the line pool and the number of origin-destination pairs. Table 3.3 shows computation times of the (LPMT1) relaxation for different line pool sizes and origin-destination sets.
Even though the LP-relaxation of (LPMT) is not solvable for real world instances, we can exploit its nice structure to decompose the problem into many small (solvable) subproblems linked by only few coupling constraints. In Section 5.2 we will present a solution approach based on Dantzig-Wolfe decomposition to solve the LP-relaxation of our problem. Note that in this case the subproblems can also be computed on parallel machines since they are independent.

The size of (LPMT) depends on two criteria:

1. the size of the line pool: the change&go-network is constructed out of the line pool and thus the size of the node-arc incidence matrix θ increases with with the line pool size

2. the number of origin-destination pairs: we solve a shortest path problem for each origin-destination pair and so the number of constraints increases if we treat more origin-destination pairs

Table 3.3 shows the CPU times for the LP-relaxation of (LPMT1) for instances with different sizes of line pools $|\mathcal{L}|$ and sets of origin-destination pairs $|\mathcal{R}|$. If the problem was not solvable due to lack of memory, this is indicated by "M".

$\|\mathcal{L}\|$	$\|\mathcal{R}\|$=2	$\|\mathcal{R}\|$=10	$\|\mathcal{R}\|$=50	$\|\mathcal{R}\|$=100	$\|\mathcal{R}\|$=150	$\|\mathcal{R}\|$=200	$\|\mathcal{R}\|$=1476
10	0.187	0.860	4.219	8.563	12.469	16.453	154.281
50	4.875	23.610	118.844	239.782	361.063	M	M
100	25.172	124.359	626.640	M	M	M	M
132	52.047	257.906	M	M	M	M	M
150	M	M	M	M	M	M	M

Table 3.3: CPU times for the LP-relaxation of (LPMT1) for different line pool sizes and origin-destination sets. "'M"' denotes out of memory.

Instance	$\|\mathcal{L}\|$	Budget	(LPMT1)	(LPMT2)	(LPMT3)	(LPMT4)
0	3	1	0.03	0.03	0.03	0.03
1	10	7	154.8	146.2	153.3	M
2	50	30	M	M	M	M

Table 3.4: CPU times for the LP-relaxation of the four formulations using XpressMP, "M" denotes out of memory)

In the next section we will present a method to check if the trivial solution, i.e. the solution we get, if we compute the shortest path problem on the change&go-network constructed of all lines of the line pool, is already an optimal solution of the LP-relaxation. Depending on the formulation (LPMT1), (LPMT2), (LPMT3), and (LPMT4), this may appear quite often.

3.4.3 Special cases

In some special cases the solution of the LP-relaxation of the (LPMT1) is easy to find.

Proposition 3.42. If $C_l = 1 \; \forall \; l \in \mathcal{L}$ and if for each origin-destination pair $(s,t) \in \mathcal{R}$ there is a line $l \in \mathcal{L}$ that serves (s,t) directly on shortest path, then the solution that chooses exactly this special line for each origin-destination pair is always feasible and optimal for the LP-relaxation of (LPMT1).

Proof. Let $\bar{x}^e_{st}$ be the optimal solution of the shortest path problem on the change&go-network constructed of all lines of the line pool and

$$\bar{y}_l := \frac{\sum_{(s,t)\in\mathcal{R}} k^l_{st}}{|\mathcal{R}||\mathcal{E}^l|}$$

for all $l \in \mathcal{L}$ with $k^l_{st} := \sum_{e\in\mathcal{E}^l} \bar{x}^e_{st}$. Note that $k^l_{st} \leq |\mathcal{E}^l|$ for all $(s,t) \in \mathcal{R}, l \in \mathcal{L}$. Since by assumption each passenger can travel directly on one line, the

number of non-zero k_{st}^l is equal $|\mathcal{R}|$.
This solution $(\bar{x}, \bar{y})$ satisfies the coupling constraints of (LPMT1) (3.5).
It also satisfies the budget constraint (3.4)

$$\sum_{l\in\mathcal{L}} C_l \bar{y}_l = \frac{1}{|\mathcal{R}|} \underbrace{\sum_{l\in\mathcal{L}} \sum_{(s,t)\in\mathcal{R}} \overbrace{\frac{k_{st}^l}{|\mathcal{E}^l|}}^{\leq 1}}_{\leq |\mathcal{R}|} \leq 1$$

independently of the choice of the budget since a budget smaller than one does not make sense if $C_l = 1$ for all $l \in \mathcal{L}$. So, the solution $(\bar{x}, \bar{y})$ is feasible for (LPMT1). It is also optimal since there is no better way than traveling on shortest path without transfer. ■

Definition 3.43. A *trivial solution* $(\bar{x}, \bar{y}^1)$, $(\bar{x}, \bar{y}^2)$, $(\bar{x}, \bar{y}^3)$, $(\bar{x}, \bar{y}^4)$ of (LPMT1), (LPMT2), (LPMT3), (LPMT4), respectively, is defined as the solution of the shortest path problem on the change&go-network constructed of all lines of the line pool $\bar{x}_{st}^e$ and

$$\bar{y}_l^1 := \frac{\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} \bar{x}_{st}^e}{|\mathcal{E}^l||\mathcal{R}|} \quad \forall\, l \in \mathcal{L} \quad \text{(for (LPMT1))}$$

$$\bar{y}_l^2 := \frac{\max_{e\in\mathcal{E}^l} \sum_{(s,t)\in\mathcal{R}} \bar{x}_{st}^e}{|\mathcal{R}|} \quad \forall\, l \in \mathcal{L} \quad \text{(for (LPMT2))}$$

$$\bar{y}_l^3 := \frac{\max_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} \bar{x}_{st}^e}{|\mathcal{E}^l|} \quad \forall\, l \in \mathcal{L} \quad \text{(for (LPMT3))}$$

$$\bar{y}_l^4 := \max_{(s,t)\in\mathcal{R}} \max_{e\in\mathcal{E}^l} \bar{x}_{st}^e \quad \forall\, l \in \mathcal{L} \quad \text{(for (LPMT4))}$$

It is in general not unique and need not to be feasible in the sense that it fulfills the budget constraint.

In real world instances it appears quite often that a trivial solution is an optimal solution of the LP-relaxation of (LPMT1) even if not all assumptions of Proposition 3.42 are fulfilled. This is clear since the right hand sides $|\mathcal{R}||\mathcal{E}^l|$ of the coupling constraints (3.5) are chosen such that all passengers could use all edges of all lines. In real world only few edges of the network are used and so $K_l := \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e$ is much smaller than $|\mathcal{R}||\mathcal{E}^l|$, hence

$$\sum_{l\in\mathcal{L}} C_l \bar{y}_l^1 = \sum_{l\in\mathcal{L}} C_l \frac{K_l}{|\mathcal{E}^l||\mathcal{R}|} \leq B$$

is often satisfied.

No.	$\lvert\mathcal{L}\rvert$	obj.val.	T1	T2	T3	T4	CPU
1	10	2271.3	0.69	0.99	9.53	10	0
2	50	9459.9	0.20	0.35	25.31	48	10
3	100	24780.0	0.13	0.29	41.83	96	60
4	132	31654.2	0.11	0.26	53.12	129	131
5	200	15128.9	0.07	0.19	54.89	197	342
6	250	19096.0	0.05	0.16	61.07	235	418
7	275	20118.2	0.04	0.15	63.47	252	579
8	300	26598.3	0.06	0.19	72.35	282	811
9	330	26817.7	0.04	0.16	74.44	302	1052
10	350	26450.0	0.07	0.23	90.04	331	966
11	375	27517.8	0.06	0.20	90.75	345	1215
12	400	34781.3	0.06	0.20	100.05	370	1571
13	423	35135.5	0.06	0.20	102.19	389	1972

Table 3.5: Minimal budgets such that trivial solution is optimal solution of the LP-relaxation of the different formulations of the (LPMT), see Lemma 3.44. CPU times in seconds.

Lemma 3.44. Let $i \in \{1, 2, 3, 4\}$ and let $(\bar{x}, \bar{y}^i)$ be a trivial solution of (LPMTi), as defined in Definition 3.43. If

$$Ti := \sum_{l \in \mathcal{L}} C_l \bar{y}^i_l \leq B$$

is satisfied, the trivial solution $(\bar{x}, \bar{y}^i)$ is an optimal solution of the (LPMTi). Note that for $i = 4$ the solution $(\bar{x}, \bar{y}^4)$ of the LP-relaxation of (LPMT4) is integer and thus if $T4 \leq B$ holds, the trivial solution is an optimal solution to the original problem.

Proof. Clear by the formulation of (LMPT1), (LMPT2),(LPMT3), and (LMPT4). ■

Table 3.5 shows some computational results of real world instances of Lemma 3.44 with line costs $C_l = 1 \; \forall \; l \in \mathcal{L}$. T1, T2, T3 and T4 represent the minimal budgets such that the trivial solution still is the optimal solution of the LP-relaxation of the corresponding formulation. This means, if the given budget is smaller than the Ti-value, the trivial solution does not fulfill the budget constraint and we have to use other methods to find a solution of the LP-relaxation. Our approach is a Dantzig-Wolfe decomposition approach and will be presented in Chapter 5.

If the given budget is higher than the Ti-value, it is sufficient to solve the shortest path problem which can be done in polynomial time. Even if it seems simple, this result is very useful for the computational realization of a solution method, as we will see in Chapter 6.
If we have a closer look on the Ti-values, we see that the T1- and the T2-values in Table 3.5 are always smaller than one, so an optimal LP-relaxation solution of (LPMT1) and (LPMT2) is for all instances and for all budgets (remember that a budget smaller than one does not make sense) equal to the trivial solution. One might think that this is due to the fact that the line costs are all set to one. The budget then represents the maximum number of lines allowed to be chosen into the line concept. But the following example will show that even if the line costs are set to one, there may exists a PTN and a line pool such that the T1- and the T2-value is bigger than 1.

Example 3.45. We consider a PTN consisting of four nodes $S := \{1, 2, 3, 4\}$ and three edges $E := \{(1, 2), (2, 3), (3, 4)\}$. The line pool contains three lines $\mathcal{L} := \{l1, l2, l3\} := \{(1, 2), (2, 3, 4), (1, 2, 3, 4)\}$ where line $l3$ is a regional (slow) train and $l1$ and $l2$ are fast trains. The line costs are all equal to one. The time the trains need from one station to another station is shown in Figure 3.17.

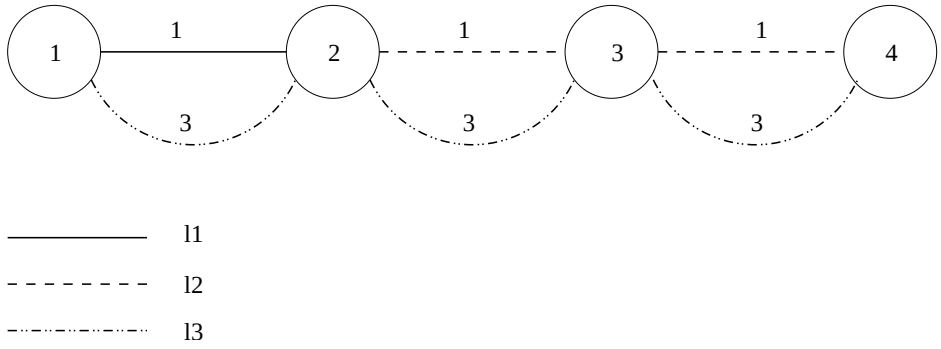

Figure 3.17: Line pool of Example 3.45.

We have three origin-destination pairs $\mathcal{R} := \{(1, 2), (1, 4), (2, 4)\}$. The edge costs are set to the driving times on the driving-edges and to 4 on the changing-edges. Obviously the solution of the shortest path problem if we allow all lines of the line pool is: origin-destination pair $(1, 2)$ uses line $l1$ without transfer, origin-destination pair $(2, 4)$ uses line $l2$ without transfer, and origin-destination pair $(1, 4)$ uses line $l1$ and line $l2$ with transfer at station 2.

In this case, we get the following Ti-values as defined in Lemma 3.44:

$$T1 = \frac{2}{3} + \frac{4}{6} + \frac{0}{9} = \frac{4}{3}$$
$$T2 = \frac{2}{3} + \frac{2}{3} + \frac{0}{3} = \frac{4}{3}$$
$$T3 = \frac{1}{1} + \frac{2}{2} + \frac{0}{3} = 2$$
$$T4 = 1 + 1 + 0 = 2$$

Thus, choosing line $l1$ and line $l2$ would be the optimal solution if the budget is set to 2 or 3. If it is set to 1, the optimal solution is the line concept $L = \{l3\}$ because it is the only line that serves all customers.

The T3-values are much "better" than the T1- and the T2-values in the sense that the solution of the LP-relaxation of (LPMT3) is not equal to the trivial solution if the given budget is smaller than the relatively high T3-value, and hence is stronger. So, in practice we can say, that the (LPMT3) formulation is stronger than the (LPMT2) formulation even if we have shown in Theorem 3.40 that they are not comparable. But this means only that there exists a case in which one formulation is stronger than the other and vice versa. This case is mentioned in the proof of the theorem. It uses the fact that $\bar{y}_l^2$ is near one if there is an edge of a line l that is used by almost all customers which hardly ever appears in real world instances. On the other hand $\bar{y}_l^3$ is near one if almost all edges of a line l are used by some customer, which is much more likely since there may be a customer that uses a line from the start- to the end-node and by doing so, he reaches his destination without transfer. Unfortunately the (LPMT3) formulation also has much more constraints than (LPMT2) and so the solution of the LP-relaxation is computationally hard for larger instances (see Table 3.4).
As we have shown in Section 3.4 the (LPMT4) formulation yields the best Ti-values of all formulations. Calculating its LP-relaxation would rise up to a highly integer solution but unfortunately it has too many constraints to solve even small instances with standard LP solvers. Even the resulting master problem in the Dantzig-Wolfe approach presented in Chapter 5 gets too big, so it can only be used for heuristic approaches.

Part II

Solution Methods

Chapter 4

Heuristics

As we have seen in Section 3.2 our problem belongs to the class of NP-hard problems. That is, there is strong evidence that no efficient algorithms for its solution exist. Fortunately our integer formulations of (LPMT) have a nice structure. All computational results presented in this section are computed on a Pentium 4 processor with 3Ghz and 2GB RAM. The heuristics are implemented using Visual C++ 6.0. Shortest path subproblems are calculated using the Dijkstra algorithm.

Notation 4.1. In this section we denote the edge set of a change&go-network constructed by the lines of a given line concept $L \subseteq \mathcal{L}$ by

$$\mathcal{E}(L) := \cup_{l \in L} \mathcal{E}^l \cup \mathcal{E}_{change} \cup \mathcal{E}_{OD}.$$

Furthermore, to simplify the notation, we will call the origin-destination pairs $r \in \mathcal{R}$ instead of $(s,t) \in \mathcal{R}$.

4.1 Variable Fixing

As we mentioned in Section 3.3.2, the (LPMT4) formulation (3.12) is the strongest among our formulations and so we will use it as a basis of various heuristics presented in this chapter.

Therefore we first recall (LPMT4). We remember that the formulation consists of the objective function (3.2), the coupling constraints (3.8), the shortest path problems (3.3) and the budget constraint (3.4).

(LPMT4)

$$\min \sum_{r \in \mathcal{R}} \sum_{e \in \mathcal{E}} w_r \; c_e \; x_r^e \qquad (3.2)$$

$$\begin{aligned}
s.t.\; & x_r^e \leq y_l && \forall\, r \in \mathcal{R}, e \in \mathcal{E}^l : l \in \mathcal{L} && (3.8)\\
& \sum_{e \in \mathcal{E}} \theta_{ie} x_r^e = b_r^i && \forall\, r \in \mathcal{R}, i \in \mathcal{V} && (3.3)\\
& \sum_{l \in \mathcal{L}} C_l y_l \leq B && && (3.4)\\
& x_r^e, y_l \in \mathbb{B} && \forall\, r \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L} &&
\end{aligned}$$

We can divide the variables of the model into two groups: the x-variables and the y-variables. The x-variables correspond to the choice of the passenger paths and the y-variables correspond to lines chosen to be in the line concept.

What happens if we fix one of the variable groups?

1. Fixing the y-variables:
   This means, that we assume the line concept L to be known. The budget-constraint thus reduces to

   $$\sum_{l \in L} C_l \leq B$$

   which does not contain any variables and only makes sure that the chosen line concept does not exceed the given budget and thus is feasible. The coupling constraints fix the x_r^e to zero for all $e \in \mathcal{E}_{go} \setminus \mathcal{E}(L)$ and thus the shortest path constraints reduce to

   $$\sum_{e \in \mathcal{E}(L)} \theta_{ie} x_r^e = b_r^i \quad \forall\, r \in \mathcal{R}, i \in \mathcal{V}$$

   The remaining problem we have to solve is

(LPMT(L))

$$\min \sum_{r \in \mathcal{R}} \sum_{e \in \mathcal{E}(L)} w_r \; c_e \; x_r^e \tag{4.1}$$

$$\begin{aligned} s.t. \quad & \sum_{e \in \mathcal{E}(L)} \theta_{ie} x_r^e = b_r^i \;\; \forall \; r \in \mathcal{R}, i \in \mathcal{V} \\ & \sum_{l \in L} C_l \leq B \\ & x_r^e \in \mathbb{B} \qquad\qquad \forall \; r \in \mathcal{R}, e \in \mathcal{E} \end{aligned}$$

This problem is a shortest path problem and can be solved in polynomial time. We denote an optimal solution of (LPMT(L)) by $\bar{x}(L)$.

2. Fixing the x-variables:
This means, we have the paths of the customers already given. Since each driving edge corresponds to exactly one line, we get the solution of the y-variables straightforwardly by setting

$$y_l := \max_{r \in \mathcal{R}, e \in \mathcal{E}^l} x_r^e \in \{0, 1\}$$

for all $l \in \mathcal{L}$. Again we have to check the budget constraint to find out if the corresponding solution is feasible.

4.2 Greedy heuristics

The idea of a *Greedy heuristic* is to construct a solution, choosing at each step the item bringing the "best" immediate reward. Often, the solutions are constructed from scratch starting with the empty set.

4.2.1 Starting with empty set

More formally, we suppose that the problem can be written as a combinatorial problem in the form. Let $c(Q)$ be a real-valued function defined on all subsets of X, we consider the problem

$$\min\{c(Q) : Q \subseteq X\}$$

.
The pure Greedy heuristic can formally be formulated as followed.

Algorithm 4.2. ([Wol98], [NW88]) Greedy heuristic - general form

1. Set $Q^0 = \emptyset$ (start with the empty set). Set $t = 1$.
2. Set $j_t = \arg\min_{j \in X \setminus Q^{t-1}} (c(Q^{t-1} \cup \{j\}) - c(Q^{t-1}))$ (choose the element whose additional cost is minimum).
3. If the previous solution Q^{t-1} is feasible, and the objective value has not decreased, i.e. $c(Q^{t-1}) \leq c(Q^{t-1} \cup \{j_t\})$, stop with $Q^G = Q^{t-1}$.
4. Otherwise set $Q^t = Q^{t-1} \cup \{j_t\}$.
5. If $Q^t = X$, and solution Q^t is feasible, stop with $Q^G = Q^t$.
   If Q^t is not feasible, stop: problem is infeasible.
6. Otherwise set $t := t + 1$, and return to 2.

In our case we get a combinatorial problem if we minimize over all possible line concepts. The corresponding reformulation is:

(CLPMT)

$$\min_{L \subseteq \mathcal{L}} \sum_{r \in \mathcal{R}} \sum_{e \in \mathcal{E}(L)} w_r \; c_e \; x_r^e \tag{4.2}$$

$$s.t. \sum_{e \in \mathcal{E}(L)} \theta_{ie} x_r^e = b_r^i \qquad \forall \; r \in \mathcal{R}, i \in \mathcal{V} \tag{4.3}$$

$$\sum_{l \in L} C_l \leq B \tag{4.4}$$

$$x_r^e \in \mathbb{B} \qquad \forall \; r \in \mathcal{R}, e \in \mathcal{E}(L) \tag{4.5}$$

Corresponding to the notation of Algorithm 4.3, we define $L := Q$ (line concept), $L^t := Q^t$ (line concept in t^{th} iteration),

$$c(L) = \sum_{r \in \mathcal{R}} \sum_{e \in \mathcal{E}(L)} w_r \; c_e \; \bar{x}_r^e(L) := c(Q),$$

where $\bar{x}(L)$ is a solution of the shortest path problem in the change&go-network constructed from the corresponding line concept L, i.e. a solution of (LPMT(L)).

A line concept L is *feasible* if it satisfies the constraints

$$\sum_{l \in L} C_l \leq B, \tag{4.6}$$

and

$$\sum_{e \in \mathcal{E}(L)} \theta_{ie} \bar{x}_r^e(L) = b_r^i \quad \forall\ r \in \mathcal{R}, i \in \mathcal{V}. \tag{4.7}$$

We now get the following Greedy algorithm for (LPMT).

Algorithm 4.3. Greedy for (LPMT)

1. Set $L^0 = \emptyset$ (start with the empty set). Set $t = 1$.

2. Set
$$l_t = \arg \min_{l \in \mathcal{L} \setminus L^{t-1}} (c(L^{t-1} \cup \{l\}) - c(L^{t-1})))$$

$$= \arg \min_{l \in \mathcal{L} \setminus L^{t-1}} (\sum_{e \in \mathcal{E}(L^{t-1} \cup \{l\})} \sum_{r \in \mathcal{R}} w_r c_e \bar{x}_r^e(L^{t-1} \cup \{l\}) - \sum_{e \in \mathcal{E}(L^{t-1})} \sum_{r \in \mathcal{R}} w_r c_e \bar{x}_r^e(L^{t-1}))$$

   (choose the line that serves passengers that have not been served yet and shortens the path of passengers that have been served already by another line).

3. If the previous line concept L^{t-1} is feasible, and the objective value has not decreased, i.e. $c(L^{t-1}) \leq c(L^{t-1} \cup \{l_t\})$, stop with $L^G = L^{t-1}$.

4. Otherwise set $L^t = L^{t-1} \cup \{l_t\}$.

5. If $t = |\mathcal{L}|$, and line concept L^t is feasible, stop with $L^G = L^t$.
   If L^t is not feasible, stop: problem is infeasible.

6. Otherwise set $t := t + 1$, and return to 2.

Unfortunately the calculation of the "best" line in Step 2 is too time consuming for real world instances. So, we have to find different functions to find a "good" line.

Different line choice criteria in Step 2

The main changes between different Greedy heuristics occur in Step 2 where we have to find a criterion that tells us which line to choose for the line concept. It should be the one bringing the "best" immediate reward, as we mentioned above. As we are not allowed to exceed a given financial budget, we should choose a line with low line cost C_l. Thus the line costs will be in the nominator in Step 2. On the other hand, we have to find a line concept that serves all passengers. One possibility is to choose the line that covers the most not yet covered stations in the PTN. The criterion in Step 2 can thus be formulated as:

$$l_t = \arg \min_{l \in \mathcal{L} \setminus L^{t-1}} \frac{C_l}{|S(l) \setminus \cup_{i=1}^{t-1} S(l_i)|}$$

with $S(l)$ be the set of all stations of line l.
We call the resulting Greedy algorithm *Cover Greedy* since we try to cover all stations of the PTN first to find a feasible solution as fast as possible. Table 4.1 shows the results and CPU times for different real world instances explained on Table 3.2. The line costs are set equal to one for all lines.

No.	$\lvert\mathcal{L}\rvert$	budget	obj.val.	CPU
1	10	7	2560.7	0
2	50	30	10837.6	89
3	100	50	27550.1	221
4	132	70	32059.3	740
5	200	70	16298.9	1245
8	300	70	30521.0	11921

Table 4.1: Solutions of the cover greedy algorithms for different line pool sizes

The next heuristic is based on the following idea: We calculate a trivial solution $(\bar{x}, \bar{y}^3)$ of (LPMT3), as defined in Definition 3.43. A line l with a high $\bar{y}_l^3$-value is a line, that is used by some customer on a large number of edges respectively to the total number of its edges. This means, there is strong evidence that this client travels directly on shortest path from its origin to its destination and so, if on the other hand the line cost of this line C_l is low, it may be a good choice for the line concept.
The criterion in Step 2 can thus be formulated as:

$$l_t = \arg \min_{l \in \mathcal{L} \setminus L^{t-1}} \frac{C_l}{\bar{y}_l^3}$$

Note, that the "importance" $(C_l/\bar{y}_l^3)$ of a line does not depend on the current line concept L^{t-1}. So, it is possible to sort the lines by their importance $(l_{i_1}, \ldots, l_{i_{|\mathcal{L}|}})$ before starting the algorithm. During the algorithm the lines are added successively according their precalculated importance to the line concept until a stopping criterion is fulfilled.
We call the resulting Greedy algorithm *List Greedy* since we add the lines according to a precalculated list. Table 4.2 shows the results and CPU times for different real world instances explained in Table 3.2. The line costs are equal to one for all lines.

No.	$\lvert\mathcal{L}\rvert$	budget	obj.val.	CPU
1	10	6	infeasible	0
1	10	9	2290.3	0
2	50	30	infeasible	12
2	50	47	9460.9	117
3	100	50	infeasible	16
3	100	91	24945.1	130
4	132	70	infeasible	134
4	132	121	32059.3	292
5	200	70	infeasible	166

Table 4.2: Solutions of the list greedy algorithms for different line pool sizes

Other line-choice criteria are possible, such as

- Choose the $\bar{y}^1$-value or the $\bar{y}^2$-value instead the $\bar{y}^3$-value in the List Greedy algorithm.
- Let OD_l denote the number of origin-destination pairs that use one or more driving edges of line l on their shortest path in the change&go-network constructed on the whole line concept:

$$OD_l := \sum_{r \in \mathcal{R}} (\max_{e \in \mathcal{E}^l} \bar{x}_r^e) \tag{4.8}$$

  Sort the lines by the minimal cost and maximum OD_l value: $\frac{C_L}{OD_l}$ and apply this list to the List Greedy algorithm.

Note that in the first iterations of these heuristics the shortest path feasibility criterion (4.7) will likely not be fulfilled since there are not enough lines to serve all passengers. To save computation time it is possible to check feasibility in these first iterations by checking shortest path in the corresponding (smaller) line-change-graph presented in Section 3.3.1 (see Figure 3.7).

4.2.2 Starting with line pool

So far we started with an empty line concept and added lines successively. In this subsection we present heuristics that work the opposite way around. We start with the complete line pool as line concept and delete successively expensive, unimportant lines until the budget constraint is fulfilled and no improvement of the objective value is possible. We call this kind of heuristic *delete heuristics*.
Note that this is a way to implement the Relaxation-and-Separation Heuristic explained in Section 4.3, since we successively set some $y_{l_t} = 0$ and by adjusting the corresponding change&go-network, we set the corresponding $x^e_{st} = 0$ for all $e \in \mathcal{E}^{l_t}$.

Algorithm 4.4. Delete heuristic - general form

1. Set $L^0 = \mathcal{L}$ (start with line pool as line concept). Set $t = 1$.
2. Set $l_t = \arg\max_{l \in \mathcal{L} \setminus L^t} \frac{C_l}{W(l)}$ (choose the line with maximal cost and minimal relevance, measured by some relevance function W).
3. If the previous solution L^{t-1} is feasible, and the objective function value has not decreased, stop with $L^D = L^{t-1}$.
4. Otherwise set $L^t = L^{t-1} \setminus \{j_t\}$.
5. Otherwise set $t := t + 1$, and return to 2.

As before, feasibility has to be checked concerning

- the budget constraint (4.6)
- the shortest path constraints (4.7)

But now, in the beginning of the algorithm, the budget constraint is violated and the shortest path constraints are satisfied. Therefore we have to check in each iteration the length of the shortest paths in the real change&go-graph and not as in the first iterations of the Greedy heuristics that start with an empty set, in the shrunken line-change-graph and thus the computational effort increases.

We have various possibilities to choose the relevance function in Step 2 of the algorithm.

1. Let $OD_l(L)$ denote the number of origin-destination pairs that use one or more driving edges of line l on their shortest path in the change&go-network constructed by the current line concept L:

$$OD_l(L) := \sum_{r \in \mathcal{R}} (\max_{e \in \mathcal{E}^l} \bar{x}_r^e(L)) \tag{4.9}$$

   Then, the relevance function can be set to:

$$W(l) := OD_l$$

   We call this algorithm *OD-delete heuristic*. Table 4.3 shows solution values and CPU times for different instance sizes.

2. Let $\bar{x}_{st}^e(L)$ denote the solution of the shortest path problem of origin-destination pair $(s,t) \in \mathcal{R}$ in the change&go-network constructed by the current line concept L. Then

$$W(l) := \sum_{e \in \mathcal{E}^{l_t}} \sum_{(s,t) \in \mathcal{R}} \bar{x}_{st}^e(L)$$

   We call this algorithm *path-delete heuristic*. Table 4.4 shows solution values and CPU times for different instance sizes.

Of course many other relevance functions are possible such as sorting the lines in advance by their $\bar{y}^i$-value of Lemma 3.44 for $i = 1, 2$, or 3 and setting the relevance function to

$$W(l) := \bar{y}^i.$$

4.3 Relaxation and Separation

In this section we *relax* the given problem (LPMT4) by ignoring the coupling constraints

$$x_{st}^e \leq y_l \quad \forall\, (s,t) \in \mathcal{R}, e \in \mathcal{E}^l : l \in \mathcal{L}$$

to get a polynomial solvable problem, namely a shortest path problem with solution vector $\bar{x}$. In a second step, the *separation step*, we calculate the values of the $\bar{y}$-variables as mentioned in Section 4.1 (Fixing the x-variables) and consider the coupling constraints that are not satisfied by the shortest path solution. In this case there is a $\bar{x}_{st}^e$ with

$$\bar{x}_{st}^e > \bar{y}_l.$$

Now we have two possibilities. We can set $\bar{x}_{st}^e := 0$ or $\bar{y}_l := 1$.

No.	$\|\mathcal{L}\|$	budget	obj.val.	CPU
1	10	6	2615.2	3
2	50	30	9828.1	161
3	100	50	27695.3	2131
4	132	70	31854.5	4405
5	200	70	15721.2	30393
5	200	180	15133.4	4441
6	250	200	19116.3	24675
7	300	250	26624.8	46185
8	330	50	28510.2	348654

Table 4.3: Delete OD-heuristics for different line pool sizes

No.	$\|\mathcal{L}\|$	budget	obj.val.	CPU
1	10	6	2615.2	2
2	50	30	9926.9	161
2	50	40	9525.4	85
3	100	50	26008.4	3036
3	100	80	14912.2	1255
4	132	70	32504.0	8954
4	132	100	31843.2	4806
5	200	70	15689.2	29764
5	200	180	15131.3	4169
6	250	200	19114.5	16206
7	300	250	26616.5	35750
8	330	50	28415.6	331672

Table 4.4: Delete path-heuristics for different line pool sizes

1. If we set $\bar{x}^e_{st} := 0$, this means that we do not allow any passenger to use this driving edge. We delete the edge from the change&go-graph, but we do not change the problem itself. The remaining problem still is a shortest path problem.

2. It we set $\bar{y}_l := 1$, this means that this line has to be in the line concept and we can delete its cost C_l from the budget.

More formally we get the following algorithm:

Algorithm 4.5. Relaxation and separation - general form

1. Solve (LPMT4) without the coupling constraints 3.8 with solution $\bar{x}$ and compute the corresponding $\bar{y}$-values.

2. Feasibility check: If $\bar{x}^e_{st} \leq \bar{y}_l$ is true for all $(s,t) \in \mathcal{R}, e \in \mathcal{E}^l : l \in \mathcal{L}$ then stop with $L := \{l \in \mathcal{L} : y_l = 1\}$
   Otherwise: there is at least one $x^e_{st} > y_l, e \in \mathcal{E}^l$

3. Add the constraint $x^e_{st} = 0$ to the relaxed (LPMT4) and goto Step 1.

This idea will be discussed in more detail in Section 6.

4.4 Line segment heuristic

In the heuristics introduced so far, the lines have always been chosen from a given line pool. The heuristic we will present in this section will construct the lines from scratch. Nevertheless it uses the idea of the change& go-network.

Notation 4.6. We denote by $\mathcal{V}_s := \{(s,l) \in \mathcal{V}\}$ the set of all nodes in the change& go-graph that correspond to node $s \in S$ of the underlying PTN and by $\mathcal{E}_s := \{\{(s,l_1),(s,l_2)\} \in \mathcal{E}_{change}\}$ all changing edges of the change& go-graph between lines that stop at station $s \in S$ of the PTN.

Idea: The line pool is initially constructed by defining each edge in the PTN to be a line. We construct the corresponding change&go-network and calculate shortest paths for all origin-destination pairs to get the solution $\bar{x}^e_{st}$. So, we get values v_e for all changing edges $e \in \mathcal{E}_{change}$:

$$v_e := \sum_{(s,t)\in\mathcal{R}} \bar{x}^e_{st} \quad \forall\ e \in \mathcal{E}_{change}.$$

Now, for each station in the PTN, we consider all corresponding changing edges $\mathcal{E}_s$ and solve the weighted matching problem for them, which is solvable in polynomial time.

Definition 4.7. ([Wol98]) Given a graph $G = (V, E)$. A *matching* $M \subseteq E$ is a set of disjoint edges, that is, at most one edge of a matching is incident to any node $v \in V$.

Definition 4.8. ([NW88]) Given a graph $G = (V, E)$ with m nodes and n edges, and integral weights c_e for $e \in E$. The *weighted matching problem* can be given in the integer programming formulation as follows:

$$\begin{aligned} &\max \sum_{e \in E} c_e x_e \\ &s.t. \sum_{e \in \delta(i)} x_e \leq 1 \quad \forall\, i \in V \\ &x \in \mathbb{Z}_+^n \end{aligned}$$

where $\delta(i)$ is the set of edges incident to node i. This means, we want to find the matching with the maximum sum of the edge-costs of the edges contained in the matching.

If an edge $e = \{(s, l_1), (s, l_2)\}$ is in the matching M we join the lines l_1 and l_2 to a new (longer) line l^*. Doing so for each station of the PTN we end up with a line concept that serves all passengers. As we join the line segments where initially many people had to change to a new line we assure that these people have no transfer at this station and so the number of transfers in the network stays small.

Example 4.9. In Figure 4.1 we see a part of a change& go-network. In the big circle we see all nodes and edges corresponding to station $s \in S$ of the underlying PTN. The numbers on the changing edges correspond to the edge costs. They represent the number of passengers that would have to change between the corresponding lines. The thick lines represent the solution of the weighted matching. We now join the lines $l2 = (b, s)$ and $l5 = (s, e)$ to a new line $l^* := (b, s, e)$ and the lines $l3 = (c, s)$ and $l4 = (s, d)$ to a new line $l' := (c, s, d)$. Line $l1 = (a, s)$ ends at station s.

4.5 Summary

The heuristics starting with the empty set (in the following named *add-heuristics* are much faster than the heuristics starting with the complete line pool (called *delete-heuristics*). This has many reasons:

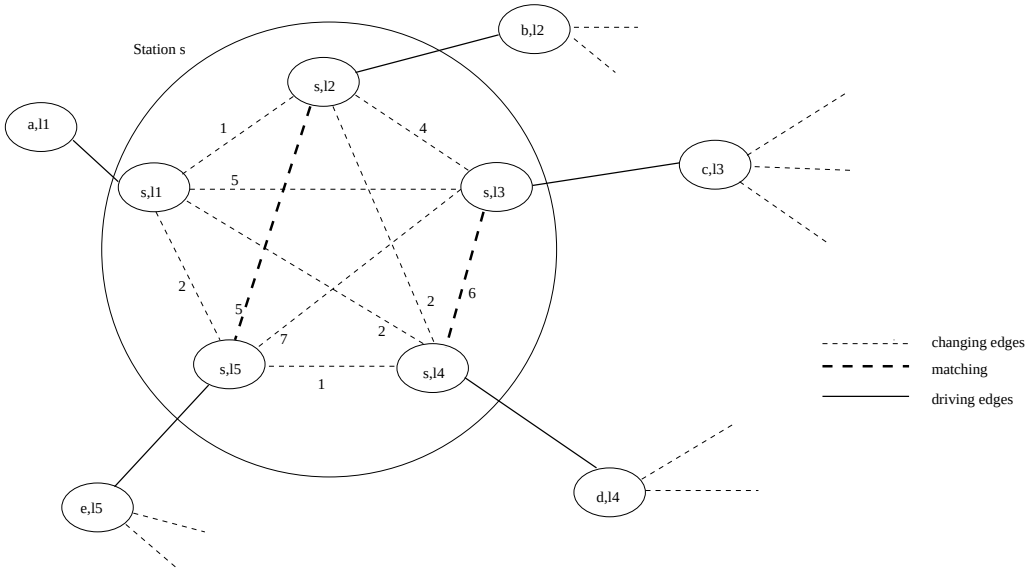

Figure 4.1: Joining lines in the line segment heuristic.

- In real-world instances the budget is much smaller than the number of possible lines in the line pool. Thus, a heuristic that starts with an empty set and adds in each iteration one line needs less iterations than a heuristic that deletes a line from the complete line pool in each iteration.

- The solutions of the add-heuristics are in the first iterations infeasible with respect to the shortest path constraints. Thus, as we mentioned above, it is sufficient to check feasibility using the much smaller line-change-graph. This is not possible for the delete-heuristics since their solutions satisfy the shortest path constraints in the beginning, but violate the budget constraint.

- Solving the shortest path problems of the current change&go-network is the most time consuming step and has to be done in each iteration in the delete-heuristics in contrary to the add-heuristics, where once, the current line concept is feasible and the shortest paths for all origin-destination pairs are computed, we only have to update the paths that may change by adding the new line. This can be done much faster than solving the shortest path problem of the updated network for all origin-destination pairs in each step.

Figure 4.2 shows graphically the CPU times of the three main heuristics for different line pool sizes.
On the other hand, the delete-heuristics are more reliable in finding a feasible solution. As we have seen in Table 4.2, the list-heuristic, which is one of the add-heuristics hardly ever finds a feasible solution. This happens if the sum of the costs of the lines of the current line concept exceeds the budget while not all customers are served yet. This happens especially at add-heuristics that work with a precalculated list of lines. The new line that is chosen to be in the line concept is chosen beforehand without considering the actual situation. As calculating in each step the currently best line to choose, as it is done in the cover-heuristic, costs some computation time, the list-heuristic is much faster. The possibility that a delete-heuristic will find a feasible solution is much higher because they start with a shortest-path-feasible solution. If a line is selected to be deleted that violates the shortest path constraints, it is easy to not delete it and to select a different line. That means that some kind of interchange-heuristic or local search is included to make the procedure more reliable.
Regarding the solution values in Figure 4.3, the delete-heuristics are slightly better. So, it depends on the instance size if the decision maker prefers a fast add-heuristic or a delete-heuristic that will provide a better solution value and a lower risk of not finding a feasible solution.
A speed-up of all heuristics can be done by preprocessing (see Section 6.3) and the use of faster shortest path algorithms. The solution can be enhanced by *local search methods* such as replacing lines from the line concept by lines that are not in the line concept (for details on local search see [NW88], [Wol98]).
The line segment heuristic is somehow different since it is the only method that does not need a line pool but constructs lines itself. Therefore, it will soon be subject of a project work at the Georg-August University of Göttingen.

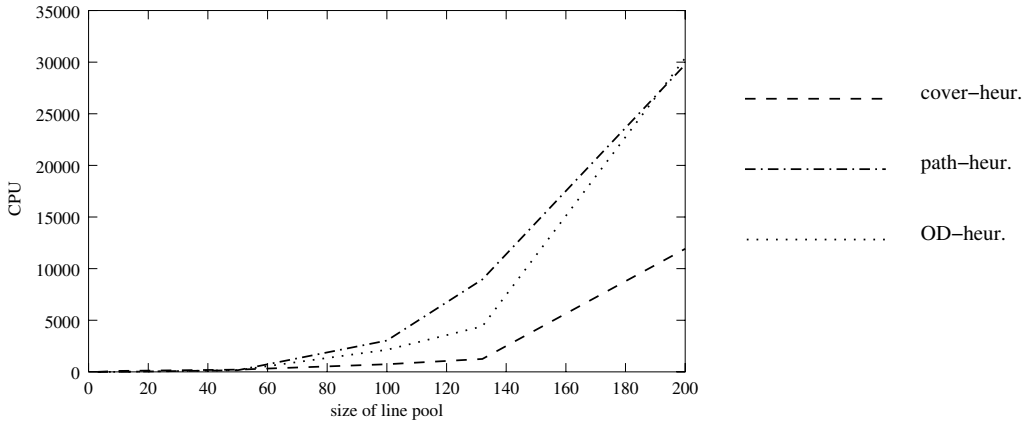

Figure 4.2: CPU times of different heuristics.

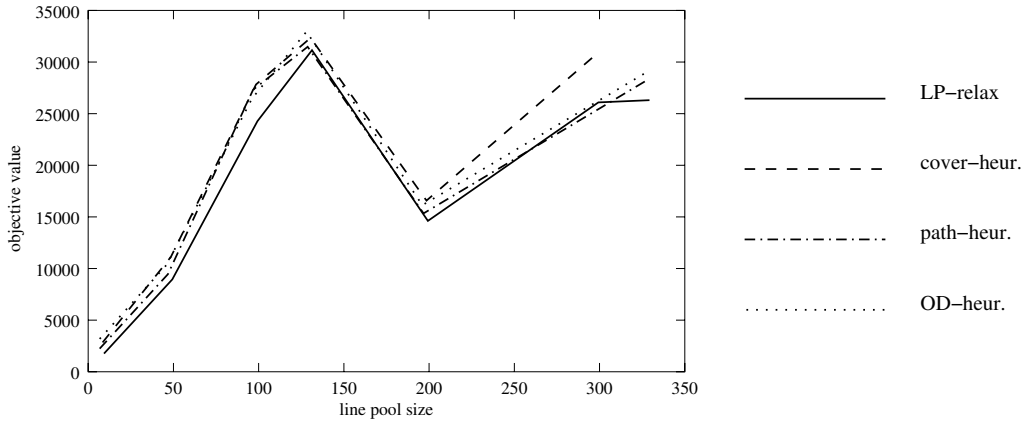

Figure 4.3: Objective values of different heuristics.

Chapter 5

Dantzig-Wolfe-Decomposition

5.1 Theory

Dantzig-Wolfe decomposition (see [DW60]) is a classic approach for structured linear programming problems. It is an important tool to solve structured models that cannot be solved using standard linear programming algorithms as they exceed the capacity of the available LP solvers. The approach also has a potential using parallel computer architectures.
Although this approach can be found in the literature ([Wol98], [Las70]), for the sake of completeness of this thesis, we give a short introduction.

Consider the LP:

(P)

$$\begin{aligned} \min\ & cx \\ s.t.\ & Ax = b \\ & x \geq 0 \end{aligned}$$

Matrix A has block diagonal structure with coupling constraints:

$$Ax = \begin{pmatrix} A_0 & A_1 & A_2 & \cdots & A_K \\ & B_1 & & & \\ & & B_2 & & 0 \\ & 0 & & \ddots & \\ & & & & B_K \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_K \end{pmatrix}$$

where A_0 is a $m_0 \times n_0$ matrix. The constraints

$$\sum_{k=0}^{K} A_k x_k = b_0$$

corresponding to the top row of sub-matrices are called *coupling constraints*. Note that every linear program with more than one constraint can be written in block angular structure for $K = 1$.
Linear programs of type (P) that are appropriate to be solved using Dantzig-Wolfe decomposition preferably contain a number of variables much larger than the number of constraints, and so large that the matrix A can not be stated completely (in practice this means that it can not be completely stored in the memory of the computer). We assume that the matrix is known *implicitly*, i.e. its columns are well defined, but very large in number.
The idea of the Dantzig-Wolfe approach is to decompose this problem, such that never a problem has to be solved with all subproblems $B_k x_k = b_k$ included. Instead an equivalent *master problem* is devised which only concentrates on the coupling constraints and contains only a few more rows than the number of coupling constraints m_0 but with very many columns. This problem is solved without tabulating all these columns by generating them by solving subproblems when the simplex method needs them, a technique that is called *column generation*. The *subproblems* are solved individually. As a result only a series of smaller problems needs to be solved. The subproblems receive a set of parameters (simplex multipliers or prices) from the master program. They send their solution in terms of new columns to the master program, which combines these with previous solutions and computes new simplex multipliers, which are send again to the subproblems. This process proceeds until an optimality test is passed.
The procedure has an elegant economic interpretation, in which the master program coordinates the actions of the subproblems by setting prices on resources used by these problems.

The development of the Dantzig-Wolfe decomposition depends principally on two ideas: column generation and the Minkowski Theorem.

First, we need some basic definitions of polyhedral theory.

Definition 5.1. ([NW88]) Let $Q^0 = \{r \in \mathbb{R}^n : Ar \leq 0\}$. If $Q = \{x \in \mathbb{R}^n : Ax \leq b\}$ is non-empty, then $r \in Q^0 \setminus \{0\}$ is called a *ray* of Q.

Definition 5.2. ([NW88]) A ray r of Q is an *extreme ray* if there do not exist rays $r^1, r^2 \in Q^0 = \{r \in \mathbb{R}^n : Ar \leq 0\}$, $r^1 \neq \lambda r^2$ for any $\lambda \in \mathbb{R}_+$ such that $r = \frac{1}{2}r^1 + \frac{1}{2}r^2$.

Proposition 5.3. ([NW88]) A polyhedron has a finite number of extreme points and extreme rays.

Theorem 5.4. (Minkowski's Theorem, [NW88])
If the feasible region of $Q = \{x : Ax = b, x \geq 0\}$ is non-empty , then any point $x \in Q$ can be characterized as a convex combination of a finite number of its extreme points $x^{(j)}$ plus a non-negative linear combination of extreme rays $r^{(j)}$, i.e.

$$\begin{aligned} x = &\sum_j \lambda_j x^{(j)} + \sum_i \mu_i r^{(i)} \\ &\sum_j \lambda_j = 1 \\ &\lambda_j \geq 0 \\ &\mu_i \geq 0 \end{aligned}$$

A more compact formulation is sometimes used:

$$x = \sum_j \lambda_j x^{(j)} \tag{5.1}$$

$$\begin{aligned} &\sum_j \delta_j \lambda_j = 1 \\ &\lambda_j \geq 0 \end{aligned}$$

where

$$\delta_j = \begin{cases} 1 & \text{if } x^{(j)} \text{ is an extreme point} \\ 0 & \text{if } x^{(j)} \text{ is an extreme ray} \end{cases}$$

If Q is bounded, this result is as follows:

Theorem 5.5. ([NW88])
If the feasible region of $Q = \{x : Ax = b, x \geq 0\}$ is non-empty and bounded, then any point $x \in Q$ can be characterized as a convex combination of a finite number of its extreme points $x^{(j)}$, i.e.

$$\begin{aligned} x = &\sum_j \lambda_j x^{(j)} \\ &\sum_j \lambda_j = 1 \\ &\lambda_j \geq 0 \end{aligned}$$

This means, we can describe the problem in terms of variables λ instead of the original variables x. In practice this reformulation can not be applied directly, as the number of variables λ_j becomes very large.

Back to our problem (P), let $P_k = \{x_k : B_k x_k = b_k, x_k \geq 0\}$ be the feasible region defined by the submatrices and $x_k^{(j)}$ its extreme points and rays. By substituting equation (5.1) into (P), we get the an equivalent problem, the so-called *master problem*.

(Master)

$$\min\ c_0 x_0 + \sum_{k=1}^{K} \sum_{j=1}^{p_k} (c_k x_k^{(j)}) \lambda_k^{(j)} \tag{5.2}$$

$$s.t. \quad A_0 x_0 + \sum_{k=1}^{K} \sum_{j=1}^{p_k} (A_k x_k^{(j)}) \lambda_k^{(j)} = b_0 \tag{5.3}$$

$$\sum_{j=1}^{p_k} \delta_k^{(j)} \lambda_k^{(j)} = 1 \quad \forall\ k = 1, \ldots, K \tag{5.4}$$

$$x_0 \geq 0 \tag{5.5}$$

$$\lambda_k^{(j)} \geq 0 \tag{5.6}$$

where

$$\delta_k^{(j)} = \begin{cases} 1 & \text{if } x_k^{(j)} \text{ is an extreme point of } P_k \\ 0 & \text{if } x_k^{(j)} \text{ is an extreme ray of } P_k \end{cases}$$

Constraints (5.3) are called *coupling constraints* and (5.4) are called *convexity constraints* of the master program.

This is a huge LP. Although the number of rows is reduced, the number of extreme points and rays $x_k^{(j)}$ of each subproblem is very large, resulting in the enormous number of variables $\lambda_k^{(j)}$. The idea is that only variables with a promising reduced cost will be considered, what is also known as *column generation* algorithm.

Definition 5.6. ([HK00]) Assume that an initial feasible basic solution $x_{\mathcal{B}}$ of (P) is available with associated basis matrix $\mathcal{B}$ and cost coefficients $c_{\mathcal{B}}$.

Then

$$\pi = c_{\mathcal{B}}\mathcal{B}^{-1}$$

are called the *simplex multipliers* or *duals* associated with $\mathcal{B}$. Denoting the jth column of A by A^j,

$$\bar{c} = c_j - \pi A^j$$

are called the *reduced cost coefficients*.

Note that a feasible basic solution, if one exists, can be found by using Simplex phase I procedure or a heuristic. The simplex multipliers are always available by the simplex method.

The attractiveness of a variable $\lambda_k^{(j)}$ can be measured by its *reduced cost*. The operation to find reduced costs is often called *pricing*.

Let $\mathcal{B}$ be a feasible $(m_0 + K) \times (m_0 + K)$ basis matrix of the master program, and let $\pi := (\pi_1, \pi_2)$ be the simplex multipliers for this basis, with $\pi_1 := (\pi_{11}, \ldots, \pi_{1m_0})$ associated with the coupling constraints (5.3) and $\pi_2 := (\pi_{21}, \ldots, \pi_{2K})$ with the convexity constraints (5.4), then the reduced cost for variable $\lambda_k^{(j)}$ is

$$\bar{c}_k^{(j)} = (c_k x_k^{(j)}) - \pi \begin{pmatrix} A_k x_k^{(j)} \\ \delta_k^{(j)} \end{pmatrix} = (c_k - \pi_1 A_k) x_k^{(j)} - \pi_{2k} \delta_k^{(j)} \tag{5.7}$$

If for fixed k

$$\min_j \bar{c}_k^{(j)} < 0$$

then, barring degeneracy, the current solution may be improved by introducing $\lambda_k^{(j)}$ into the basis via a pivot transformation.
Assuming the subproblems to be bounded, the problem of finding, for fixed k, $\min_j \bar{c}_k^{(j)}$ is equivalent to solving the kth *subproblem*:

(Subproblem k)

$$\min z_k^0 = (c_k - \pi_1 A_k) x_k^{(j)} \tag{5.8}$$

$$\begin{aligned} s.t. \quad & B_k x_k = b_k \\ & x_k \geq 0 \end{aligned}$$

If the minimal reduced cost obtained by all subproblems is non-negative, i.e.

$$\min_k \min_j \ \bar{c}_k^{(j)} = \min_k (z_k^0 - \pi_{2k}) \geq 0 \tag{5.9}$$

stop. The current solution $x^{opt} = (x_1^{opt}, \ldots, x_K^{opt})$ is an optimal solution to (P) with

$$x_k^{opt} = \sum_{j \ basic} \lambda_k^{(j)} x_k^{(j)} \tag{5.10}$$

If not, the column to enter the basis is that with

$$\min_k (z_k^0 - \pi_{2k}).$$

If the minimum above occurs for $k = s$ and $x_s(\pi)$ solves the subproblem s, the column entering the basis with cost coefficient $c_s x_s(\pi)$ is given by

$$\begin{pmatrix} A_s x_s(\pi) \\ - - - - - \\ e_s \end{pmatrix} \tag{5.11}$$

where e_s is the K-component unit vector.

Until now, the Dantzig-Wolfe decomposition solves optimization problems only on the subproblem-level, but not on the master-level where only a single pivot operation is performed. Furthermore similar columns to the column that has entered the basis could have been generated from the subproblem solution of other subsystems. Probably some of them had negative reduced cost and might be used to reduce the objective in one of the next iterations with new simplex multipliers since successive sets of simplex multipliers should not differ greatly.
Therefore we now introduce a *restricted master program*. This is simply the master program (5.2)-(5.6) with all columns dropped but those in the current basis and one column to enter for each subsystem. Let $x_1^*, \ldots, x_K^*$ be the latest subproblem solutions.

(Restricted Master)

$$\min\ c_0x_0 + \sum_{k=1}^{K} \sum_{j\ basic} (c_k x_k^{(j)})\lambda_k^{(j)} \tag{5.12}$$

$$s.t.\quad A_0x_0 + \sum_{k=1}^{K} \sum_{j\ basic} (A_k^j x_k^{(j)})\lambda_k^{(j)} + \sum_{k=1}^{K} (A_k x_k^*)\lambda_k^* = b_0 \tag{5.13}$$

$$\sum_{j\ basic} \lambda_k^{(j)} + \lambda_k^* = 1 \quad \forall\ k = 1, \ldots, K \tag{5.14}$$

$$x_0 \geq 0 \tag{5.15}$$

$$\lambda_k^{(j)}, \lambda_k^* \geq 0 \tag{5.16}$$

where $\lambda_k^{(j)}$, for "$j\ basic$" are the current basic variables and λ_k^* are the variables entering.
This program has $m_0 + K$ constraints and $n_0 + m_0 + 2K$ variables. In general a greater decrease in the objective value in each step results by adding more than one column in each iteration.

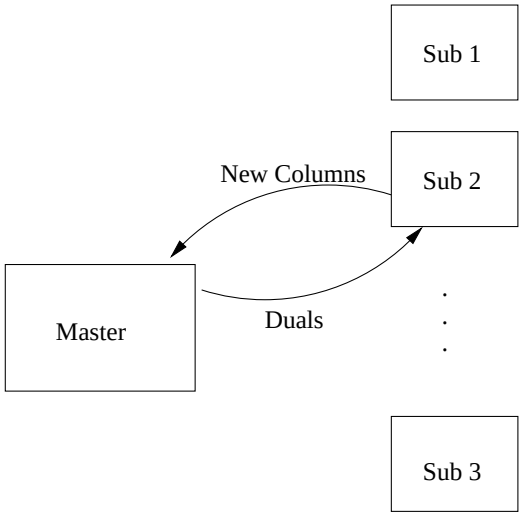

Figure 5.1: Communication between restricted master and subproblems.

Figure 5.1 shows the communication between the restricted master and the subproblems. A basic Dantzig-Wolfe decomposition algorithm can now be formulated.

Algorithm 5.7. Dantzig-Wolfe decomposition algorithm

1. {Initialization}
   Choose initial subset of variables.

2. {Master problem}
   Solve the restricted master problem to get
   π_1 := simplex multipliers of coupling constraints
   π_2 := simplex multipliers of convexity constraints

3. {Subproblems}
   forall $k = 1, \ldots, K$ do
   Adjust subproblem k by the new simplex multipliers π_1 and π_{2k} obtained in Step 2, see (5.8).
   Solve subproblem k
   If $\min_j \bar{c}_k^{(j)} < 0$, and x_k^* the solution of subproblem k: generate new column $\begin{pmatrix} A_k x_k^* \\ - - - \\ e_k \end{pmatrix}$, see (5.11)
   Add this new column with cost coefficient $c_k x_k^*$ to the restricted master program and goto Step 2.

4. If no new columns are generated, i.e. if (5.9) is satisfied, then Stop: optimal.

If the master program is non degenerate, then each iteration decreases the objective value by a non-zero amount. Since there are only a finite number of possible bases, and none is repeated, the decomposition principle will find the optimal solution in a finite number of iterations.
Note that the optimal solution x^{opt} need not to be one of the subproblem solutions. By (5.10) $x^{opt} = (x_1^{opt}, \ldots, x_K^{opt})$ is a convex combination of a number of such solutions. Thus, the function of the Master program is not only to send appropriate prices to the subproblems. It has to combine the subproblem solutions to an overall optimum.

We did not yet pay attention to the initialization of the decomposition. The first thing we can do is solve each subproblem

$$\begin{aligned} \min\ & c_k x_k \\ & B_k x_k = b_k \\ & x_k \quad \geq 0 \end{aligned}$$

If any of the subproblems is infeasible, the original problem is infeasible. Otherwise, we can use the optimal values x_k^* to generate an initial set of

columns.
We also can start with any feasible basic solution for the master program, that we have e.g. by some heuristic.
The initial columns may violate the coupling constraints. We can formulate a Phase I problem by introducing artificial variables and minimizing those. The use of artificial variables is explained in any textbook on Linear Programming (e.g. [HK00]). It is noted that the reduced costs for a Phase I problem are slightly different from the Phase II problem.
As an example, if the coupling constraints are

$$\sum_j x_j \leq b,$$

we can add an artificial variable $x_a \geq 0$ as follows:

$$\sum_j x_j - x_a \leq b$$

The Phase I objective will be

$$\min x_a$$

The reduced cost of a variable x_j is now as in equation (5.7) but with $c_k = 0$.
It is noted that it is important to remove artificials once a Phase II starts.

5.2 Dantzig-Wolfe applied on (LPMT)

The line planning problem introduced in Chapter 3 is NP-hard, and, moreover in real-world instances, gets huge (see Table 3.2). But fortunately the formulations of (LPMT) and (LPMTF) presented in Section 3.3.2 and 3.3.4 have block diagonal structure with only few coupling constraints. Moreover, in both models, all blocks (except the one containing the single budget constraint) are totally unimodular since they are network flow problems. We take advantage of this structure by using a Dantzig-Wolfe decomposition. In this section we present our approach for the (LPMT) formulations. The method can also be applied for solving (LPMTF) since the model structure is very similar. However, the numerical results deal with (LPMT).
The block diagonal structure of the formulations presented in Section 3.3.2 is exemplified on the following reformulation of the LP-relaxation of the (LPMT1) formulation (3.9). The other formulations (LPMT2), (LPMT3), (LPMT4) as well as (LPMTF) can be reformulated analogously.

(LPMT1(LP))

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st} c_e x_{st}^e \tag{5.17}$$

$\boxed{\sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l \quad \forall\, l \in \mathcal{L}}$ coupling constraints

$$\left.\begin{array}{l} \boxed{X^{s_1,t_1}} \\ \qquad \boxed{X^{s_2,t_2}} \\ \qquad\qquad \ddots \\ \qquad\qquad\quad \boxed{X^{s_r,t_r}} \\ \qquad\qquad\qquad\quad \boxed{Y} \end{array}\right\} |\mathcal{R}|+1 \text{ blocks}$$

where

$$X^{st} := \{x_{st} \in \mathbb{R}^{|\mathcal{E}|} : \theta x_{st} = b_{st}, 0 \leq x_{st}^e \leq 1,\ \forall\, e \in \mathcal{E}\}$$
$$Y := \{y \in \mathbb{R}^{|\mathcal{L}|} : C^T y \leq B, 0 \leq y_l \leq 1,\ \forall\, l \in \mathcal{L}\}$$

The coupling constraints can be written as

$$\sum_{(s,t)\in\mathcal{R}} A_X\ x_{st} - A_Y\ y \leq 0$$

where the coefficient matrix $(A_X|\ldots|A_X| - A_Y)$ of the coupling constraints looks as follows:

- A_X is an $|\mathcal{L}| \times |\mathcal{E}|$ matrix given by elements $a_{le} = 1$, if $e \in \mathcal{E}^l$, zero otherwise. It is equal for each origin-destination pair.
- A_Y is an $|\mathcal{L}| \times |\mathcal{L}|$ diagonal matrix containing $|\mathcal{R}||\mathcal{E}^l|$ as its lth diagonal element.

So, we get the following coefficient matrix of (LPMT1):

$$\begin{pmatrix} A_X & A_X & \ldots & A_X & A_Y \\ \theta & & & & \\ & \theta & & & \\ & & \ddots & & \\ & & & \theta & \\ & & & & C^T \end{pmatrix}$$

5.2.1 Master formulations

As we have seen in Section 5.1, the formulation of the Master Problem depends on the definition of the blocks. The blocks we have defined in formulation (5.17) is the finest decomposition we can think of. But it is also possible to keep the budget constraint in the set of coupling constraints or to treat all shortest path blocks as one block. Thus, we can formulate four different Master Problems. Many others are also possible if we join not all but some shortest path blocks together to bigger blocks, e.g. for all $(s,t) \in \mathcal{R}$ starting at a given station s.
In the following we define the *weight-cost-parameters* $c^e_{st} := w_{st}c_e$.

1. The Master Problem corresponding to the decomposition (5.17):

   (Master 1)

$$z = \min \sum_{(s,t)\in\mathcal{R}} \sum_i (c_{st}\ x^{(i)}_{st})\alpha^i_{st} \tag{5.18}$$

$$\begin{aligned}
s.t.\ & \sum_{(s,t)\in\mathcal{R}} \sum_i (A_X\ x^{(i)}_{st})\alpha^i_{st} - \sum_i (A_Y\ y^{(i)})\beta^i + Iv = 0 \\
& \sum_i \alpha^i_{st} = 1 \qquad \forall\ (s,t) \in \mathcal{R} \\
& \sum_i \beta^i = 1 \\
& v_l, \alpha^i_{st}, \beta^i \geq 0
\end{aligned}$$

   where the $|\mathcal{L}|$-vector v are slack variables, and $x^{(i)}_{st}$ and $y^{(i)}$ are the extreme points of X^{st}, and Y, respectively. This problem has $|\mathcal{L}|$ coupling constraints and $|\mathcal{R}| + 1$ convexity constraints.

   For each $(s,t) \in \mathcal{R}$ we obtain the following subproblem

$$z_{st} = \min(c_{st} - \pi A_X)x_{st} - \mu_{st}$$

$$\text{s.t.} \quad x_{st} \in X^{st}$$

   The subproblem of the Y-block is

$$z = \min(-\pi A_Y)y - \mu_{00}$$

$$\text{s.t.} \quad y_l \in Y,$$

where $\{\pi_i\}_{i\in\mathcal{L}}$ are the dual variables of the coupling constraints, $\{\mu_{st}\}_{(s,t)\in\mathcal{R}}$ are the dual variables of the alpha convexity constraints and μ_{00} is the dual variable of the single beta convexity constraint.

Since the X^{st} blocks correspond to shortest path problems which are known to be totally unimodular, the $x_{st}^{(i)}$-values in $\{0,1\}^{|\mathcal{E}|}$.

2. If we add the budget constraint to the set of coupling constraints, we get the following reformulation:

(LPMT1)

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} c_{st}^e x_{st}^e \tag{5.19}$$

$$\boxed{\begin{array}{l} \sum_{(s,t)\in\mathcal{R}} \sum_{e\in l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l \quad \forall\ l \in \mathcal{L} \\ \sum_{l\in\mathcal{L}} C_l y_l \leq B \end{array}} \quad \text{coupling constraints}$$

$$\left.\begin{array}{l} \boxed{X^{s_1,t_1}} \\ \qquad \boxed{X^{s_2,t_2}} \\ \qquad\qquad \ddots \\ \qquad\qquad\qquad \boxed{X^{s_r,t_r}} \end{array}\right\} \quad |\mathcal{R}| \text{ blocks}$$

The corresponding Master Program is:

(Master 2)

$$z = \min \sum_{(s,t)\in\mathcal{R}} \sum_{i} (c_{st}\ x_{st}^{(i)})\alpha_{st}^{i} \tag{5.20}$$

$$\begin{aligned}
s.t.\ & \sum_{(s,t)\in\mathcal{R}} \sum_{i} (A_X\ x_{st}^{(i)})\alpha_{st}^{i} - A_Y y + Iv = 0 \\
& \sum_{l\in\mathcal{L}} C_l y_l \leq B \\
& \sum_{i} \alpha_{st}^{i} = 1 \qquad\qquad \forall\ (s,t) \in \mathcal{R} \\
& v_l, \alpha_{st}^{i}, y_l \geq 0
\end{aligned}$$

where the $|\mathcal{L}|$-vector v are slack variables, and $x_{st}^{(i)}$ are the extreme points of X^{st}. This problem has $|\mathcal{L}| + 1$ coupling constraints and $|\mathcal{R}|$ convexity constraints.

For each $(s,t) \in \mathcal{R}$ we obtain the same subproblems as for our first formulation, namely

$$z_{st} = \min(c_{st} - \pi A_X)x_{st} - \mu_{st}$$

$$\text{s.t.} \quad x_{st} \in X^{st}$$

where $\{\pi_i\}_{i\in\mathcal{L}}$ are again the dual variables of the coupling constraints, and $\{\mu_{st}\}_{(s,t)\in\mathcal{R}}$ are the dual variables of the alpha convexity constraints.

Again the X^{st} blocks correspond to shortest path problems which are known to be totally unimodular, hence the $x_{st}^{(i)}$-values are in $\{0,1\}^{|\mathcal{E}|}$.

3. If we join the shortest path problems to one common block and the budget constraint as a second block we get the following reformulation:

(LPMT1)

$$\min \sum_{e \in \mathcal{E}} c^e x^e \tag{5.21}$$

$\boxed{\sum_{e \in l} x^e \leq |\mathcal{R}||\mathcal{E}^l| y_l \quad \forall\, l \in \mathcal{L}}$ coupling constraints

$\boxed{X}$

$\boxed{Y}$ $\Big\}$ 2 blocks

with $X := \{x \in \mathbb{R}^{|\mathcal{E}|} : x^e = \sum_{(s,t) \in \mathcal{R}} x^e_{st}\ \forall e \in \mathcal{E}, x_{st} \in X^{st}\}$,
$Y := \{y \in \mathbb{R}^{|\mathcal{L}|} : C^T y \leq B, 0 \leq y_l \leq 1,\ \forall\, l \in \mathcal{L}\}$, and
$c^e := \sum_{(s,t) \in \mathcal{R}} c^e_{st}$.

The corresponding Master Program is

(Master 3)

$$z = \min \sum_i (c\ x^{(i)}) \alpha^i \tag{5.22}$$

$$\begin{aligned} s.t. \sum_i (A_X\ x^{(i)}) \alpha^i - \sum_i (A_Y\ y^{(i)}) \beta^i + Iv &= 0 \\ \sum_i \alpha^i &= 1 \\ \sum_i \beta^i &= 1 \\ v_l, \alpha^i, \beta^i &\geq 0 \end{aligned}$$

where the $|\mathcal{L}|$-vector v are slack variables, and $x^{(i)}$ and $y^{(i)}$ are the extreme points of X, and Y, respectively. This problem has $|\mathcal{L}|$ coupling constraints and only two convexity constraints.

The subproblem of the X-block is

$$z = \min \sum_{(s,t) \in \mathcal{R}} (c_{st} - \pi A_X) x_{st} - \mu$$

$$\text{s.t.} \quad x_{st} \in X^{st}$$

and $x^e := \sum_{(s,t)\in\mathcal{R}} x^e_{st}$.

The subproblem of the Y-block is the same as for our first formulation:

$$z = \min(-\pi A_Y)y - \mu_{00}$$

$$\text{s.t.} \quad y_l \in Y,$$

where $\{\pi_i\}_{i\in\mathcal{L}}$ are the dual variables of the coupling constraints, μ is the dual variable of the alpha convexity constraint and μ_{00} is the dual variable of the beta convexity constraint.

In this formulation the $x^{(i)}$-values are integer because they are the component wise sum over shortest path problem solution which are in $\{0, 1\}$.

4. If we add the budget constraint to the coupling constraints and treat the X^{st}-blocks as one block we get the following reformulation:

(LPMT1)

$$\min \sum_{e\in\mathcal{E}} c^e x^e \tag{5.23}$$

$\sum_{e\in l} x^e \le \lvert\mathcal{R}\rvert\lvert\mathcal{E}^l\rvert y_l \quad \forall\, l \in \mathcal{L}$ $\sum_{l\in\mathcal{L}} C_l y_l \le B$	coupling constraints
X	1 block

with $X := \{x \in \mathbb{R}^{|\mathcal{E}|} : x^e = \sum_{(s,t)\in\mathcal{R}} x^e_{st} \ \forall e \in \mathcal{E}, x_{st} \in X^{st}\}$ and $c^e := \sum_{(s,t)\in\mathcal{R}} c^e_{st}$.

The corresponding Master Program is:

(Master 4)

$$z = \min \sum_i (c\ x^{(i)})\alpha^i \tag{5.24}$$

$$\begin{aligned} s.t. \sum_i (A_X\ x^{(i)})\alpha^i - A_Y\ y + Iv &= 0 \\ \sum_{l \in \mathcal{L}} C_l y_l &\leq B \\ \sum_i \alpha^i &= 1 \\ v_l, \alpha^i, y_l &\geq 0 \end{aligned}$$

where the $|\mathcal{L}|$-vector v are slack variables, and $x^{(i)}$ are the extreme points of X. This problem has $|\mathcal{L}| + 1$ coupling constraints and one convexity constraints.

The subproblem of the X-block is

$$z = \min \sum_{(s,t) \in \mathcal{R}} (c_{st} - \pi A_X) x_{st} - \mu$$

$$\text{s.t.} \quad x_{st} \in X^{st}$$

and $x^e := \sum_{(s,t) \in \mathcal{R}} x^e_{st}$.

where $\{\pi_i\}_{i \in \mathcal{L}}$ are the dual variables of the coupling constraints, μ is the dual variable of the alpha convexity constraint.

As in the previous formulation, the $x^{(i)}$-values are again integer because they are the component wise sum over shortest path problem solution which are in $\{0, 1\}$.

Table 5.1 summarizes the characteristics of the presented Master Programs: number of coupling constraints, number of convexity constraints and if the extreme points of the network flow problems are integer or $\{0, 1\}$.

So far, we only considered the possible decompositions and corresponding Master programs of the (LPMT1) formulation. But what about the other

Master No.	coupling	convexity	$x^{(i)}$
1	$\|\mathcal{L}\|$	$\|\mathcal{R}\|+1$	$\{0,1\}$
2	$\|\mathcal{L}\|+1$	$\|\mathcal{R}\|$	$\{0,1\}$
3	$\|\mathcal{L}\|$	2	integer
4	$\|\mathcal{L}\|+1$	1	integer

Table 5.1: Characteristics of the Master programs of (LPMT1).

formulations?
As we already mentioned, the extreme points of the X-subproblem of the (Master 3) and the (Master 4) are not $\{0,1\}$, but integer. This is due to the fact that we sum up the $\bar{x}^e_{st}$ over all origin-destination pairs. By doing this, we loose the information of the exact paths of the customers which are needed in the (LPMT3), (LPMT4) and (LPMTF) formulation. So, it is easy to see that only (Master 1) and (Master 2) can be adapted to these three formulations, but (Master 3) and (Master 4) cannot. (LPMT2) can be decomposed in all mentioned ways like the (LPMT1) without restriction as the x^e_{st} are summed up over all origin-destination pairs in its coupling constraints, too.
Table 5.2 summarizes our results.

	Master 1	Master 2	Master 3	Master 4
(LPMT1)	yes	yes	yes	yes
(LPMT2)	yes	yes	yes	yes
(LPMT3)	yes	yes	no	no
(LPMT4)	yes	yes	no	no
(LPMTF)	yes	yes	no	no

Table 5.2: Possible masters for the different formulations

5.2.2 Strength of the Master Program

Since the original and the master formulation may differ in their linear programming relaxations, we will now discuss the strength of the linear programming masters introduced in this chapter.

Given the following integer program:

(P)

$$\begin{aligned} \min\ & c^T x \\ s.t.\ & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

where A has block-angular structure:

$$Ax = \begin{pmatrix} A_0 & A_1 & A_2 & \cdots & A_K \\ & B_1 & & & \\ & & B_2 & & \\ & & & \ddots & \\ & & & & B_K \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_K \end{pmatrix}$$

Then, the bound provided by the LP-relaxation of (P) is

$$z_{LP}(P) = \min\{cx : \sum_{k=0}^{K} A_k x_k = b_0, \sum_{k=1}^{K} B_k x_k = b_k, x_k \geq 0\}$$

Theorem 5.8. ([Wol98]) The bound provided by the corresponding Master program (M) is

$$z_{LP}(M) = \min\{\sum_{k=0}^{K} c_k x_k : \sum_{k=0}^{K} A_k x_k = b_0, x_k \in conv(X^k)\ \forall\ k = 1 \dots K\}$$

with $X^k := \{x_k \in \mathbb{N}^{n_k} : B_k x_k = b_k\}$.

Proof. (M) can be obtained from the original problem (P) by substituting $x_k = \sum_i x_k^{(i)} \lambda_{k,i}$ with $\sum_i \lambda_{k,i} = 1$, $\lambda_{k,i} \geq 0$. This is equivalent to substituting $x_k \in conv(X^k)$. ■

This theorem tells us precisely how strong a bound is we obtain from the Master.
From integer programming theory (see e.g. [Wol98]), we know that

$$z_{LP}(P) \leq z_{LP}(M) \leq z_{IP}$$

where z_{IP} is the integer solution value ($z_{IP} = z_{IP}(P) = z_{IP}(M)$).

In certain cases the bound provided by the Master has the same strength as the linear programming relaxation.

If $conv(\{x_k \in \mathbb{N}^{n_k} : B_k x_k = b_k\}) = \{x_k \in \mathbb{R}_+^{n_k} : B_k x_k = b_k\}$ for all $k = 1 \dots K$ (i.e. if the current formulation of the subsystem already has the *integrality property*, that is, all extreme points of the current formulation are integral), then

$$z_{LP}(P) = z_{LP}(M).$$

As we have mentioned, the shortest path blocks X^{st} in (LPMT) as well as the corresponding network flow blocks in (LPMTF) are totally unimodular and thus have the integrality property. We hence get

$$z_{LP}(LPMT) = z_{LP}(\text{Master 2}) = z_{LP}(\text{Master 4})$$

The same result holds for (LPMTF) with the corresponding Masters.

In our case the LP-relaxation of any formulation of (LPMT) is only solvable for small instances, see Table 3.4. Since the LP-relaxation thus is not solvable for real world instances due to the size of the resulting change&go network, a decomposition makes sense even if the provided bound is not better.

5.2.3 Initialization

In Section 5.1 we mentioned that we can use the solution of the subproblems to generate an initial set of proposals and what to do if they violate the coupling constraints.
If in the case of the (LPMT) the solution of the subproblems satisfies the coupling constraints, we know by Lemma 3.44 that these columns are already optimal. Thus, we only need to use the Dantzig-Wolfe approach, if the solution of the subproblems violates the coupling constraints which, using the (LPMT1) or (LPMT2) formulation scarcely appears. In this case, finding a feasible basic solution of the restricted Master program can be done in a much more efficient way than a Simplex phase I procedure. It is also more efficient since some commercial Simplex solvers do not offer a tool that returns the whole Simplex tableau after a solution is found. But in our case the tableau is needed because we want to continue with a column generation procedure. Theoretically this problem can be solved as we know the basis B after the Simplex I phase and thus only have to multiply our new columns with B^{-1} but in real world instances our matrix gets huge and so computing the inverse of it is too costly.

The way we tackle the problem is the following:
We first convert our Master problem to the form

$$\begin{aligned} &\min\ cx \\ &s.t.\ Ax = b \\ &\qquad x \geq 0 \end{aligned}$$

To this end we have to introduce non-negative slack variables with zero cost for all coupling constraints $(v_1, \ldots, v_{|\mathcal{L}|})$ and the budget constraint (v_B). The result is that we nearly have an initial feasible basic solution. As only problem remain the convexity constraints which are equality constraints. For each of these rows we introduce also a non-negative slack variable v_{C_r} with a very high cost M (higher than the highest objective value of the shortest path problems, i.e. $M > \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} c_{st}^e$).
The resulting Simplex starting tableau of (Master 4) is

$y_1 \ldots y_{\lvert\mathcal{L}\rvert}$	v_B	$v_1 \ldots v_{\lvert\mathcal{L}\rvert}$	v_C	b
$0 \ldots 0$	0	$0 \ldots 0$	M	0
$C_1 \ldots C_{\lvert\mathcal{L}\rvert}$	1	$0 \ldots 0$	0	B
	0		0	0
$-A_Y$	$\vdots$	I	$\vdots$	$\vdots$
	0		0	0
$0 \ldots 0$	0	$0 \ldots 0$	1	1

respectively

$y_1 \ldots y_{\lvert\mathcal{L}\rvert}$	v_B	$v_1 \ldots v_{\lvert\mathcal{L}\rvert}$	v_C	b
$0 \ldots 0$	0	$0 \ldots 0$	0	-M
$C_1 \ldots C_{\lvert\mathcal{L}\rvert}$	1	$0 \ldots 0$	0	B
	0		0	0
$-A_Y$	$\vdots$	I	$\vdots$	$\vdots$
	0		0	0
$0 \ldots 0$	0	$0 \ldots 0$	1	1

with $v_B, v_1 \ldots v_{|\mathcal{L}|}, v_C$ as feasible basic solution.

If we now add columns and solve the new Master to get the dual variables, we have to keep in mind to subtract M from the dual of the convexity constraint before adjusting the subproblem with it.

Decomposition	No. 0	No. 1	No. 13
LP relax	0.01	M	M
Master 1	0.19	M	M
Master 2	0.15	M	M
Master 3	0.12	4	17318
Master 4	0.1	1	8715

Table 5.3: CPU times of the LP-relaxation of (LPMT1) using Dantzig-Wolfe approach with different Masters. Instances are explained in Table 3.2. M denotes "out of memory".

5.3 Computational results

We implemented the Dantzig-Wolfe decomposition approach of (LPMT) using Xpress MP 2003 and Microsoft Visual C++ 6.0. The CPU times of this section are based on a 3.06 GHz Intel4 processor with 512 MB RAM.
Our computational experience shows that the (Master 4) variation of the (LPMT1) while solving the subproblem with Dijkstra's shortest path algorithm finds an optimal solution of the LP-relaxation within minutes. Even for the biggest instance a solution was found in less than three hours. Note that line planning is part of the strategic planning level and so even longer computation times of some hours are acceptable. In general only few iterations are needed.

5.3.1 Variations of the Decomposition

First, we implemented the Dantzig-Wolfe decomposition algorithm for all four Masters for (LPMT1). Table 5.3 shows the CPU times for some selected instances. Row "'LP relax"' repeats the computation times solving the LP-relaxation using standard optimization software. Note, that the computation times in this section are based on a machine with less memory than the solutions presented in the rest of this thesis, namely 512 MB instead of 2 GB RAM.
Our computational experience shows that it is better for solving the complete problem if the budget constraint belongs to the coupling constraints (i.e. (Master2) and (Master4)). This is, because it is only one constraint and its variables do not appear in the objective function. Thus, we only need a feasible solution and solving the corresponding subproblem to optimality takes unnecessary computation time.
Normally it is known to be better to split the remaining constraints into as

many blocks as possible in order to reduce the number of iterations and thus the computation time. Of course, the number of convexity constraints in the Master problem increases but since the structure of the convexity constraints is very simple, modern optimization software can manage them easily. In our case, the number of convexity constraints increases enormously if we treat the X^{st}-blocks individually, namely in (Master1) and (Master2), and so memory problems arise even for small instances. On the other hand, we figured out that even if we treat the shortest path problems as one subproblem and thus only have one convexity constraint, we only need very few iterations such that there is no need for further decomposition.
With regard to the Ti-values of Table 3.5, we are mainly interested in the solution of the LP-relaxation of (LPMT3) and (LPMT4). With our knowledge about the inefficiency of treating the budget constraint as a separate block, we implemented the Dantzig-Wolfe approach for these two formulations using (Master2) (remember, that there is no (Master4) for them, see Table 5.2). Unfortunately, even using the Dantzig-Wolfe approach, it is not possible to solve even small instances of (LPMT4). Whereas (LPMT3) could be solved for instances up to 116 lines and 16274 origin-destination pairs (see Table 5.4).

5.3.2 Variations of the line pool

As already specified, the problem size and thus the CPU time depends mainly on the size of the line pool and on the number of origin-destination pairs (see e.g. Table 3.3). We calculated the 14 instances of Table 3.2 to show the dependency between the running time and the line pool size. As we already mentioned, we always used the same origin-destination matrix, but to avoid infeasibility we deleted such origin-destination pairs that could not be served even if all lines of the line pool were used. The remaining number of elements of $\mathcal{R}$ is shown in Table 5.4 in column "$|\mathcal{R}|$".
In column "CPU1", we show the CPU time of an implementation of (LPMT1) with (Master4). Since we only need very few iterations, we can save a lot of running time by checking whether the π change in comparison to the last iteration or not after solving the subproblems. If not, there is no need to recalculate the shortest paths in the next iteration since no distances will change. In this case, we just have to adjust the last objective value by $-\mu$. The CPU times of this variation are shown in Table 5.4 in column "CPU1 check" for (LPMT1) with (Master 4) and in column "CPU3 check" for (LPMT3) with (Master 2). In all implementations we solved the subproblems by Dijkstra's algorithm.
Figure 5.2 shows graphically the results of Table 5.4.

No.	$\|\mathcal{L}\|$	$\|\mathcal{R}\|$	CPU1	CPU1 check	CPU3 check
0	3	2	0.1	0.05	0.1
1	10	2602	2	1	228
2	50	4766	10	3	606
3	100	11219	52	16	8706
4	132	18238	92	48	M
5	200	10126	155	78	M
6	250	13246	689	329	M
7	275	14071	1193	691	M
8	300	17507	2167	1171	M
9	330	18433	3246	1911	M
10	350	17095	2769	1814	M
11	375	18350	5150	2727	M
12	400	22191	6525	4789	M
13	423	22756	15243	8715	M

Table 5.4: CPU times of the LP-relaxation of (LPMT1) and (LPMT3) using Dantzig-Wolfe approach with (Master4) and (Master2), respectively, for different line pool sizes. Instances are explained in Table 3.2. M denotes "'out of memory"'.

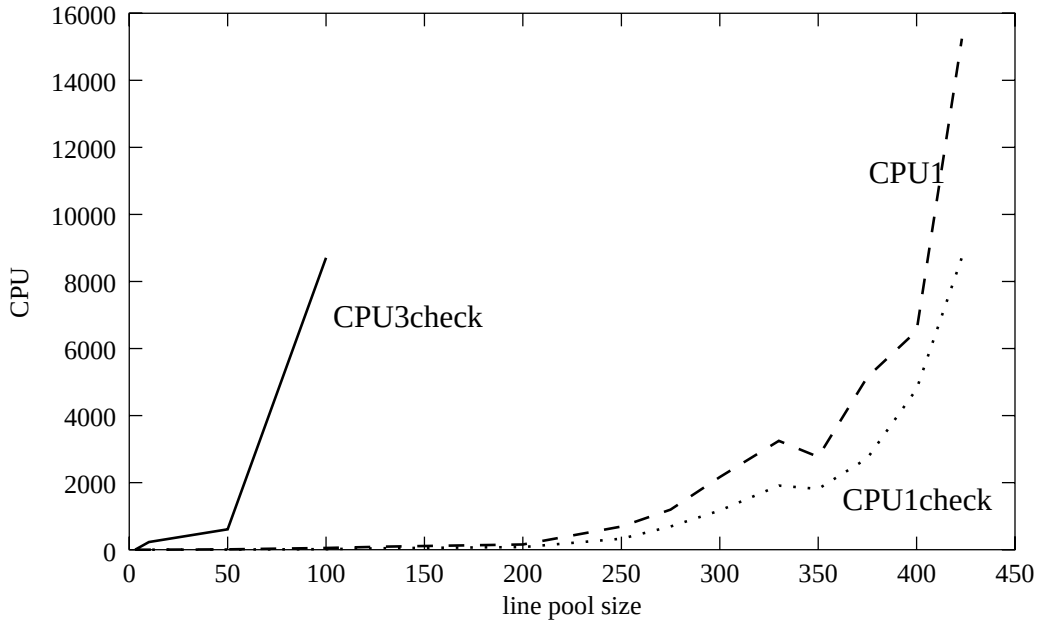

Figure 5.2: Graphical presentation of Table 5.4.

Chapter 6

Exact solution method

We are now in a position to present an exact solution method of the line planning problem with minimal transfers. This method is based on a branch and bound procedure with the LP-relaxation of one of the (LPMT) formulations as upper bounds. The choice of the formulation depends on the size of the network. The bounds can be often calculated in polynomial time by using Lemma 3.44. If this lemma fails we calculate them by using the Dantzig-Wolfe procedure presented in Section 5.2. Note that the x^e_{st} variables are always in $\{0, 1\}$ since they represent the solution of a shortest path problem. So, we only have to branch along the y_l variables. Since they are $\{0, 1\}$ variables and the number of 1s is bounded by the budget, the enumeration tree itself is relatively small.

6.1 Branch & Bound

The most common way to solve integer programs is to use implicit enumeration, or *branch & bound*, in which linear programming relaxations provide the bounds.

Main steps in the branch & bound method:

1. *Selecting* a subproblem.

2. *Relax* the subproblem: For each of the subproblems, we need to obtain a bound on how good its best feasible solution can be. This is usually done by solving a relaxation of the problem. For integer programming problems, the most widely used relaxation is the linear programming relaxation. We can also use other kinds of relaxations, for example Lagrangian relaxation or Dantzig-Wolfe decomposition. In our case

these three possibilities lead to the same bounds (see Section 5.2.2 and [Wol98], Theorem 10.3)

3. *Pruning*: Pruning or terminating is done partially by using the bound to discard a subproblem from further investigation. A subproblem P is pruned if one of the following cases happens:

   (a) *Pruning by infeasibility*: The relaxation of P has no feasible solution.

   (b) *Pruning by bound*: The optimal objective value of the relaxation of P is not better than the objective value of the best feasible solution found so far.

   (c) *Pruning by optimality*: The optimal solution for the relaxation of P is integer.
   Note that if this solution gives better objective value than the current best solution, it becomes the new best solution.

4. *Branching*: The branching is done by dividing the original problem into smaller subproblems. The optimal solution of the relaxed problem will be cut away through the branching.

The general form of the branch & bound algorithm for binary programs (BIP) is:

Algorithm 6.1. Branch & Bound Algorithm for binary programs

1. Initialization. Set $\bar{z} = \infty$, $list = \{(BIP)\}$, n=0.

2. Select subproblem P_k from list. Solve the relaxation of this subproblem P_k. We denote the solution by x^{P_k} and the objective value by z_{P_k}. z_{P_K} gives a lower bound on subproblem P_k.

3. *Pruning by infeasibility*: If there is no feasible solution to P_k, goto Step 7.

4. *Pruning by bound*: If $z_{P_k} \geq \bar{z}$, goto Step 7.

5. *Pruning by optimality*: If x^{P_k} is an integer feasible solution, set $\bar{z} := z^{P_k}$ and $\hat{x} := x^{P_k}$, goto Step 7.

6. Branch on a variable x_j whose value is not integer by fixing the variable at value either 0 or 1. We obtain two new subproblems P_{n+1} and P_{n+2}. Set $list = list \cup \{P_{n+1}, P_{n+2}\}$ and $n := n + 2$. Goto Step 2.

7. Set $list = list \setminus \{P_k\}$.
   Optimality test: If $list = \emptyset$, stop: the current best solution $\hat{x}$ with objective value $\bar{z}$ is optimal to the original (BIP). Otherwise goto Step 2.

6.2 Branch & Bound applied on (LPMT)

As the size of the change&go network and thus the size of the model formulation is huge in real world instances, it is in general not possible to calculate the LP-relaxation using standard optimization software (see Table 3.4). Therefore, we developed in this thesis two other ways to calculate the LP-relaxation:

1. Using the trivial solution if Lemma 3.44 holds.
2. Using Dantzig-Wolfe decomposition if Lemma 3.44 does not hold.

We will now show the Branch & Bound algorithm using the (LPMT1) and the (LPMT3) formulation:

Example 6.2. We consider the PTN shown in Figure 6.1 together with the

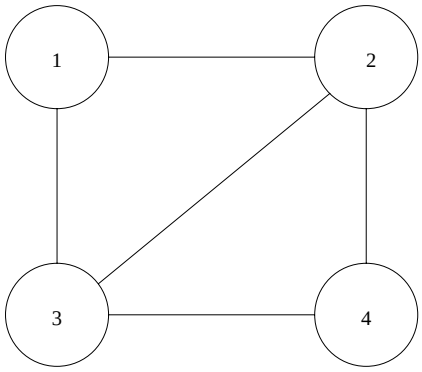

Figure 6.1: PTN of Example 6.2.

line pool

$$\mathcal{L} := \{(l_1, l_2, l_3, l_4\} = \{(1,2,3),(2,4,3),(1,3,4),(1,3)\}$$

and the origin-destination pairs

$$\mathcal{R} := \{(1,3),(1,4),(2,3)\}$$

The edge costs for the driving edges are 1 and the costs of the changing edges are 4. The line costs are set to 1 for all lines and the budget is 2.
A trivial solution is: customer $(1,3)$ uses l_4 (or l_3), customer $(1,4)$ uses l_3 and customer $(2,3)$ uses l_1. Using Lemma 3.44, we get the following T-values representing the minimal budget such that the trivial solution still is the optimal solution of the LP-relaxation of the corresponding formulation:

$$T1 = \frac{1}{6} + \frac{0}{6} + \frac{2}{6} + \frac{1}{3} = \frac{5}{6} < 2$$

$$T2 = \frac{1}{3} + \frac{0}{3} + \frac{1}{3} + \frac{1}{3} = 1 < 2$$

$$T3 = \frac{1}{2} + \frac{0}{2} + \frac{2}{2} + \frac{1}{1} = \frac{5}{2} > 2$$

$$T4 = 1 + 0 + 1 + 1 = 3 > 2$$

We see, that the trivial solution can be used as LP-relaxation for the (LPMT1) and the (LPMT2) formulation, but not for the (LPMT3) and the (LPMT4) formulation. For these, the relaxation must be calculated using Dantzig-Wolfe decomposition or, as it is still possible in this small example, using standard optimization software.
We will now show the branch & bound algorithm for the (LPMT1) and the (LPMT3) formulation.

1. (LPMT1):
   Since the x^e_{st}-values are always integer, it is sufficient to consider only the y_l variables. The LP-relaxation solution of the original problem is

   $$y^{(0)} = (0.17, 0, 0.33, 0.33), \quad z^{(0)} = 4$$

   Since this solution is fractional ($y_{l_1} = 0.17$), our next step is to partition (branch) the original set of feasible solutions into subsets by fixing the value of the variable y_{l_1} at $y_{l_1} = 0$ for one subset and at $y_{l_1} = 1$ for the other subset. Thus, the original problem is divided into two smaller subproblems as follows:
   Subproblem 1: $(y_{l_1} = 0)$

   $$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x^e_{st}$$

$$
\begin{aligned}
s.t. \quad & \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l && \forall\, l \in \mathcal{L} \\
& \theta x_{st} = b_{st} && \forall\, (s,t) \in \mathcal{R} \\
& \sum_{l\in\mathcal{L}} C_l y_l \leq B \\
& y_{l_1} = 0 \\
& x_{st}^e, y_l \in \mathbb{B} && \forall\, (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L}
\end{aligned}
$$

which is equivalent to:

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x_{st}^e$$

$$
\begin{aligned}
s.t. \quad & \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l && \forall\, l \in \mathcal{L}\setminus\{l_1\} \\
& \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^{l_1}} x_{st}^e = 0 \\
& \theta x_{st} = b_{st} && \forall\, (s,t) \in \mathcal{R} \\
& \sum_{l\in\mathcal{L}\setminus\{l_1\}} C_l y_l \leq B \\
& x_{st}^e, y_l \in \mathbb{B} && \forall\, (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L}
\end{aligned}
$$

which is equivalent to solve (LPMT) with a new line pool $\mathcal{L}^* := \mathcal{L}\setminus\{l_1\}$.

Subproblem 2: $(y_{l_1} = 1)$

$$\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x_{st}^e$$

$$
\begin{aligned}
s.t. \quad & \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l && \forall\, l \in \mathcal{L} \\
& \theta x_{st} = b_{st} && \forall\, (s,t) \in \mathcal{R} \\
& \sum_{l\in\mathcal{L}} C_l y_l \leq B \\
& y_{l_1} = 1 \\
& x_{st}^e, y_l \in \mathbb{B} && \forall\, (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L}
\end{aligned}
$$

which is equivalent to:

$$
\min \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}} w_{st}\ c_e\ x_{st}^e
$$

$$
\begin{aligned}
s.t. \quad & \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^l} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^l| y_l && \forall\, l \in \mathcal{L} \setminus \{l_1\} \\
& \sum_{(s,t)\in\mathcal{R}} \sum_{e\in\mathcal{E}^{l_1}} x_{st}^e \leq |\mathcal{R}||\mathcal{E}^{l_1}| && \text{redundant!} \\
& \theta x_{st} = b_{st} && \forall\, (s,t) \in \mathcal{R} \\
& \sum_{l\in\mathcal{L}\setminus\{l_2\}} C_l y_l \leq B - C_{l_1} \\
& x_{st}^e, y_l \in \mathbb{B} && \forall\, (s,t) \in \mathcal{R}, e \in \mathcal{E}, l \in \mathcal{L}
\end{aligned}
$$

which is equivalent to the original problem with slightly changed budget constraint.

It is easy to see that the developed solution methods of the LP-relaxation also work for the subproblems. Solving the LP-relaxation of subproblem 1 and 2 using Lemma 3.44, which has to be adapted for subproblem 2 to

$$T1 := \sum_{l \in \mathcal{L} \setminus \{l_1\}} C_l \frac{T1_l}{|\mathcal{R}||\mathcal{E}^l|} = \frac{1}{3} \leq 1 = B - C_{l_1}$$

yields their optimal solutions:

$$y^{(1)} = (0, 0.33, 0.5, 0), \quad z^{(1)} = 5$$

$$y^{(2)} = (1, 0, 0.5, 0), \quad z^{(2)} = 4$$

Both solutions are feasible and the objective values are smaller than the current best upper bound $\bar{z} = \infty$.
Now we have to choose a selection-rule to decide which subproblem should be branched next. We could do the *best first search* and choose the subproblem with the currently best lower bound z or we could do the *breadth first search* and discharge a whole level of a tree before going deeper or we could do the *depth first search* and continue in one direction until all variables are fixed. We will use the depth first search in this example.

So, we branch subproblem 1 and get subproblem 3 by fixing $y_{l_2} = 0$. Solving subproblem 3, we find no feasible solution. This is obvious since there is no more line serving station 2 if we forbid lines l_1 and l_2. We now go back to subproblem 1 and get subproblem 4 by fixing $y_{l_2} = 1$ whose objective value is 5 and thus still better than the best feasible solution found so far. Note that if we fix a variable to 1 it is sufficient to check if the last solution still works with the new budget constraint. If it is so, this is an optimal solution to the new subproblem. Otherwise, we have to do the Dantzig-Wolfe approach.

The solution procedure of the Branch & Bound method for these two iterations can be described by a *tree search* which is shown in Figure 6.2. The nodes of this tree represent the subproblem (node 0 represents the original problem) and the edges specify how variables are fixed in the solution process. At edge $\{0, 1\}$ and $\{0, 2\}$, the value of y_{l_1} is specified. This variable is referred to as fixed variable. All variables whose values are unspecified at a node are called free variables. The numbers in the nodes indicate the sequence in which the subproblems have been investigated. The numbers at the nodes are the optimal solutions with

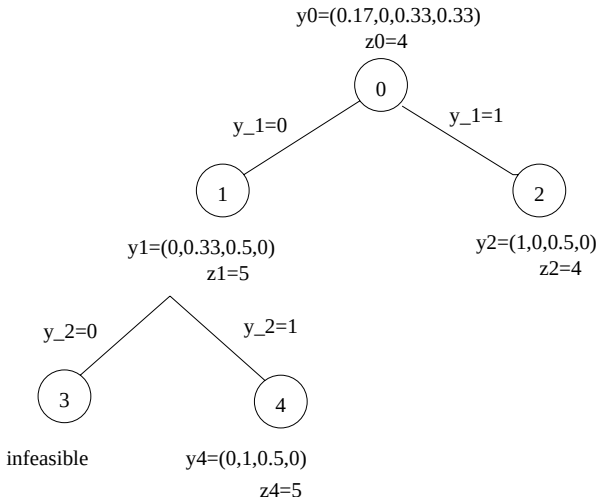

Figure 6.2: The Branch & Bound tree after two iterations.

their objective values of the LP-relaxation of the subproblem. This tree continues "growing branches" iteration by iteration. It is referred to as the *branch and bound tree*.

Figure 6.1 shows the complete branch and bound tree of the example.

Subproblem 5 yields the first integer solution. In this step the upper bound is set to $\bar{z} = 9$, because it is the currently best feasible solution. In subproblem 6, we get again an integer solution with a better objective value than 9, so we set $\bar{z} = 5$ and store this solution as currently best solution. With this problem, searching for solutions with $y_{l_1} = 0$ ends and we continue branching subproblem 2. In subproblem 7 we find a feasible solution of the LP-relaxation of the subproblem with an objective value bigger than $\bar{z}$ and so we can stop branching this problem because we won´t find a better solution in this branch (*pruning by bound*). Finally in subproblem 8 we find the optimal solution.

Note that in this example the depth first rule was not applied consequently, since subproblem 2 was treated before subproblem 6. This was done to show in the beginning of the example how both kinds of subproblems are build. The correct sequence of subproblems using depth first search would be 0,1,3,4,5,6,2,7,8.

2. (LPMT3):

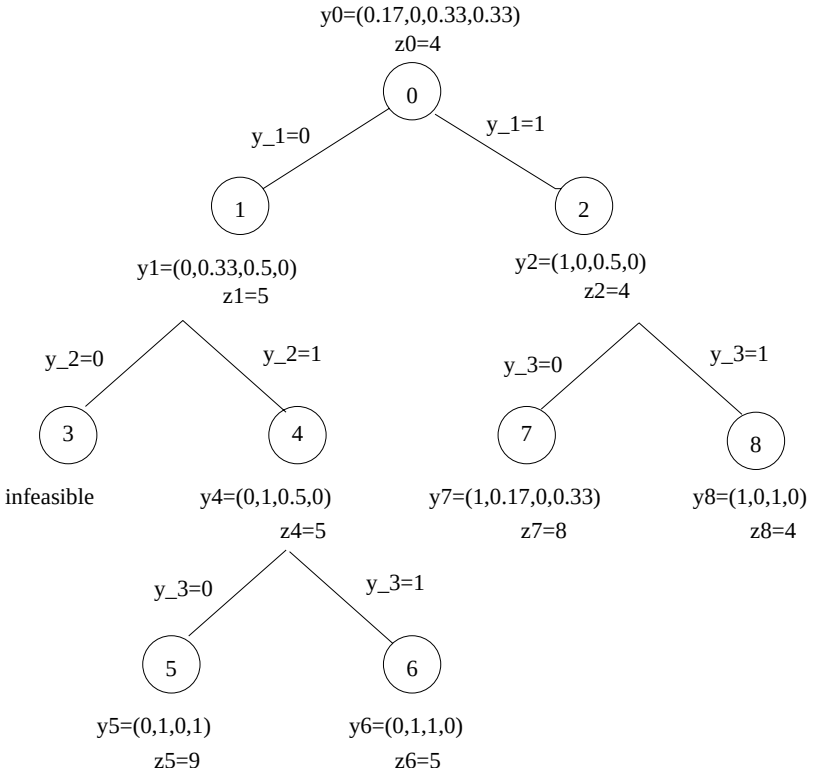

Figure 6.3: The complete Branch & Bound tree (LPMT1).

If we use the (LPMT3) formulation, solving the LP-relaxation is harder because the trivial solution is not feasible. But on the other hand, the bounds provided by this formulation are much better and so the branch and bound tree is much smaller as we can see in Figure 6.4 that shows the complete branch and bound tree of this example.

Concluding, we can say, that the number of iterations, i.e. the size of the tree, depends on the strength of the formulation used, but as we have seen in Section 5.3, the better the strength of a formulation, the harder it is to solve the LP-relaxation.

6.3 Preprocessing

As we have seen, the main problem of our approach is the size of the change&go network whose size depends mainly on the size of the line pool. A wise choice of a possibly small line pool is therefore advisable. On the other hand it makes sense to analyze the underlying PTN. For example if two lines go parallel for

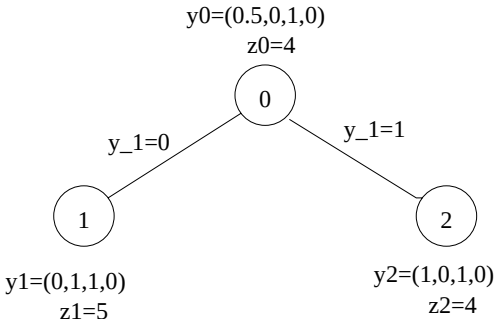

Figure 6.4: The complete Branch & Bound tree (LPMT3).

a long time, it is sufficient to add changing arcs only at the first and the last station. Also arcs between stations without changing possibility can be shrunken together to decrease the size of the network.

Chapter 7

Conclusions

In this thesis we investigated both, the mathematical and the practical aspects of the line planning problem. First, we reviewed some existing approaches of customer and cost oriented line planning problems.

In Chapter 3 we developed a new customer oriented line planning problem (LPMT) that minimizes the total travel time of the customers including penalties for transfers needed. For this, we extended the given public transportation network to a new network, the so-called change&go-network. The choice of its edge costs provides a wide variety of objectives, beginning from "count transfers" with the reduced line-change-network up to the possibility to forbid or to favor special stations for transfers, e.g. because of their infrastructure.
We presented four binary programming formulations (LPMTi), $i = (1, \ldots, 4)$ for (LPMT) and have shown their equivalence. Furthermore, we discussed bicriteria formulations of (LPMT) and have shown that applying the ϵ-constrained method for finding the efficient solutions of the problem is equal to our single-objective binary programming formulations if we set the ϵ to a given budget. Finally, we extended our model to include frequencies of the lines and presented an integer programming formulation for this extension.

We continued with a discussion of the strength of the bounds provided by the LP-relaxation of our four binary, single-objective formulations and figured out that in some special cases the trivial solution which can be found in polynomial time is equal to the solution of the LP-relaxation. Computational results have shown that for some of the formulation this special case often appears. If the trivial solution is infeasible, we have to find other ways to find a solution of the LP-relaxation. In real world instances it is not possible to use standard optimization software due to the problem size. Therefore, in

Chapter 5, we took advantage of the block diagonal structure of the (LPMTi) formulations and proposed an Dantzig-Wolfe decomposition approach. The subproblems turned out to be shortest path problems which can be solved by efficient algorithms and thus the problem was solvable even for large real-world instances of long distance trains of German Railway within less than two hours.

Finally, in Chapter 6 we proposed a branch & bound procedure to find an exact solution. As lower bound, we used the solution value of the LP-relaxation which can be found by the trivial solution or if it is infeasible by the Dantzig-Wolfe approach.

Another way to tackle our problem was based on heuristics. In Chapter 4 we presented and discussed various different heuristics and heuristic ideas. One of them, namely the line segment heuristic, plays a special role since it is the only method in this thesis that constructs lines instead of choosing them from a set of possible lines. This approach will soon be subject of a project work at the Georg-August University of Göttingen.

List of Symbols

Public transportation network:

$PTN = (S, E)$ ..... public transportation network
S ..... set of stations
E ..... set of direct rides
t_e ..... driving time of edge e

Passengers demand:

$\mathcal{R}$ ..... set of origin-destination pairs
w_{st} ..... number of customers traveling from s to t

Line Planning:

l ..... line
$E(l)$ ..... set of edges belonging to line l
$S(l)$ ..... set of stations belonging to line l
L ..... line concept without frequencies
(L, f) ..... line concept with frequencies
$\mathcal{L}$ ..... line pool
f_l ..... frequency of line l
$f(e)$ ..... frequency of edge e
f_e^{min}, f_e^{max} ..... minimum, maximum frequency of edge e
C_l ..... cost of line l
$\mathcal{L}(s)$ ..... set of lines passing through station s

Change&go-network:

$G_{CG} = (\mathcal{V}, \mathcal{E})$ ..... Change&go-network
$\mathcal{V}$ ..... set of nodes
$\mathcal{V}_{CG}$ ..... set of station-line-pair-nodes
$\mathcal{V}_{OD}$ ..... set of in/out-nodes
$\mathcal{E}$ ..... set of edges
$\mathcal{E}_{change}$ ..... set of changing edges
$\mathcal{E}^l$ ..... set of driving edges of line l

$\mathcal{E}_{go}$ ..... set of driving edges
$\mathcal{E}_{OD}$ ..... set of in/out-edges
$\mathcal{E}(L)$ ..... edge set of G_{CG} constructed of the lines of L
c_e ..... cost of edge e
$c_{e_{OD}}$ ..... cost of an in/out-edge
θ ..... node-arc-incidence matrix of G_{CG}
B ..... budget
$(\bar{x}, \bar{y}^i)$ ..... trivial solution of (LPMTi), $i \in \{1, \ldots, 4\}$
Ti ..... minimal budget such that $(\bar{x}, \bar{y}^i)$ is feasible
$\bar{x}(L)$ ..... solution of (LPMT(L))

Set of numbers:
$\mathbb{Z}$ ..... set of integer numbers
$\mathbb{R}$ ..... set of real numbers
$\mathbb{B}$ ..... $\{0, 1\}$
$\mathbb{N}$ ..... set of non-negative integer numbers

List of Figures

List of Tables

Bibliography

[Abr98] T. Abrahamsson, *Estimation of origin-destination matrices using traffic count - a literature survey.*, Technical report Interim Report IR98-021, International Institute for Applied System Analysis, Laxenburg, Austria, 1998.

[BGP04] R. Borndörfer, M. Grötschel, and M.E. Pfetsch, *Models for line planning in public transport*, ZIP-Report 04-10, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2004.

[BKZ96] M.R. Bussieck, P. Kreuzer, and U.T. Zimmermann, *Optimal lines for railway systems*, European Journal of Operational Research **96** (1996), no. 1, 54–63.

[BLL03] M.R. Bussieck, T. Lindner, and M.E. Luebbecke, *A fast algorithm for near cost optimal line plans*, Tech. Report 2003/43, Technische Universität Berlin, 2003.

[BLL04] M.R. Bussieck, T. Lindner, and M.E. Lübbecke, *A fast algorithm for near cost optimal line plans*, mathematical methods of operational research **59** (2004), no. 2, 205–220.

[Bus98] M.R. Bussieck, *Optimal lines in public transport*, Ph.D. thesis, Technische Universität Braunschweig, 1998.

[BZ97] M.R. Bussieck and U.T. Zimmermann, *Schlußbericht - Optimale Linienführung und Routenplanung in Verkehrssystemen (Schienenverkehr)*, 1997.

[Car94] M. Carey, *A model and strategy for train pathing with choice of lines, platforms, and routes*, Tranportation Res. Part B **28** (1994), no. 5, 333–353.

[Cla94] M.T. Claessens, *De kost-lijnvoering*, Master's thesis, University of Amsterdam, 1994.

[CvDZ98] M.T. Claessens, N.M. van Dijk, and P.J. Zwaneveld, *Cost optimal allocation of rail passenger lines*, European Journal on Operational Research **110** (1998), no. 3, 474–489.

[Die78] H. Dienst, *Linienplanung im spurgeführten Personenverkehr mit Hilfe eines heuristischen Verfahrens*, Ph.D. thesis, Technische Universität Braunschweig, 1978.

[DW60] G.P. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Operations Research **8** (1960), 101–111.

[Ehr00] M. Ehrgott, *Multiple criteria optimization*, Lecture Notes in Economics and Mathematical Systems, vol. 491, Springer, Berlin, 2000.

[FW81] P.G. Furth and N.H.M. Wilson, *Setting frequencies on bus routes: Theory and practice*, Transportation Research Record **818** (1981), 1–7.

[Goo04] J.-W. Goossens, *Models and algorithms for railsway line planning problems*, Ph.D. thesis, University of Maastricht, 2004.

[GvHK02] J.-W. Goossens, C.P.M. van Hoesel, and L.G. Kroon, *On solving multi-type line planning problems*, Tech. Report RM/02/009, University of Maastricht, 2002, METEOR Research Memorandum.

[GvHK04] J.-W. Goossens, S. van Hoesel, and L.G. Kroon, *A branch-and-cut approach for solving railway line-planning problems.*, Transportation Science **38** (2004), no. 3, 379–393.

[HC83] Y.Y. Haimes and V. Chankong, *Multiobjective decision making — theory and methodology*, North Holland, New York, 1983.

[Hen89] H. Hensen, *Entwicklung eines interaktiven Programmsystems zur Linienplanung im ÖPNV*, Master's thesis, Technische Universität Berlin, 1989.

[HK00] H.W. Hamacher and K. Klamroth, *Linear and network optimization problems - lineare und netzwerk optimierungsprobleme*, Vieweg Verlag, 2000, Bilingual Lecture Notes.

[Kli00] H. Klingele, *Verfahren zur Optimierung eines Linienkonzeptes der Deutschen Bahn AG*, Master's thesis, Universität Karlsruhe, 2000.

[Kre94] P. Kreuzer, *Linienoptimierung im schienengebundenen Personenverkehr*, Master's thesis, Technische Universität Braunschweig, 1994.

[Las70] L.S. Lasdon, *Optimization theory for large systems*, The Macmillan Company, New York, 1970.

[LS67] W. Lampkin and PD Saalmans, *The design of routes, service frequencies and schedules for a municipal bus undertaking*, Operations Research Quarterly **18** (1967), no. 4, 375–397.

[Min86] M. Minoux, *Mathematical programming, theory and algorithms*, John Wiley & Sons, 1986.

[MUP84] B.R. Marwah, F.S. Umrigar, and S.B. Patnaik, *Optimal design of bus routes and frequencies for ahmedabad*, Transportation Research Record **994** (1984), 41–47.

[New79] G.F. Newell, *Some issues relating to the optimal design of bus routes*, Transportation Science **13** (1979), no. 1, 20–35.

[NW88] G.L. Nemhauser and L.A. Wolsey, *Integer and combinatorial programming*, John Wiley & Sons, 1988.

[Olt94] C. Oltrogge, *Linienplanung für mehrstufige Bedienungssysteme im öffentlichen Personenverkehr*, Ph.D. thesis, Technische Universität Braunschweig, 1994.

[Pat25] A. Patz, *Die richtige Auswahl von Verkehrslinien bei großen Straßenbahnnetzen*, Verkehrstechnik **50/51** (1925).

[PK03] M. Peeters and L.G. Kroon, *Circulation of railway rolling stock: A branch-and-price approach.*, Technical report, Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam, The Netherlands, 2003.

[PRE95] U. Pape, Y. Reinecke, and Reinecke E., *Line network planning*, computer-aided transit scheduling ed., vol. 430, pp. 1–7, Springer, 1995.

[Qua03] C.B. Quack, *Bus line planning*, Master's thesis, University of Delft, 2003.

[Rai] DB German Railway, http://www.bahn.de.

[RR92] Y. Reinecke and E. Reinecke, *Entwurf und Impelmentierung eines Linienplaungssystems für den Busverkehr im ÖPNV*, Master's thesis, Technische Universität Berlin, 1992.

[Sch93] A. Schrijver, *Minimum circulation of railway stock*, CWI Quaterly **3** (1993), 205–217.

[Sch01a] A. Schöbel, *Optimization models for traffic planning, lecture notes*, 2001, http://www.num.math.uni-goettingen.de/schoebel/vorlesung/node1.html.

[Sch01b] M. Schmidt, *Modelle zur Linienoptimierung im Zugverkehr unter Berücksichtigung der Nachfrage*, Master's thesis, Universität Kaiserslautern, 2001.

[Sch03] A. Schöbel, *Customer-oriented optimization in public transportation*, 2003, habilitation thesis, Technische Universtät Kaiserslautern.

[Sim80] C. Simonis, *Omnibusnetze zur Erschließung von Verdichtungsräumen und deren Randzonen*, Bundesminister für Verkehr (1980).

[Sim81a] ———, *Die nachfrageorientierte Optimierung von Omnibuslinien im Stadtbereich durch Verknüpfung von Teilstrecken nach unterschiedlichen Modellansätzen*, Ph.D. thesis, Institut für Stadtbauwesen, RWTH Aachen, 1981.

[Sim81b] ———, *Optimierung von omnibuslinien*, Berichte Stadt - Region - Land, Institut für Stadtbauwesen, RWTH Aachen, 1981.

[Son77] H. Sonntag, *Linienplanung im öffentlichen Personennahverkehr*, Ph.D. thesis, Technische Universität Berlin, 1977.

[Son79] H. Sonntag, *Ein heuristisches Verfahren zum Entwurf nachfrageorientierter Linienführung im öffentlichen Personennahverkehr*, Z. Oper. Res. Ser. A-B **23** (1979), B15–B31.

[SS03] A. Schöbel and S. Scholl, *Planung von Linien mit minimalen Umsteigevorgängen*, Proceedings of the GOR-workshop on "Optimierung im öffentlichen Nahverkehr" (D. Mattfeld, ed.), 2003, http://server3.winforms.phil.tu-bs.de/gor/tagung34/, pp. 69–89.

[SS04a] ———, *Line planning with minimal transfers*, NAM Bericht 72, Georg-August Universität Göttingen, 2004.

[SS04b] ———, *Planning lines with minimal transfers - a dantzig-wolfe approach*, Proceedings of TRISTAN V - The Fifth Triennial Symposium on Transportation Analysis, Le Gosier, Guadeloupe, 13-18 June, 2004.

[Völ01] M. Völker, *Busliniennetze optimal gestalten - Ein multikriterieller Algorithmus zur rechnergestützten Planung*, Der Nahverkehr **10** (2001).

[Weg74] H. Wegel, *Fahrplangestaltung für taktbetriebene Nahverkehrsnetze*, Ph.D. thesis, Technische Universität Braunschweig, 1974.

[Wol98] L.A. Wolsey, *Integer programming*, John Wiley & Sons, 1998.

[ZBKW97] U.T. Zimmermann, M.R. Bussieck, M. Krista, and K.-D. Wiegand, *Linienoptimierung - Modellierung und praktischer Einsatz*, pp. 595–607, Hoffmann, K.-H. and Jaeger, W. and Lohmann, Th. and Schunck, H., 1997.

[ZCvD96] M.T. Zwaneveld, M.T. Claessens, and N.M. van Dijk, *A new method to determine the cost optimal allocation of passenger lines*, Defence or Attack: Proceedings 2nd TRAIL Phd Congress 1996, part 2, TRAIL Research School, 1996.

Wissenschaftlicher Werdegang

06/1994	Abitur am Elisabeth-Gymnasium Mannheim
10/1994-06/2001	Studium der Mathematik mit Anwendungsfach Physik an der Universität Kaiserslautern
06/1997	Vordiplom im Studiengang Mathematik
09/1997-07/1998	Erasmus-Stipendium an der Universidad de Sevilla, Spanien
06/2001	Diplom in Mathematik an der Universität Kaiserslautern
10/2001-09/2004	Stipendium des Graduiertenkollegs "Mathematik und Praxis" an der Technischen Universität Kaiserslautern
seit 10/2004	Wissenschaftliche Mitarbeiterin am "Institut für numerische und angewandte Mathematik" (NAM) der Georg-August Universität Göttingen
03/2005	Einreichung der Dissertation
07/2005	Disputation