Analyzing Train Time Table Graphs

Annegret Liebers

University of Konstanz
Department of Computer and
Information Science

May 2001

Analyzing Train Time Table Graphs

Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften
an der Universität Konstanz
Fachbereich Informatik
und Informationswissenschaft

Annegret Liebers

Mai 2001

Tag der mündlichen Prüfung: 20. Juli 2000
Hauptreferentin: Prof. Dr. Dorothea Wagner, Universität Konstanz
Koreferent: Prof. Dr. Karsten Weihe, Universität Bonn

Teile dieser Arbeit wurden bereits in den Proceedings des *25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'99)* [LWW99], im *International Journal of Foundations of Computer Science* [LWW00], sowie in den Proceedings des *4th Workshop on Algorithm Engineering (WAE 2000)* [LW] veröffentlicht.

Preface

The work on this thesis has been carried out during my employment at the University of Konstanz, Germany, and was partially supported by DFG grant Wa 654/10-2.

I am very grateful to my advisor Dorothea Wagner for giving me the opportunity to come to Konstanz, for her supervision in writing this thesis, and for her generous support, encouragement, and guidance. I am also thankful to Karsten Weihe for our work together during the past years and for his agreement to be my second thesis reviewer. I have learned a lot from both of them. Sincere thanks also go to Marc H. Scholl and Rainer Kuhlen for joining my thesis examination committee.

For our work together within the project that this thesis grew out of I want to further thank Stephan Schmidt, Frank Schulz, and Thomas Willhalm. And I thank Matthias Klein, Stefan Kowski, and Andreas Schikarski of the *TLC* in Frankfurt, the industrial partner in this project.

Rolf H. Möhring and my former colleagues at the Technical University of Berlin encouraged me to learn about discrete mathematics and gave me the freedom to do so, for which I thank them all. To them and to my colleagues in Konstanz I am grateful for the friendly and cooperative working atmosphere they provided. Among them, my office mates Ulrik Brandes and Sabine Cornelsen deserve special thanks for their support and friendship, for many a half an orange or cup of tea, and for maintaining sanity when I was forever fretting over this or that.

I thank the members of my family in Berlin, the Flowes in Virginia, and my friends in Berlin, Konstanz, and elsewhere for their support and kindness. Finally and most importantly, I thank my husband, Bill Flowe.

Contents

English Summary

The project that this thesis grew out of stems from a cooperation, begun in 1996, between the group of Dorothea Wagner at the University of Konstanz and the *Transport-, Informatik- und Logistik-Consulting GmbH/EVA-Fahrplanzentrum* (*TLC* for short), a subsidiary of the German railway company *Deutsche Bahn AG*. This cooperation deals with time tables of public transport, and in this thesis we restrict ourselves to trains (as opposed to busses and ships). Each time table contains the train stations with their arrival and departure times. The abstract representation of the time table data is a directed *time table graph* induced by the time tables: Each train station that appears in at least one time table is represented as a vertex in the graph. If for two vertices u and v there is at least one time table in which u and v are consecutive stops, then the directed edge (u, v) belongs to the time table graph. Such a time table graph for the European train traffic contained some 28 000 vertices and 82 000 edges in the summer of 1997.

The starting point for this thesis is now the following question in which the *TLC* is interested: For every edge e of a time table graph, decide whether e corresponds to a segment of the physical railroad network, or whether the trains that induce e are passing through one or more train stations without stopping there while traveling along e. So every edge in the time table graph has to be classified as either *real* or *transitive*. Additionally, for every transitive edge e, the sequence of train stations passed through by trains while traveling along e is needed. This classification should be done automatically, and solely on the basis of the time table data.

This thesis deals with a structural approach for solving this edge classification problem, which is based on the following observation: Often, a time table graph seems to consist of something like *edge bundles* along lines of railroad tracks, delimited by branching points of the underlying railroad network. If we can identify such edge bundles, then the edge classification problem within the bundles is easy: The real edges form exactly the Hamilton path of the bundle. Modeling the edge classification problem based on this intuitive notion of edge bundles now results in a formalization of the term *bundle* and in formulating the *bundle recognition problem* as a graph theoretic optimization problem. A solution to the bundle recognition problem consists of a set of potential bundle delimiting vertices which

induces a partition of the edge set of the time table graph into bundles. These bundles either correspond to the intuitive edge bundles along lines of railroad tracks (the favourable case) or they form a *trivial bundle* consisting of only one edge (the unfavourable case). The goal of the optimization is to have as few such trivial bundles as possible.

For three versions of the bundle recognition problem NP-completeness results using reductions from 3-SATISFIABILITY or one of its variants can be obtained.

In order to still obtain solutions for the underlying application problem, this thesis develops a heuristic for the bundle recognition problem. This heuristic starts with an initial vertex set of potential bundle delimiting vertices. On the one hand, individual vertices that are not needed are then removed from this set. On the other hand, wherever the edge sets induced by the vertex set do not fulfill the formal bundle conditions, vertices are added to the set. The thesis motivates and illustrates this heuristic in detail, and suggests different criteria for choosing an initial vertex set.

Finally, an extensive computational study compares the results of the heuristic for the European time table graph for the summer of 1997 for different choices of the initial vertex set. Some 80 % of the time table graph edges for the summer of 1997 can be classified as real or transitive. The results are broken down for 13 countries, and not only do some initial vertex sets yield better results than others, but in some countries the time table graph displays the bundle structure more prominently than in others. Accordingly, the heuristic is considerably more successful in some countries than in others (the percentages of edges classified with the bundle recognition approach vary between 93 and 62 for the data sets considered in this study).

So this thesis shows that even though the chosen structural approach for solving the edge classification problem in time table graphs yields difficult problems in terms of complexity theory, it can still lead to an algorithmic procedure with meaningful results for the data occurring in practice.

Deutsche Zusammenfassung

Die dieser Arbeit zugrundeliegende Aufgabenstellung entstammt einer 1996 begonnenen Kooperation zwischen dem Lehrstuhl für Praktische Informatik von Dorothea Wagner an der Universität Konstanz und der *Transport-, Informatik- und Logistik-Consulting GmbH/EVA-Fahrplanzentrum* (kurz *TLC*), einer Tochterfirma der *Deutschen Bahn AG*. Der Untersuchungsgegenstand in dieser Zusammenarbeit sind Fahrplandaten von öffentlichen Verkehrsmitteln, wobei wir uns in der vorliegenden Arbeit auf Züge (im Gegensatz zu Bussen und Schiffen) beschränken. Jeder Fahrplan enthält dabei die angefahrenen Bahnhöfe mit ihren jeweiligen Ankunfts- und Abfahrtszeiten. Die Abstraktion der Fahrplandaten besteht in einem durch die Fahrpläne induzierten gerichteten *Fahrplangraphen*: Jeder Bahnhof, der in mindestens einem Fahrplan auftritt, wird als Knoten im Graph dargestellt. Gibt es für zwei Knoten u und v mindestens einen Fahrplan, in dem u und v direkt aufeinanderfolgende Halte sind, dann gibt es im Fahrplangraphen die gerichtete Kante (u, v). Ein solcher Fahrplangraph für den europäischen Zugverkehr umfasste im Sommer 1997 gut 28 000 Knoten und 82 000 Kanten.

Der Ausgangspunkt für die vorliegende Arbeit ist nun folgende für die *TLC* interessante Aufgabenstellung: Entscheide für jede Kante eines Fahrplangraphen, ob sie einem Teilstück des physikalischen Schienennetzes entspricht, oder ob die diese Kante induzierenden Züge auf ihrem Weg entlang der Kante einen oder mehrere andere Bahnhöfe ohne Halt durchfahren. Im Fahrplangraph muss also jede Kante als *minimal* oder *transitiv* klassifiziert werden. Außerdem wird für jede transitive Kante die Folge der von sie induzierenden Zügen durchfahrenen Bahnhöfe benötigt. Diese Klassifizierung soll automatisiert und allein auf der Grundlage der Fahrplandaten geschehen.

Die vorliegende Arbeit befasst sich mit einem strukturellen Lösungsansatz für dieses Kantenklassifizierungsproblem, dem folgende Beobachtung zugrunde liegt: Betrachtet man einen Fahrplangraphen, so scheint er oft in so etwas wie *Bündel von Kanten* entlang von Schienensträngen zu zerfallen, die von den Verzweigungspunkten des Schienennetzes begrenzt werden. Kann man solche Kantenbündel identifizieren, so wird das Kantenklassifizierungsproblem innerhalb der Bündel einfach: Die den Schienenstrang ausmachenden minimalen Kanten bilden gerade den Hamiltonpfad des Bündels. Der auf dieser intuitiven Vorstellung von Kantenbündeln beruhende Modellierungsschritt leistet nun eine Formalisie-

rung des Begriffs *Bündel* und die Formulierung des *Bündelerkennungsproblems* als graphentheoretisches Optimierungsproblem. Eine Lösung des Bündelerkennungsproblems besteht dabei aus einer Menge von potentiellen Bündelendknoten, die eine Partitionierung der Kantenmenge des Fahrplangraphen in Bündel induziert. Diese Bündel entsprechen (im günstigsten Fall) den intuitiven Kantenbündeln entlang von Schienensträngen, oder sie bestehen (im ungünstigsten Fall) als *triviale Bündel* aus einer einzigen Kante. Das Optimierungsziel ist es, möglichst wenige solche triviale Bündel zu erhalten.

Für drei betrachtete Varianten des Bündelerkennungsproblems lassen sich durch Reduktionen von 3-SATISFIABILITY bzw. einer Variante davon NP-Vollständigkeitsresultate zeigen.

Um dennoch Lösungen für das zugrundeliegende Anwendungsproblem zu erzielen, entwickelt die Arbeit eine Heuristik für das Bündelerkennungsproblem. Diese Heuristik geht von einer initialen Menge potentieller Bündelendknoten aus. Sie nimmt einerseits nicht benötigte Knoten aus dieser Menge heraus und nimmt andererseits dort Knoten in diese Menge auf, wo die induzierte Kantenpartition noch nicht die Bedingungen, die an Bündel (im Sinn der Formalisierung des Bündelbegriffs) gestellt werden, erfüllt. Das Vorgehen dieser Heuristik wird detailliert motiviert und erläutert, und insbesondere werden verschiedene Kriterien für die Wahl einer initialen Knotenmenge vorgeschlagen.

Schließlich vergleicht eine umfangreiche experimentelle Studie die Ergebnisse der Heuristik für den oben erwähnten europäischen Fahrplangraphen aus dem Sommer 1997 für verschiedene Wahlen einer initialen Knotenmenge. Insgesamt lassen sich mit Hilfe der Bündelerkennungsheuristik gut 80 % der Kanten im europäischen Fahrplangraphen als minimal oder transitiv klassifizieren. Die Ergebnisse werden für 13 Länder heruntergebrochen, und es zeigt sich, dass nicht nur verschiedene initiale Knotenmengen verschieden gute Ergebnisse der Heuristik liefern, sondern dass der Fahrplangraph in einigen Ländern viel stärker von der intuitiven Bündelstruktur geprägt ist als in anderen Ländern. Der Erfolg der Heuristik ist dementsprechend in einigen Ländern deutlich höher als in anderen (die Prozentzahlen der durch den Bündelerkennungsansatz klassifizierten Kanten schwanken zwischen 93 und 62 für die in dieser Studie betrachteten Datensätze).

Die vorliegende Arbeit zeigt somit, dass der gewählte strukturelle Ansatz zur Lösung des zugrundeliegenden Kantenklassifizierungsproblems in Fahrplangraphen zwar zu komplexitätstheoretisch schweren Problemstellungen führt, dass er aber für die in der Praxis vorkommenden Daten sinnvoll in eine brauchbare Lösungen erzielende algorithmische Vorgehensweise umgesetzt werden kann.

Chapter 1

Introduction

In 1996, a cooperation between the group of Dorothea Wagner at the University of Konstanz and the *Transport-, Informatik- und Logistik-Consulting GmbH*[1] / *EVA-Fahrplanzentrum* (*TLC* for short), a subsidiary of the national German railway company *Deutsche Bahn AG*[2] began. The subject of this cooperation is the analysis of time table data that the *TLC* collects and maintains.

The *TLC* receives time table information for trains, busses, and ships from various providers throughout Europe. Among them are national railway companies such as the *Deutsche Bahn AG* in Germany or the *SNCF*[3] in France, but also regional public transport authorities. The *TLC* is responsible for combining this information into a unified set of European time tables that can serve as input for time table information systems such as the widely used *HAFAS*-algorithm by the German company *HaCon*[4].

Updates of time tables are delivered to the *TLC* throughout the year and in varying formats. An additional difficulty is that as with any large amount of data collected from various sources the data contains omissions and errors. It takes thus a continuous effort for the *TLC* to maintain a database for European time tables with as few errors as possible. The starting point of our cooperation was the idea that interpreting the time table data as a graph as formally defined in Section 1.1 and analyzing this time table graph may be helpful for this task.

The time table graph constructed from a complete set of European time table data as available to the *TLC* in the summer of 1997 contains about 30 000 vertices (each corresponding to a train station, bus stop, or ship landing) and 90 000 edges (corresponding to pairs of consecutive stops of trains, busses, or ships), together with vertex and edge weights such as the earliest time of day there is a departure at a vertex, or the number of minutes it takes to travel along an edge and many more. One means to support the detection of omissions and er-

[1] see `http://www.tlc.de/`
[2] see `http://www.bahn.de/`
[3] see `http://www.sncf.fr/`
[4] see `http://www.hacon.de/`

rors in the input data is to identify and visualize parts of the graph where the trains fulfill certain criteria constituting plausibility checks, or where the graph constructed from one set of time tables differs from the graph constructed from the corresponding time tables of, say, the previous year. Several people have participated in this cooperation and have designed and implemented a C++ [Str97] class library for time table graph construction, querying and visualization for use by the *TLC* [Doc99]. This project also served as a case study for the concepts developed in [KW96, KW97, Wei97, GKW98].

Section 1.1 will now define the time table graph and give examples. Section 1.2 will then introduce the concrete application problem that is the specific focus of this thesis.

1.1 Time Table Graphs

A time table is a sequence of consecutive stops with times of arrival and departure for each stop. Figure 1.1 shows the time table for a train. The time table data available to the *TLC* in the summer of 1997 consists of time tables for more than 190 000 trains, busses, and ships. A set of time tables induces a directed graph as follows: Each train station (or bus stop or ship landing) appearing in some time table becomes a vertex of the graph, and if there is a train (or bus or ship) leaving r and having s as its next stop, then there is a directed edge (r, s) in the graph. Since the application problem introduced in Section 1.2 deals with trains only, we will from now on ignore time tables for busses and ships. The definition of a time table graph then becomes:

Definition 1.1 *A* time table graph *for a set of train time tables is a directed graph* $G = (V, A)$ *with* $V = \{$ *train stations that appear in the time tables* $\}$ *and* $A = \{ (r, s) \mid \exists$ *a train in the time tables that stops at* r*, goes on to* s*, and does not stop in between* r *and* $s \}$.

Typically, many trains contribute to one edge. So by definition, a time table graph has no multiple edges. Figure 1.2 shows an excerpt of a time table graph, including the edges to which the train from Figure 1.1 contributes. In the European time table data for the summer of 1997, there are time tables for more than 140 000 trains. The time table graph they induce consists of 28 280 vertices and 82 594 edges and is shown in Figure 1.4.

1.2 Problem Formulation and Overview

In the remainder of this thesis, we will deal with the following problem: Decide, solely on the basis of the train time tables, for each edge of the time table graph, whether trains traveling along it pass through other train stations on the way

Sinsheim		05:37
Steinsfurt	05:40	05:41
Reihen	05:44	05:44
Ittlingen	05:48	05:49
Richen	05:52	05:52
Eppingen	05:58	

Figure 1.1: The time table of a local train from Sinsheim to Eppingen. Figure 1.2 shows the corresponding time table graph excerpt.

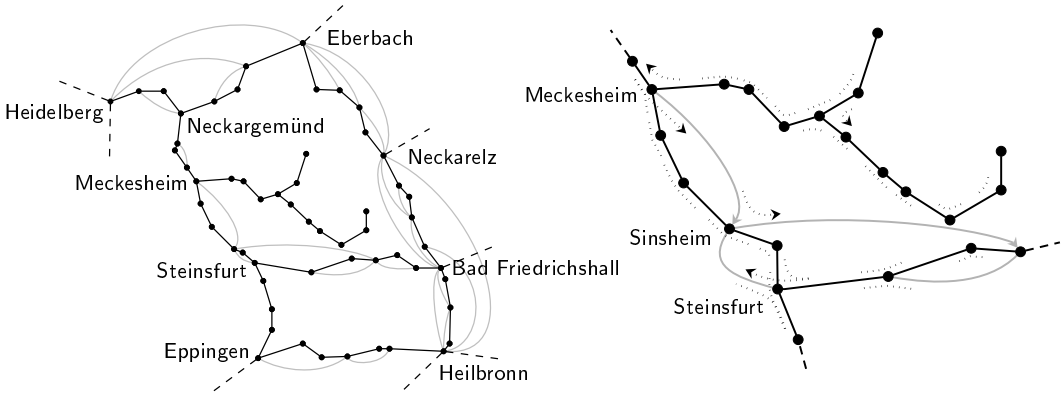

Figure 1.2: *Left:* An example of the underlying undirected graph of a time table graph. Real edges are drawn in black, transitive edges are drawn in grey. *Right:* Line graph edges induced by the trains operating on this part of the time table graph are drawn in dotted lines; edges shown undirected mean that there is a directed edge in either direction.

without stopping there. If so, the sequence of these other train stations is also sought. In other words, the edges have to be classified into *real* edges representing segments of the physical railroad network and *transitive* edges corresponding to paths in this network, and for each transitive edge (s, t), the s-t-path consisting of only real edges that corresponds to the travel route of the trains contributing to (s, t) is sought. Real and transitive edges are indicated in Figure 1.2. So in particular, the train whose time table is given in Figure 1.1 passes through one other train station without stopping there on its way from Sinsheim to Steinsfurt, but on the remainder of its journey to Eppingen, the train stops at every station.

Notice that this problem formulation assumes that the physical railroad network is a subgraph of the given time table graph. According to everyday life experience this is usually so, but time table graphs that do not contain the whole physical railroad network as a subgraph are conceivable as illustrated in Figure 1.3. In such situations, it is unclear what a meaningful edge classification would look like. Since there is no way to reliably identify such situations and no

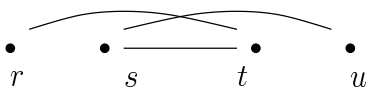

Figure 1.3: In a situation like this, the physical railroad network most likely consists of a line of railroad tracks from r through s and t to u, and in this case, the physical railroad network is not contained in the time table graph shown.

meaningful way to address them, we will pretend they do not exist. But since we know that such situations do not occur often, we are confident that our edge classification results will still carry the desired meaning for almost all edges contained in time table graphs induced by the sets of time tables collected by the *TLC*.

In connection with other projects besides providing quality assurance for the time table data as motivated in the beginning of this chapter, the *TLC* is interested in an algorithmic solution for the edge classification problem based solely on train time tables. Ad-hoc approaches for this problem using local properties such as travel times between stops have been implemented for the *TLC* [Doc99], but this thesis will now investigate the following structural approach: Exploration of railway data suggests that a railroad network decomposes into lines of railroad tracks between branching points of the underlying railroad network, where the lines of railroad tracks correspond to bundles consisting of (usually many) transitive edges and one path of real edges (this is illustrated in Figure 2.1 on page 8). Guided by this intuition, the structural approach consists of identifying bundles and according branching points. Chapter 2 first illustrates the notion of bundles of edges in time table graphs intuitively and suggests to formulate and solve a *bundle recognition problem* in order to solve the edge classification problem. It then formally defines the bundle recognition problem as a graph theoretic optimization problem by formalizing the notion of a bundle. Chapter 3 gives NP-completeness results for (variations of) the bundle recognition problem. Chapter 4 presents a heuristic for bundle recognition, and Chapter 5 describes a computational study carried out on the European time table graph for the summer of 1997.

The heuristic was implemented as a prototype in C++ using additionally the *Standard Template Library (STL)* [MS95] and the *Library of Efficient Data Types and Algorithms (LEDA)*[5] [MN99].

Familiarity with basic terms of graph theory and with the theory of NP-completeness is assumed [BM76, GJ79, CLR90].

So far, this type of problem has not been studied under an algorithmic aspect. Since this application problem does not have a canonical algorithmic formulation, this is not surprising. Tackling this problem entails different aspects of algorithm

[5] see `http://www.mpi-sb.mpg.de/LEDA/leda.html`

engineering [GM99, VZ99, Mor] from modeling application problems to experimental algorithm design and study.

Some previous work addresses related yet different problems and questions: [WBL+99] discusses general problems of experimental algorithm design and refers to the problem in this thesis as one of four examples. [Wei98, SWW99] deal with other algorithmic optimization problems on time table data, while [BW00] addresses visualization of edge classifications.

Regarding public transport in general, [BWZ97] is a survey on public rail transport planning, and [PTS, Kin95] are online bibliographies.

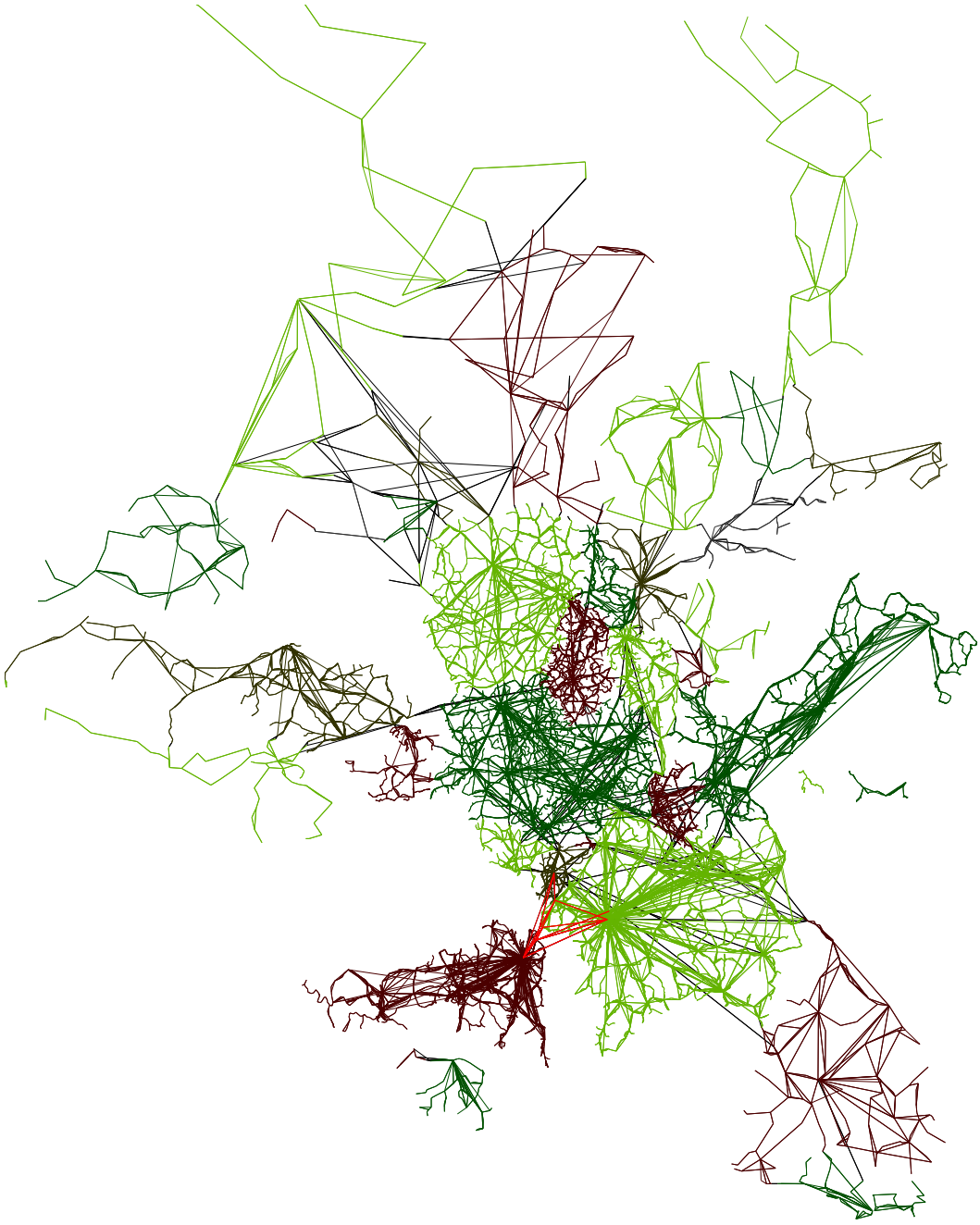

Figure 1.4: The European time table graph (north is left) induced by the train time tables available to the *TLC* for the summer of 1997 with 28 280 vertices and 82 594 edges. The shades of green and brown indicate countries, and edges crossing a country border are black. The EuroStar train through the Channel tunnel is indicated in red.

Chapter 2

Modeling

The application problem introduced in Section 1.2 consists in determining the actual travel route of trains, solely on the basis of their time tables. In time table graphs as they were defined in Section 1.1, this translates into classifying edges as real or transitive, and of finding the paths of real edges that correspond to the travel routes along transitive edges.

2.1 The Intuition of Bundles

Our approach to the given problem is based on the following observation: Often there are bundle-like edge sets along a line of railroad tracks that are delimited by the branching points of the underlying railroad network as illustrated in Figure 2.1. If we can find sets of edges that form such bundles, then we can, within each bundle, classify the edges: The physical railroad network appears as a (unique) Hamilton path of the bundle, and the Hamilton path immediately provides the desired paths for transitive edges. Notice that since a time table graph is a directed graph, along one line of railroad tracks there are usually two bundles, one in each direction.

In Figure 2.1, there are some edges (drawn in thin lines) that do not belong to any bundle. Suppose for now that the branching points and the thin edges are given. Ignoring the thin edges for a moment, the idea for finding edge sets that form bundles is the following: If a train travels along the edge (r, s), and then along the edge (s, t), and if s is not a branching point, then (r, s) and (s, t) belong to the same bundle. So the following procedure would partition the edge set of the time table graph in Figure 2.1 into the bundles that are encircled in dotted lines:

1. Initially, every edge of the time table graph is its own set.

2. For every train and for every triple (r, s, t) of consecutive stops of the train, if s is not a big black vertex, (and if neither (r, s) nor (s, t) are thin edges,) then unite the edge sets that (r, s) and (s, t) belong to.

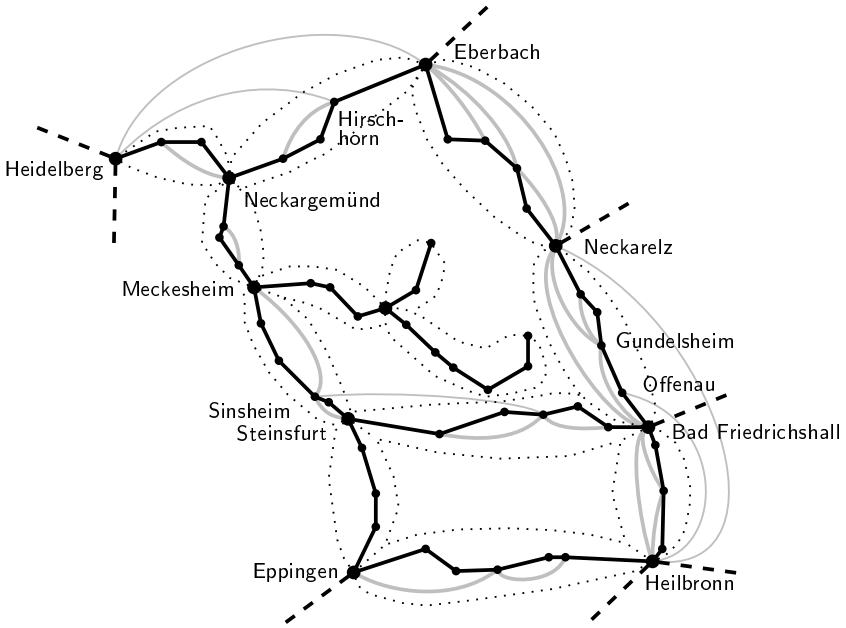

Figure 2.1: The intuition of bundles: Branching points of the physical railroad network (big black vertices) and bundles of edges between them (encircled in dotted lines). The thin edges do not belong to any bundle.

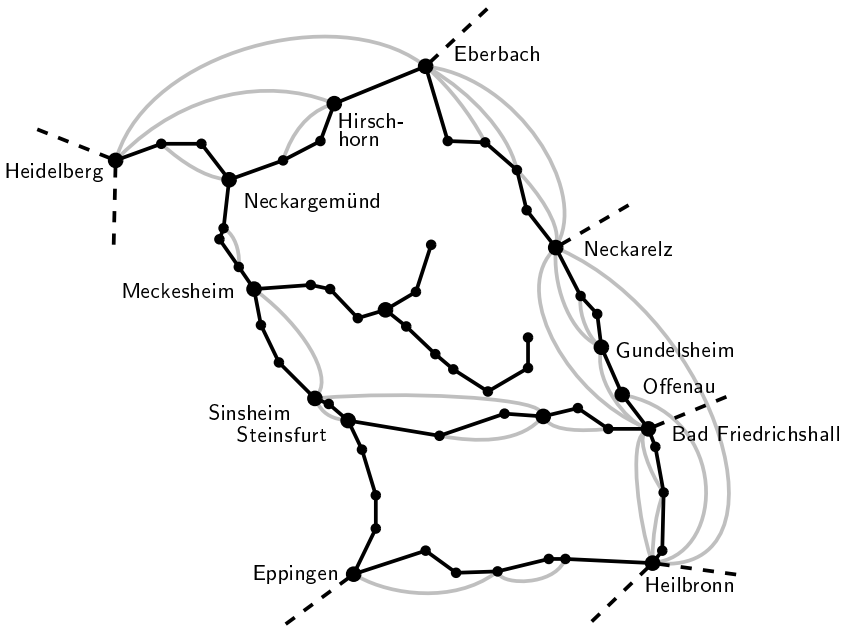

Figure 2.2: A vertex set (big black vertices) of potential bundle end points

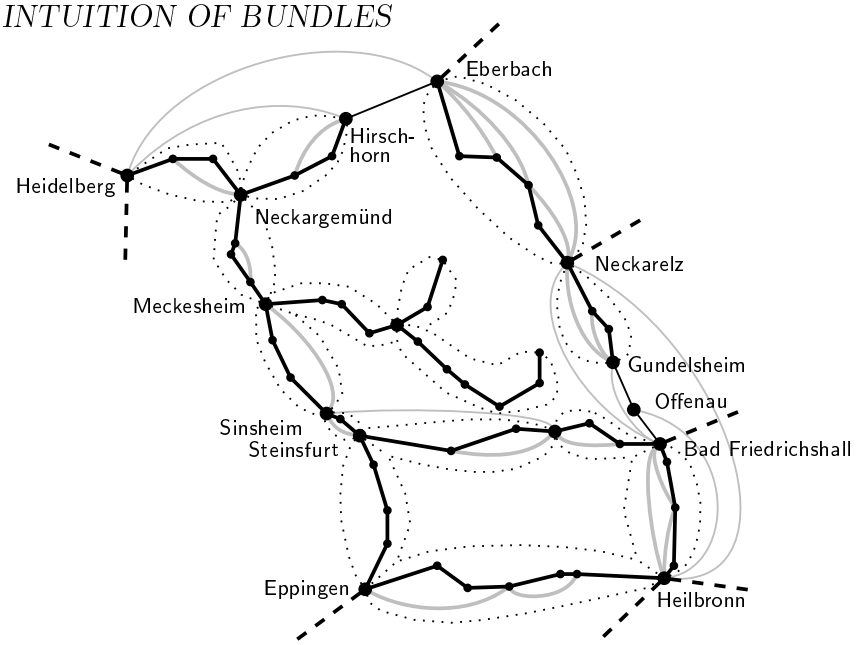

Figure 2.3: The edge partition resulting from applying steps 1. through 3. to Figure 2.2

Notice that steps 1. and 2. simply constitute a sequence of UNION-FIND [CLR90, Chapter 22] operations.

Each resulting bundle in Figure 2.1 forms a directed acyclic graph that contains a Hamilton path. Within each bundle, the edges can now be classified as real or transitive depending on whether or not they belong to the Hamilton path. But edges connecting the end points of a bundle like (Eberbach, Neckarelz) are still singletons in the partition. We can easily assign each such edge to the bundle to which it belongs:

3. For every singleton (a, b) in the partition, if there is a unique edge set A' in the partition that contains a Hamilton path starting with a and ending with b and consisting of at least two edges, then (a, b) is a transitive edge, and the trains traveling along it actually pass through the train stations of the Hamilton path of A' without stopping there.

But in contrast to the ideal scenario described so far, in reality, we of course do not know the bundle end points of a given time table graph, nor do we know which are the edges to be ignored. Now consider Figure 2.2, where more vertices than just the branching points of the railroad network are marked as potential bundle end points. If we start with this vertex set and with the assumption that every edge potentially belongs to some bundle (which is why every edge is drawn in thick lines in Figure 2.2), then the edge partition resulting from steps 1. through 3. forms the edge sets encircled in dotted lines and the singletons drawn in thin lines in Figure 2.3. Within the encircled edge sets, edges are correctly

classified, while edges in singletons remain unclassified by this approach. So by finding a vertex set of potential bundle end points (that possibly contains more vertices than just the branching points of the railroad network) such that the edge partition resulting from steps 1. through 3. consists of at least some bundles (or parts of bundles such as the one from Neckarelz to Gundelsheim in Figure 2.3) and then some singletons, we are able to classify many of the edges in a time table graph. Notice that it is the edges starting or ending in the middle of a bundle like (Offenau, Heilbronn) that cause us to include more vertices than just the branching points of the railroad network in the set of potential bundle end points. As long as time table graphs do not contain too many such edges, this idea for bundle identification seems promising.

We now proceed as follows. Since singletons like (Eberbach, Neckarelz) can be assigned to their edge set with step 3. as a trivial postprocessing step after bundles (and parts of bundles) have been found with steps 1. and 2., we drop step 3. from our considerations for now and concentrate on the core of bundle recognition given by steps 1. and 2. In Section 2.2, we will formulate conditions that edge sets have to fulfill to constitute bundles. We seek this formalization of the intuitive notion of bundles with the following goal in mind: If a vertex set of potential bundle end points is guessed, and if steps 1. and 2. of the above procedure are applied (treating every edge as if it belongs to some bundle), then the resulting edge partition should have the following property: If the edge sets fulfill the conditions for being bundles, and if within each edge set that contains more than one element the edges are classified as real or transitive depending on whether or not they belong to the Hamilton path of the edge set, then there are no wrong classifications. Obviously, there cannot be a proof that our formalization of the notion of a bundle achieves this goal. But the visualization of the heuristic results in Chapters 4 and 5 indicates that it is a workable definition.

Once the notion of a bundle has been formally defined, the bundle recognition problem then consists of finding a vertex subset of a given time table graph such that the resulting edge partition forms bundles (according to the formal definition), and such that there are few singletons in the partition.

2.2 Formalization

Recall the definition of a line graph, sometimes also called edge graph:

Definition 2.1 *Given a directed graph $G = (V, A)$, its* line graph $L = (A, A_{\mathrm{L}})$ *is the directed graph in which every edge of G is a vertex, and in which there is an edge (a, b) if and only if there are three vertices r, s, and t of G so that $a = (r, s)$ and $b = (s, t)$ are edges in G.*

A set of train time tables induces the time table graph $G = (V, A)$ together with a subset A'_{L} of its line graph edge set: If there is a train with train stations r,

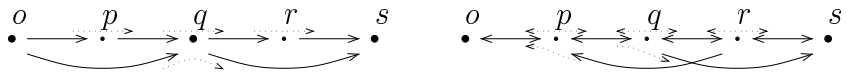

Figure 2.4: A time table graph (black vertices and solid edges) and a subset A'_{L} of line graph edges. *Left:* $V' = \{o, s\}$ would induce two edge sets, one consisting of the four short edges, and one consisting of the two long edges — for note that there are no line graph edges $((p, q), (q, s))$ or $((o, q), (q, r))$ in A'_{L}. Classifying edges along Hamilton paths as real would lead to classifying all six edges as real (which is probably wrong). $V' = \{o, q, s\}$, however, induces four edge sets, among them two singletons, resulting in four edges classified as real and two edges remaining unclassified. Figure 4.18 on page 61 shows a case with such a situation in a concrete time table graph. *Right:* $V' = \{o, s\}$ induces two edge sets that are opposite of each other.

s, and t as consecutive stops, in that order, then the line graph edge $((r, s), (s, t))$ is in A'_{L}. Figure 1.2 shows an example for such a subset of line graph edges in a time table graph.

Definition 2.2 *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edges, a vertex subset $V' \subseteq V$* induces *a partition of A by the following procedure:*

1. *Initially, every edge of A is its own set.*
2. *For every line graph edge $((r, s), (s, t)) \in A'_{\mathrm{L}}$, if s is not in V', then unite the edge sets that (r, s) and (s, t) belong to.*

We still need to specify which conditions the edge sets of a directed graph $G = (V, A)$ induced by a line graph edge subset A'_{L} and a vertex subset $V' \subseteq V$ have to fulfill to form bundles that are suitable for our edge classification approach. To begin with, such an edge set should form a directed acyclic graph that contains a Hamilton path. And since we classify edges along Hamilton paths as real, a situation as depicted on the left side of Figure 2.4 would lead to wrongly classifying two edges as real. We can avoid such wrong classifications by demanding that two different bundles do not share vertices outside V', unless, of course, the two bundles are *opposite* of each other as defined below and as illustrated on the right side of Figure 2.4.

We are now ready to cast the intuitive notion of opposite edge sets into a formal definition, and to state the bundle recognition problem as a graph theoretic problem. Consider the partial order induced by the transitive closure of a directed acyclic graph:

Definition 2.3 *For a directed acyclic graph G, define a partial order "$\leq$" on its vertex set by $r \leq s$ for two (not necessarily distinct) vertices of G if and only if there is a directed path (possibly of length zero) from r to s in G. If neither $r \leq s$ nor $s \leq r$ for two vertices r and s, we write $r||s$.*

Definition 2.4 *For a directed graph $G = (V, A)$ and a subset $A' \subseteq A$ of its edge set, let $G[A']$ denote the directed graph induced by A'. Given two disjoint edge subsets $A_1 \subseteq A$ and $A_2 \subseteq A$ of G such that $G[A_1]$ and $G[A_2]$ are acyclic, A_1 and A_2 are called* opposite *if the following two conditions hold:*

1. *There is an edge $(r, s) \in A_1$ such that $(s, r) \in A_2$.*

2. *If r and s are two distinct vertices belonging both to $G[A_1]$ and to $G[A_2]$, and if $r \leq s$ in $G[A_1]$, then either $s \leq r$ or $s||r$ in $G[A_2]$.*

Definition 2.5 (Bundle Recognition Problem) *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edge set, the* bundle recognition problem *consists of finding a vertex set $V' \subseteq V$ such that the sets of the induced edge partition fulfill the following three conditions:*

1. *Each edge set induces a directed acyclic graph.*

2. *This graph contains a Hamilton path.*

3. *Two distinct edge sets do not have vertices outside V' in common, except possibly if the edge sets are opposite.*

From now on, if a vertex set $V' \subseteq V$ solves the bundle recognition problem, then we call the edge sets of the induced partition *bundles*. Bundles that are not singletons are called *nontrivial*. Note that a solution to the bundle recognition problem will consist in a subset V' of the vertices of the given time table graph, and that the whole vertex set $V' := V$ of a time table graph will always constitute a (trivial) solution. But this solution is useless because it induces an edge partition consisting only of singletons, and our goal is to minimize the number of induced singletons. Since adding a vertex to a solution of the bundle recognition problem cannot reduce the number of induced singletons, minimizing the cardinality of V' seems to be a reasonable optimization criterion as well. Note, however, that a minimum cardinality set V' solving the bundle recognition problem does not necessarily yield the smallest possible number of induced singletons. In fact, there are instances where all vertex sets solving the bundle recognition problem and inducing the minimum number of singletons possible contain more vertices than a vertex set of minimum cardinality. Figure 2.2 shows a simple example.

In Chapter 3, we will see that finding a solution for the bundle recognition problem that contains a small number of vertices or that induces a small number of singletons is NP-hard in a very strong sense: It is already NP-complete to decide whether there is any other solution besides the trivial one.

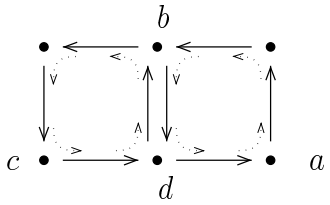

Figure 2.5: $V_1' = \{a, b, c\}$ solves the bundle recognition problem in this instance and induces no singletons. $V_2' = \{b, d\}$ also solves the bundle recognition problem, contains only two vertices instead of three, but induces two singletons, and there is no vertex subset solving the bundle recognition problem that induces no singletons and at the same time contains only two vertices.

2.3 Variations of the Bundle Recognition Problem

The NP-hardness proof for the bundle recognition problem in Section 3.1 will make extensive use of situations where for one edge set A', there are several edge sets to which A' is opposite as illustrated in Figure 2.6. Considering the motivation of the bundle recognition problem through time table graphs, one might argue that this situation is rather artificial and far removed from the original problem of finding bundles along a line of railroad tracks. Also, since it is very hard to solve the bundle recognition problem with a small vertex set or such that a small number of singletons is induced, we hope that considering a variation of the bundle recognition problem will yield a graph theoretic optimization problem that is less hard. So let us consider a variation of the bundle recognition problem where we do not allow an edge set to have more than one opposite edge set, i.e. a version of the bundle recognition problem with an extra condition on opposites, OC-bundle recognition problem for short:

Definition 2.6 (OC-Bundle Recognition Problem) *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edge set, find a vertex set $V' \subseteq V$ such that the sets of the induced edge partition fulfill the following four conditions:*

1. *Each edge set induces a directed acyclic graph.*
2. *This graph contains a Hamilton path.*
3. *Two distinct edge sets do not have vertices outside V' in common, except possibly if the edge sets are opposite.*
4. *No edge set has more than one opposite edge set.*

Unfortunately, we will see in Section 3.2 that it is still NP-hard to find a minimum cardinality vertex set V' solving the OC-bundle recognition problem

Figure 2.6: $V' = \{o, r\}$ (and actually $V' = \emptyset$ just the same) induces three edge sets, one consisting of the simple directed o-r-path, one consisting of the simple directed p-o-path, which is opposite of the o-r-path, and one consisting of the simple directed r-q-path, which is also opposite of the o-r-path.

(but we do not know anything about the complexity of finding a solution that is minimal with respect to inclusion). The proof again extends to the alternative optimization criterion, i.e. it is NP-hard to find a vertex set V' solving the OC-bundle recognition problem in such a way that the number of induced singletons is minimum.

The NP-hardness proofs for the OC-bundle recognition problem in Section 3.2 will use extensively situations where for an edge set A' with one opposite edge set A'', the Hamilton path of A' is not the reversal of the Hamilton path of A'', as illustrated in Figure 2.7. Still hoping that by considering a variation of the bundle recognition problem we might find a formulation of an optimization problem that admits a polynomial time solution algorithm, we will consider an even more restricted notion of bundles where each edge set has not only at most one opposite edge set, but where an edge set and its opposite must have their entire Hamilton paths in common, of course in reverse order:

Definition 2.7 (OHC-Bundle Recognition Problem) *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edge set, find a vertex set $V' \subseteq V$ such that the sets of the induced edge partition fulfill the following five conditions:*

1. *Each edge set induces a directed acyclic graph.*
2. *This graph contains a Hamilton path.*
3. *Two distinct edge sets do not have vertices outside V' in common, except possibly if the edge sets are opposite.*
4. *No edge set has more than one opposite edge set.*
5. *If an edge set A' and an edge set A'' are opposite, and if $v_0, v_1, \ldots, v_k$ denotes the Hamilton path of A', then $v_k, \ldots, v_1, v_0$ is the Hamilton path of A''.*

Note that one could argue that the OHC-bundle recognition problem now corresponds even more closely to the intuitive notion of bundles as lines of railroad tracks than the OC-bundle recognition problem did. But our hope of finding a problem formulation together with a polynomial time solution algorithm is still not fulfilled: In Section 3.3, we will see that it is again NP-hard to minimize the

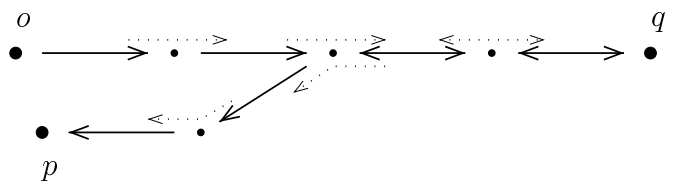

Figure 2.7: $V' = \{o, p, q\}$ (and $V' = \emptyset$ just the same) induces two edge sets, one consisting of the simple directed o-q-path, and one consisting of the simple directed q-p-path. They are opposite of each other, but their Hamilton paths are not the reverse of each other.

number of vertices in a solution V', or to minimize the number of singletons induced by a solution V'. But again, we do not know anything about the complexity of finding a solution that is minimal with respect to inclusion.

In the formulation of the OHC-bundle recognition problem, situations where a vertex in a solution V' is at the same time an inner vertex of the Hamilton path of some bundle, as illustrated in Figure 2.8, are still allowed. And again, they are an integral part of the NP-hardness proofs in Section 3.3. To require that vertices in a solution V' do not occur as inner vertices of Hamilton paths of nontrivial bundles yields yet another version of the bundle recognition problem:

Definition 2.8 (OHIC-Bundle Recognition Problem) *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edge set, find a vertex set $V' \subseteq V$ such that the sets of the induced edge partition fulfill the following six conditions:*

1. *Each edge set induces a directed acyclic graph.*
2. *This graph contains a Hamilton path.*
3. *Two distinct edge sets do not have vertices outside V' in common, except possibly if the edge sets are opposite.*
4. *No edge set has more than one opposite edge set.*
5. *If an edge set A' and an edge set A'' are opposite, and if $v_0, v_1, \ldots, v_k$ denotes the Hamilton path of A', then $v_k, \ldots, v_1, v_0$ is the Hamilton path of A''.*
6. *No vertex of V' occurs as an inner vertex of a Hamilton path of a nontrivial bundle.*

Unfortunately, we know nothing about the complexity of solving the OHIC-bundle recognition problem, and it remains open whether an optimal solution for this variation of the bundle recognition problem can be found efficiently or not.

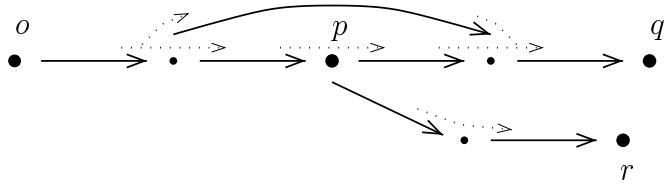

Figure 2.8: $V' = \{o, p, q, r\}$ (and $V' = \{p\}$ just the same) induces two edge sets, one with a Hamilton path starting at o and ending at q, and one with a Hamilton path starting at p and ending at r. So p belongs to V' (and is the first vertex of one of the two Hamilton paths) but is an inner vertex of the Hamilton path starting at o and ending at q.

Chapter 3

NP-Completeness Proofs

To prove the NP-completeness results, we will repeatedly use the following basic arguments. They are formulated for the bundle recognition problem, but they also hold for the OC-bundle recognition problem and for the OHC-bundle recognition problem.

Assume we are given an instance $(G = (V, A), A'_{\mathrm{L}})$ of the bundle recognition problem together with a solution $V' \subseteq V$.

(a) If for an edge $(s, t) \in A$ the edge (t, s) is not in A, if there is no edge $(r, s) \in A$ with $((r, s), (s, t)) \in A'_{\mathrm{L}}$, and if there is no edge $(t, u) \in A$ with $((s, t), (t, u)) \in A'_{\mathrm{L}}$, then in the induced edge partition, $\{(s, t)\}$ is a singleton that is not opposite to any other edge set. So by the third condition in the formulation of the bundle recognition problem, it may not have vertices outside of V' in common with any other edge set, and therefore s and t must belong to V' (unless they are not incident to any other edge besides (s, t)). In Figure 3.1, the vertices o, p, and q must belong to V' because of this argument.

(b) If G contains a directed cycle of n edges, and if A'_{L} contains the n line graph edges connecting consecutive edges of the cycle of G, then at least two vertices of the cycle must belong to V', for otherwise all n cycle edges would belong to the same edge set in the induced edge partition, contradicting that the edge sets must be acyclic (first condition in the formulation of the bundle recognition problem). So in Figure 3.1, at least two of the four white vertices a, b, c, and d of the cycle on the right must be in V'.

(c) Consider again the left side of Figure 3.1: If $r \in V'$, then the edges (r, t) and (s, t) belong to different sets of the induced edge partition, and these two edge sets are not opposite of each other. So again because of the third condition in the formulation of the bundle recognition problem, $r \in V'$ implies $t \in V'$. Now assume $t \in V'$ but $s \notin V'$. Then the edges (s, t) and (s, v) belong to the same edge set in the induced edge partition, but this edge set does not contain a Hamilton path, contradicting the second condition in the formulation of the bundle recognition problem. So $t \in V'$ implies $s \in V'$. Similarly, $s \in V'$ implies $r \in V'$.

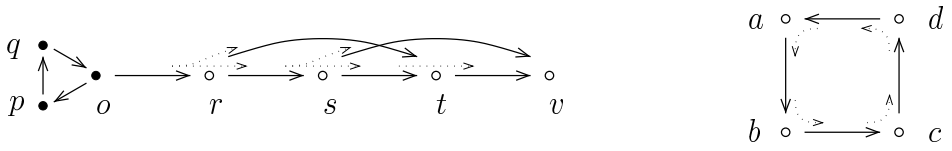

Figure 3.1: An instance of the bundle recognition problem. The directed graph G is shown with black and white vertices and black edges, and the set of line graph edges A'_{L} is shown in dotted lines. This instance is used to illustrate basic arguments when reasoning about solutions to the bundle recognition problem.

Together, these implications mean that either all of r, s, and t belong to V', or none of them does. Therefore, we call $\{r, s, t\}$ an *all-or-none-set*:

Definition 3.1 *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edge set, a vertex set $V'' \subseteq V$ is an* all-or-none-set *if for every solution $V' \subseteq V$ to the bundle recognition problem, either $V'' \subseteq V'$, or $V'' \cap V' = \emptyset$.*

In the remainder of this chapter, we will use the drawing convention of Figure 3.1 when arguing about solutions to a given instance of the bundle recognition problem: Vertices that must belong to every solution to a given instance will be drawn in black, while vertices that may or may not belong to a solution will be drawn in white. Line graph edges will be dotted.

3.1 The Bundle Recognition Problem

Finding a solution to the bundle recognition problem that contains a small number of vertices or that induces a small number of singletons is NP-hard in a very strong sense: It is already NP-complete to decide whether there is any other solution besides the trivial one.

Theorem 3.1(a) *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edge set, it is NP-complete to decide whether there is any proper subset $V' \subset V$ solving the bundle recognition problem.*

Proof: Clearly the bundle recognition problem is in NP. We use a reduction from the problem 3-Satisfiability (3Sat; see [Coo71] and [GJ79, Problem LO2]) to see that it is NP-complete. Given a 3Sat instance consisting of a set U of Boolean variables and a set C of clauses over U with exactly three literals per clause, we may assume that U contains at least two variables.We construct an instance $(G = (V, A), A'_{\mathrm{L}})$ of the bundle recognition problem as follows:

For each variable $u \in U$ there are two *literal chains*, one for the positive literal and one for the negative literal, as defined by Figure 3.2. Furthermore,

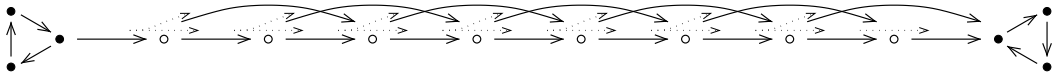

Figure 3.2: An example of a literal chain as used in the construction for the proof of Theorem 3.1(a). For a literal appearing in ℓ clauses, the corresponding literal chain is constructed to contain 6 black vertices, $6 + 2\ell$ white vertices, $18 + 4\ell$ edges, and $12 + 4\ell - 1$ line graph edges. Each literal chain is vertex and edge disjoint from every other literal chain.

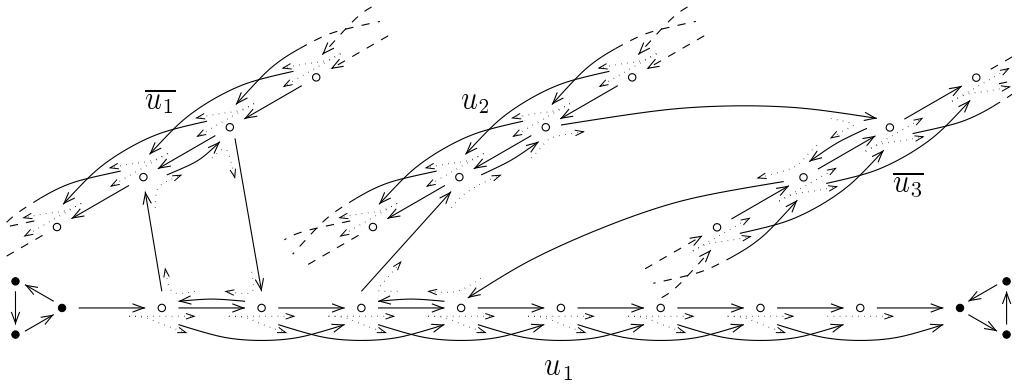

Figure 3.3: For each variable u in the construction for the proof of Theorem 3.1(a), the two literal chains for u and $\overline{u}$ are connected by a cycle of four edges and four line graph edges, called a *variable cycle*. The picture shows the variable cycle for u_1. Each variable cycle is vertex and edge disjoint from every other variable cycle.

For each clause c in the construction for the proof of Theorem 3.1(a), the literal chains corresponding to the three literals making up c are connected by a *clause cycle* of six edges and six line graph edges. The picture shows the clause cycle for a clause $c = (u_1 \vee u_2 \vee \overline{u_3})$. Each clause cycle is vertex and edge disjoint from every other clause cycle and from every variable cycle.

the two literal chains belonging to a variable are connected by a *variable cycle* as depicted in Figure 3.3. Abusing notation, the positive literal for the variable u is also denoted u. The negative literal is denoted $\overline{u}$. For each clause $c \in C$ there is a *clause cycle* as defined by Figure 3.3, connecting the literal chains corresponding to the three literals forming c.

If U consists of the n variables $u_1, \ldots, u_n$, there is a *forcing component* as defined by Figure 3.4 from the two literal chains corresponding to u_i to the two literal chains corresponding to u_{i+1} for each $1 \leq i \leq n - 1$, and there is a forcing component from the literal chains corresponding to u_n to the literal chains corresponding to u_1.

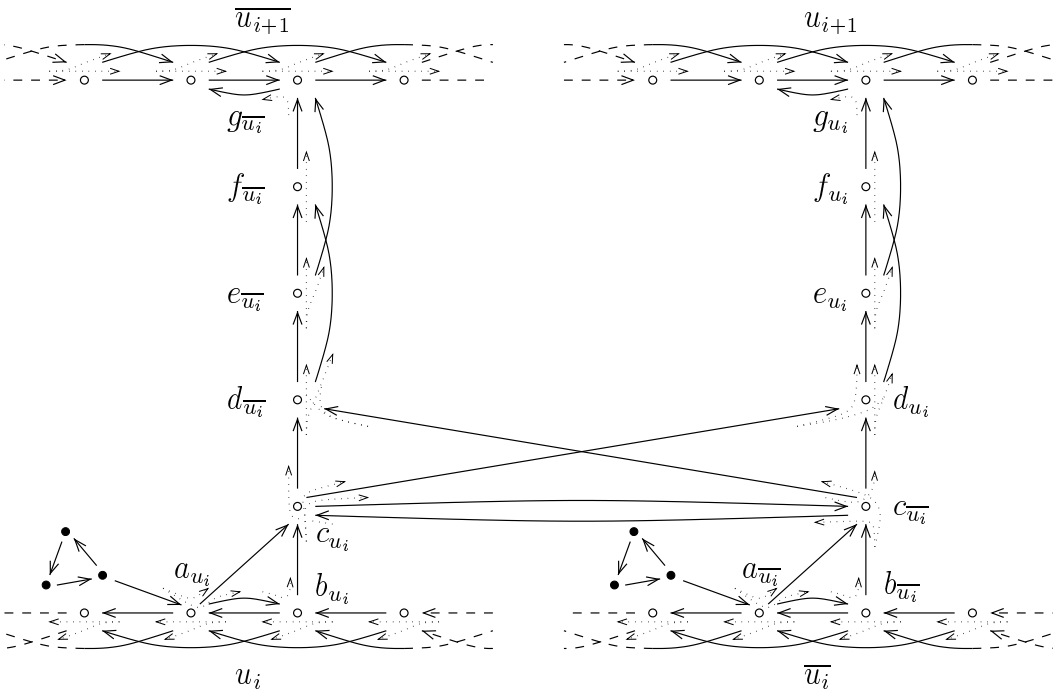

Figure 3.4: For each variable in the construction for the proof of Theorem 3.1(a), there is a forcing component connecting the two literal chains of the variable to the literal chains of another variable. Each of the $|U|$ forcing components consists of six black vertices, eight white vertices (besides the white vertices of literal chains that are part of the forcing component), 32 edges, and 28 line graph edges. The line graph edges at vertex c_{u_i} are $((c_{\overline{u_i}}, c_{u_i}), (c_{u_i}, d_{\overline{u_i}}))$, $((b_{u_i}, c_{u_i}), (c_{u_i}, c_{\overline{u_i}}))$, and $((b_{u_i}, c_{u_i}), (c_{u_i}, d_{u_i}))$. The line graph edges at vertex $d_{\overline{u_i}}$ are $((c_{u_i}, d_{\overline{u_i}}), (d_{\overline{u_i}}, e_{\overline{u_i}}))$, $((c_{u_i}, d_{\overline{u_i}}), (d_{\overline{u_i}}, f_{\overline{u_i}}))$, $((c_{\overline{u_i}}, d_{\overline{u_i}}), (d_{\overline{u_i}}, e_{\overline{u_i}}))$, and $((c_{\overline{u_i}}, d_{\overline{u_i}}), (d_{\overline{u_i}}, f_{\overline{u_i}}))$. The situation at $c_{\overline{u_i}}$ and d_{u_i} is symmetric. Every forcing component is vertex and edge disjoint from every other forcing component, and from every variable and clause cycle.

We will make use of the following lemmas:

Lemma 3.1.1 *In this instance of the bundle recognition problem, there are* $18 \cdot |U|$ *vertices (the black vertices of the literal chains and of the forcing components in Figures 3.2 and 3.4) that every solution* V' *to the bundle recognition problem must contain.*

Lemma 3.1.2 *Within every literal chain, the set of white vertices forms an all-or-none-set.*

Lemmas 3.1.1 and 3.1.2 follow immediately with the basic arguments given in the beginning of this chapter.

Lemma 3.1.3 *If for a forcing component attached to the literal chain x by the vertices a_x and b_x as shown in Figure 3.4, a solution V' to the bundle recognition problem contains a_x or b_x, then V' contains all three vertices a_x, b_x, and c_x.*

To see this, assume that V' contains a_x or b_x. Then by Lemma 3.1.2 it contains both a_x and b_x, and $\{(a_x, c_x)\}$ is a singleton in the induced edge partition. Then $\{(a_x, c_x)\}$ and the induced edge set containing (b_x, c_x) have c_x in common but are not opposite. So by the third condition in the formulation of the bundle recognition problem, c_x must belong to V' as well.

Lemma 3.1.4 *Consider a variable u and the vertices c_u and $c_{\overline{u}}$ of the corresponding forcing component as shown in Figure 3.4. If both c_u and $c_{\overline{u}}$ are contained in a solution V' to the bundle recognition problem, then the vertices d_u, e_u, f_u, g_u, $d_{\overline{u}}$, $e_{\overline{u}}$, $f_{\overline{u}}$, and $g_{\overline{u}}$ all belong to V' as well.*

For a proof, assume $d_u \notin V'$. Then in the induced edge partition, (c_u, d_u) and $(c_{\overline{u}}, d_u)$ belong to the same edge set. But this set does not have a Hamilton path, because within this set, there is no path from c_u to $c_{\overline{u}}$ or from $c_{\overline{u}}$ to c_u. So $d_u \in V'$. Then $\{(d_u, f_u)\}$ is a singleton in the induced edge partition, and $\{(d_u, f_u)\}$ and the induced edge set containing (e_u, f_u) have f_u in common without being opposite. So $f_u \in V'$ because of the third condition in the formulation of the bundle recognition problem. But $f_u \in V'$ implies $e_u \in V'$, because otherwise, in the induced edge partition, (e_u, f_u) and (e_u, g_u) would belong to the same edge set, and this edge set would not contain a Hamilton path. So $e_u \in V'$. Finally, again with the third condition in the formulation of the bundle recognition problem, it follows that $g_u \in V'$. The analogous argument holds for $d_{\overline{u}}$, $e_{\overline{u}}$, $f_{\overline{u}}$, and $g_{\overline{u}}$.

To prove Theorem 3.1(a), we need to show that there is a fulfilling truth assignment for the variables of the 3SAT instance if and only if there is a solution V' to the bundle recognition problem that is a proper subset of V. First assume we have a fulfilling truth assignment for the 3SAT instance. We construct a solution V' to the bundle recognition problem as follows. Include all black vertices in V'. For each literal x that the truth assignment sets true, include the white vertices of its literal chain in V', and include in V' the vertex c_x of the corresponding forcing component. Clearly, this set V' is a proper subset of V. It is also a solution to the bundle recognition problem: In particular, each literal chain consists of either six singletons and one nontrivial bundle (if the corresponding literal is false), or it consists of only singletons (if the corresponding literal is true). Since for every variable, exactly one of its two literals has the white vertices of its literal chain in V', no cyclic edge set is induced by the variable cycles. And since for each clause, at least one literal is true and therefore its corresponding literal chain has its white vertices in V', no cyclic edge set is induced by the clause cycles either. Finally, each forcing component consists of 11 singletons and two nontrivial bundles as illustrated in Figure 3.5.

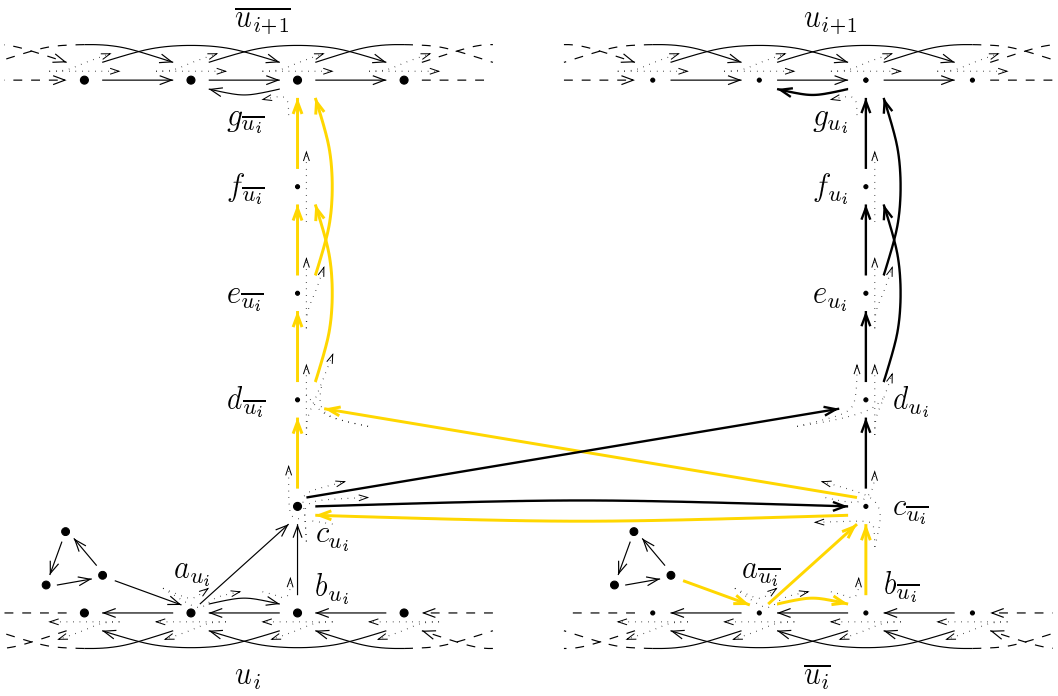

Figure 3.5: An example for 11 singletons and two nontrivial bundles in a forcing component induced by the solution V' to the bundle recognition problem that is constructed from a fulfilling truth assignment for the given 3SAT instance in the proof of Theorem 3.1(a). V' is shown in big black dots. The two nontrivial bundles are shown in thick black and yellow (grey) lines.

Now assume we are given a solution V' to the bundle recognition problem that is a proper subset of V. We claim that such a set V' yields a fulfilling truth assignment for the 3SAT instance by setting each variable to true for which the white vertices of the positive literal chain are contained in V', and by setting every variable to false for which the white vertices of the negative literal chain are contained in V'.

We first need to show that for each variable, exactly one of its two literal chains has its white vertices contained in V'. First recall that by Lemma 3.1.2, for each literal chain, either all its white vertices are in V' or none is. Now assume there is a variable u for which neither of its two literal chains has its white vertices in V'. But then the four edges of the variable cycle for u form an induced edge set that is cyclic, contradicting that V' is a solution to the bundle recognition problem. So for each variable, at least one of its literal chains has its white vertices in V'. Now assume there is a variable u for which both of its literal chains have their white vertices in V'. Applying Lemmas 3.1.3, 3.1.4, and 3.1.2 repeatedly yields that all white vertices of all literal chains and all white vertices of all forcing components must be in V'. But then $V' = V$, contradicting that

V' is a proper subset of V. So we have that for each variable, exactly one of its literal chains has their white vertices in V', and thus V' indeed induces a truth assignment for the variables.

To show that this truth assignment is fulfilling, assume that it leaves a clause c with only false literals. Then the six edges of the clause cycle corresponding to c form an induced edge set that is cyclic, contradicting that V' is a solution to the bundle recognition problem. So for each clause, at least one of its literals is set to true. This concludes the proof of Theorem 3.1(a).

Notice that in the instances of the bundle recognition problem constructed for this proof, there is a solution $V' \neq V$ if and only if there is a solution so that the induced edge sets form less than $|A|$ singletons. So corresponding to Theorem 3.1(a), we have:

Theorem 3.1(b) *Given a directed graph $G = (V, A)$ and a subset A'_{L} of its line graph edge set, it is NP-complete to decide whether there is any vertex set $V' \subseteq V$ solving the bundle recognition problem and inducing less than $|A|$ singletons.*

3.2 The OC-Bundle Recognition Problem

Theorem 3.2(a) *Given a directed graph $G = (V, A)$, a subset A'_{L} of its line graph edge set, and an integer K, it is NP-complete to decide whether there is a vertex set $V' \subseteq V$ with $|V'| \leq K$ that solves the OC-bundle recognition problem.*

Proof: First redefine all-or-none-sets in the appropriate way for OC-bundle recognition problems: From now on, an *all-or-none-set* is a vertex set V'' such that for every solution V' to the OC-bundle recognition problem, either $V'' \subseteq V'$, or $V'' \cap V' = \emptyset$.

The OC-bundle recognition problem is again in NP. We use a reduction from the following variation of One-In-Three 3Sat: Given a set U of Boolean variables and a set C of clauses over U where each clause contains exactly three positive literals, is there a truth assignment for the variables so that there is exactly one true literal in each clause ? [Sch78] shows that One-In-Three 3Sat is NP-complete, and [GJ79, Problem LO4] observes that it remains NP-complete even if there are no negative literals. This can be easily seen:

Lemma 3.2.1 One-In-Three 3Sat *remains NP-complete even if there are no negative literals.*

To show the lemma, first observe that if in an instance of One-In-Three 3Sat there is a clause $(x \vee y \vee y)$ for two Boolean variables x and y, then in every fulfilling truth assignment for this instance, x must be set to true, and y must be set to false. Also observe that if in an instance of One-In-Three 3Sat there is a

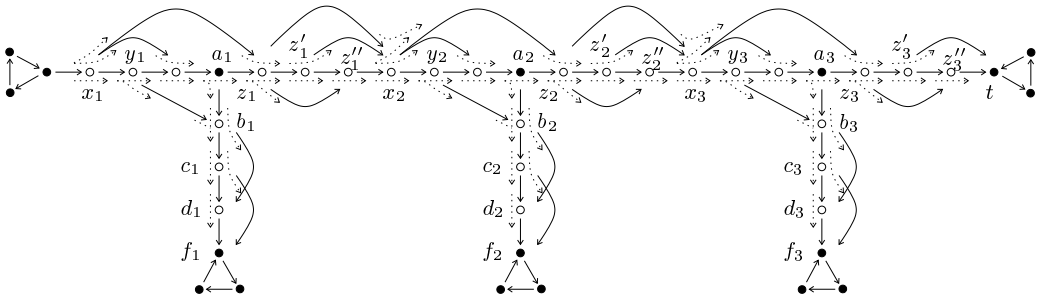

Figure 3.6: An example of a variable component as used for the proof of Theorem 3.2(a). If the positive literal of a variable appears ℓ times in clauses, then the corresponding variable component has ℓ *arms* and consists of $6 + 4\ell$ black vertices, 9ℓ white vertices, $6 + 22\ell$ edges, and $22\ell - 1$ line graph edges. The picture shows an example with three arms.

clause $(y \vee a \vee b)$ for three Boolean variables y, a, and b, then if in a fulfilling truth assignment for this instance the variable y is set to false, the variables a and b must be set so that $a = \overline{b}$. Given an instance (U, C) of One-In-Three 3Sat, we construct an instance (U', C') of One-In-Three 3Sat without negative literals as follows. For every variable $u \in U$, let U' contain u and a new variable u'. Let U' also contain two additional variables x and y, so we have $|U'| = 2 \cdot |U| + 2$. For every clause $c \in C$, let C' contain a clause c' in which every occurrence of a negative literal $\overline{u}$ for a variable u is replaced by the positive literal u'. Let C' also contain the additional clause $(x \vee y \vee y)$, and for every variable $u \in U$, let C' contain the clause $(y \vee u \vee u')$. So we have $|C'| = |C| + |U| + 1$. Now assume that there is a fulfilling truth assignment for (U, C). Then setting x to true, y to false, and $u' := \overline{u}$ for every variable $u \in U$ clearly yields a fulfilling truth assignment for (U', C'). Conversely assume that there is a fulfilling truth assignment for (U', C'). It must set x to true and y to false because of the clause $(x \vee y \vee y)$, and thus for every $u \in U$ it must set $u' := \overline{u}$ because of the clause $(y \vee u \vee u')$. Its truth assignment to the variables u of U' that are also in U is therefore a fulfilling truth assignment for the original instance (U, C) of One-In-Three 3Sat, and the proof of the lemma is complete.

Given an instance (U, C) of One-In-Three 3Sat without negative literals, we now construct an instance $(G = (V, A), A'_{\mathrm{L}}, K)$ of the OC-bundle recognition problem as follows: For each variable $u \in U$ there is a *variable component* as defined by Figure 3.6, and if the positive literal u appears ℓ times in clauses of C, then the variable component has ℓ *arms*. For each clause c in C there is a *clause component* as defined by Figure 3.7. It connects three arms of variable components corresponding to the three literals making up c. Set $K = 6 \cdot |U| + 28 \cdot |C|$.

We will make use of the following lemmas:

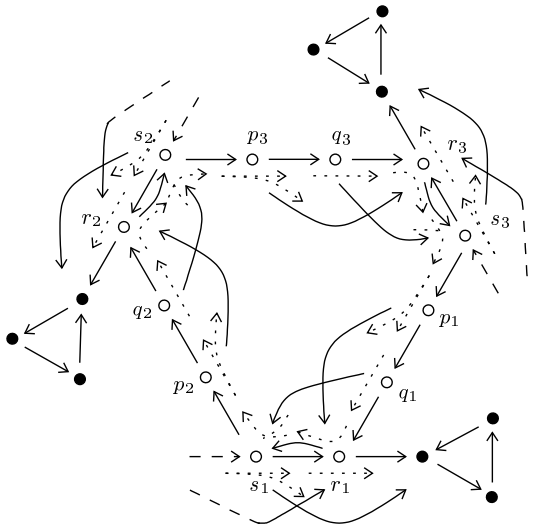

Figure 3.7: For each clause, there is a clause component connecting three arms of the variable components corresponding to the literals making up the clause. A clause component contains 6 white vertices (besides the white vertices that are part of the arms of variable components), 18 edges, and 18 line graph edges. Note that each vertex labeled r_i for some $i \in \{1, 2, 3\}$ is called d_j for some integer $j > 0$ in an arm of a variable component. Likewise, each s_i has another label c_j.

Lemma 3.2.2 *In this instance of the OC-bundle recognition problem, there are* $6 \cdot |U| + 12 \cdot |C|$ *vertices (the black vertices of the variable components) that every solution to the OC-bundle recognition problem must contain.*

To see this, observe that the black vertices in the 3-cycles of the variable components must be in any solution to the OC-bundle recognition problem because of the basic arguments given in the beginning of this chapter. And since there are as many arms of variable components as there are literals in clauses, and since there are exactly $3 \cdot |C|$ literals, there are $6 \cdot |U| + 9 \cdot |C|$ such black vertices. Also, the $3 \cdot |C|$ vertices labeled a_i for some positive integer i must be in any solution to the OC-bundle recognition problem, because otherwise there would be a variable component with an induced edge set containing both (a_i, z_i) and (a_i, b_i), and such an edge set cannot have a Hamilton path.

Lemma 3.2.3 *If a variable component has* ℓ *arms, then the vertex set* $\{x_1, z_1, z'_1, z''_1, \ldots, x_\ell, z_\ell, z'_\ell, z''_\ell\}$ *forms an all-or-none-set.*

For a proof, let V' be a solution to the OC-bundle recognition problem, consider a variabe component with ℓ arms, and observe the following:

(a) For $1 \leq i \leq \ell$: $\{x_i, z_i\}$ is an all-or-none-set (because if one of the two vertices belongs to V' but not the other, then the edge set containing (x_i, z_i) in the induced edge partition does not contain a Hamilton path).

(b) For $1 \leq i \leq \ell$: $z_i \in V'$ implies $z_i'' \in V'$ (because if $z_i \in V'$, then the singleton $\{(z_i, z_i'')\}$ and the edge set containing (z_i', z_i'') in the induced edge partition have z_i'' in common without being opposite).

(c) For $1 \leq i \leq \ell$: $z_i'' \in V'$ implies $z_i' \in V'$ (because if $z_i'' \in V'$ and $z_i' \notin V'$, then for $i < \ell$ the edges (z_i', z_i'') and (z_i', x_{i+1}), and for $i = \ell$ the edges (z_i', z_i'') and (z_i', t) are in the same edge set in the induced edge partition, and this edge set does not contain a Hamilton path).

(d) For $1 \leq i \leq \ell$: $z_i' \in V'$ implies $z_i \in V'$ (because if $z_i' \in V'$ and $z_i \notin V'$, then the edges (z_i, z_i') and (z_i, z_i'') are in the same edge set in the induced edge partition, and this edge set does not contain a Hamilton path).

(e) For $1 \leq i \leq \ell - 1$: $z_i' \in V'$ implies $x_{i+1} \in V'$ (because if $z_i' \in V'$, then the singleton $\{(z_i', x_{i+1})\}$ and the edge set containing (z_i'', x_{i+1}) in the induced edge partition have x_{i+1} in common without being opposite).

(f) For $2 \leq i \leq \ell$: $x_i \in V'$ implies $z_{i-1} \in V'$ (because if $x_i \in V'$, then the singleton $\{(z_{i-1}, x_i)\}$ and the edge set containing (z_{i-1}, z_{i-1}') in the induced edge partition have z_{i-1} in common without being opposite).

Lemma 3.2.4 *If a variable component has ℓ arms, then for each $i \in \{1, \ldots, \ell\}$, the vertex set $\{y_i, b_i, c_i, d_i\}$ forms an all-or-none-set.*

To see this, first observe with the basic arguments given in the beginning of this chapter that $\{b_i, c_i, d_i\}$ is an all-or-none-set. Then note that $\{y_i, b_i\}$ is also an all-or-none-set because if one of the two vertices belongs to a solution to the OC-bundle recognition problem but not the other, then the edge set containing (y_i, b_i) in the induced edge partition does not contain a Hamilton path.

Lemma 3.2.5 *In each clause component and for each $i \in \{1, 2, 3\}$, the vertex set $\{p_i, q_i, r_i, s_i\}$ forms an all-or-none-set.*

For a proof, assume V' is a solution to the OC-bundle recognition problem, consider a clause component and $i \in \{1, 2, 3\}$, and observe the following:

(a) $p_i \in V'$ implies $r_i \in V'$ (because if $p_i \in V'$ and $r_i \notin V'$, then the singleton $\{(p_i, r_i)\}$ and the edge set containing (q_i, r_i) in the induced edge partition have r_i in common).

(b) $r_i \in V'$ implies $s_i \in V'$ (because of Lemma 3.2.4).

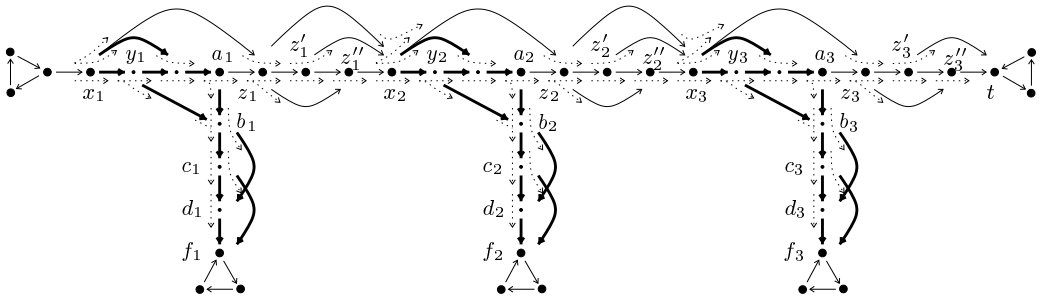

Figure 3.8: The singletons and the nontrivial bundles (one per arm) in a variable component induced by the solution V' to the OC-bundle recognition problem that is constructed from a fulfilling truth assignment for the given One-In-Three 3Sat instance in the proof of Theorem 3.2(a), for the case that the corresponding variable is set to true. V' is shown in big black dots, and the nontrivial bundles are shown in thick lines.

(c) $s_i \in V'$ implies $q_i \in V'$ (because if $s_i \in V'$ and $q_i \notin V'$, then the singleton $\{(q_i, s_i)\}$ and the edge set containing (q_i, r_i) in the induced edge partition have q_i in common).

(d) $q_i \in V'$ implies $p_i \in V'$ (because if $q_i \in V'$ and $p_i \notin V'$, then in the induced edge partition, (p_i, q_i) and (p_i, r_i) belong to the same edge set, and this edge set does not have a Hamilton path).

To show that there is a truth assignment for the variables in U fulfilling the One-In-Three 3Sat instance if and only if there is a solution to the OC-bundle recognition problem with at most K vertices, first assume there is a fulfilling truth assignment. Construct a solution V' to the OC-bundle recognition problem as follows. Let every one of the $6 \cdot |U| + 12 \cdot |C|$ black vertices be in V'. For each variable that is set to true, let the all-or-none-set containing x_1 be in V'. For every variable that is set to false, let the all-or-none-sets of the arms of the variable component be in V'. In either case, this places $12 \cdot |C|$ white vertices in V'. For each clause component, let only those four vertices be in V' that will have to belong to it because of Lemma 3.2.5, i.e. if i and j are the two false literals in a clause, then $\{p_i, q_i, p_j, q_j\} \in V'$. This set V' contains $6 \cdot |U| + 12 \cdot |C| + 12 \cdot |C| + 4 \cdot |C| = K$ vertices, and it is a solution to the OC-bundle recognition problem: Every variable component contains either one nontrivial bundle per arm as shown in Figure 3.8, or it contains exactly one nontrivial bundle as shown in Figure 3.9. Every clause component contains exactly one nontrivial bundle as shown in Figure 3.10. Note that in particular, no bundle has more than one opposite.

Now assume we are given a vertex set V' with at most K vertices that solves the OC-bundle recognition problem. We have the following lemmas:

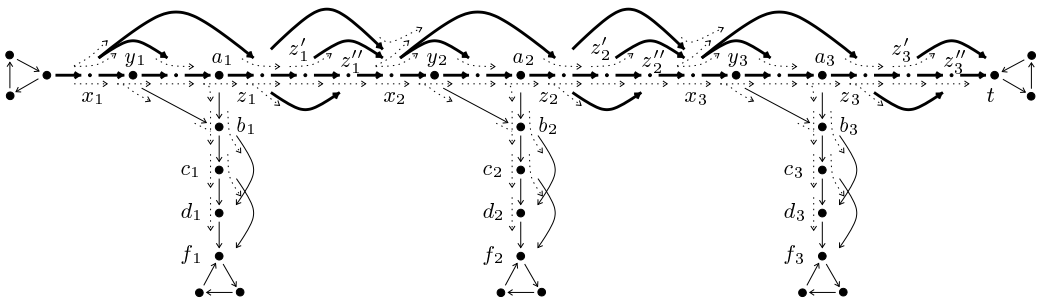

Figure 3.9: Like Figure 3.8, but for the case that the corresponding variable is set to false. The one nontrivial bundle is shown in thick lines.

Lemma 3.2.6 *If the all-or-none-set containing x_1 of a variable component is not in V', then every all-or-none-set corresponding to an arm of the variable component must be in V'. Thus, there are at least $12 \cdot |C|$ white vertices of variable components in V'.*

For a proof, consider a variable component with ℓ arms. First assume that its all-or-none-set containing x_1 is not in V' and that there is an i, $1 \leq i \leq \ell$ for which the all-or-none-set corresponding to arm i is not in V' either. But then there is an edge set in the induced edge partition containing both the edge (x_i, z_i) and the edge (y_i, b_i), and this edge set does not contain a Hamilton path. So if the all-or-none-set containing x_1 is not in V', then every all-or-none-set corresponding to an arm of the variable component must be in V', and there are at least 4ℓ white vertices of the variable component in V'. Note that the all-or-none-set containing x_1 also contains 4ℓ vertices. Since the number of arms counted over all variable components equals $3 \cdot |C|$, we have that there are at least $12 \cdot |C|$ white vertices of variable components in V'.

Lemma 3.2.7 *For each clause component, at least two of its all-or-none-sets must belong to V'.*

To see this, note that if less than two of the all-or-none-sets of a clause component are in V', then the one nontrivial bundle induced within the clause component has more than one opposite edge set.

The number of vertices that must belong to V' according to Lemmas 3.2.2, 3.2.6, and 3.2.7 is $6 \cdot |U| + 12 \cdot |C| + 12 \cdot |C| + 4 \cdot |C| = K$. Therefore, if for a variable component its all-or-none-set containing x_1 belongs to V', then no all-or-none-sets corresponding to arms of this variable component can be in V', and we have a truth assignment for the variables by setting a variable to true if and only if the all-or-none-set containing x_1 belongs to V'. This truth assignment fulfills the One-In-Three 3Sat instance because, again due to the limit of K vertices for V', no clause component can have all three of its all-or-none-sets belong to V'. This concludes the proof of Theorem 3.2(a).

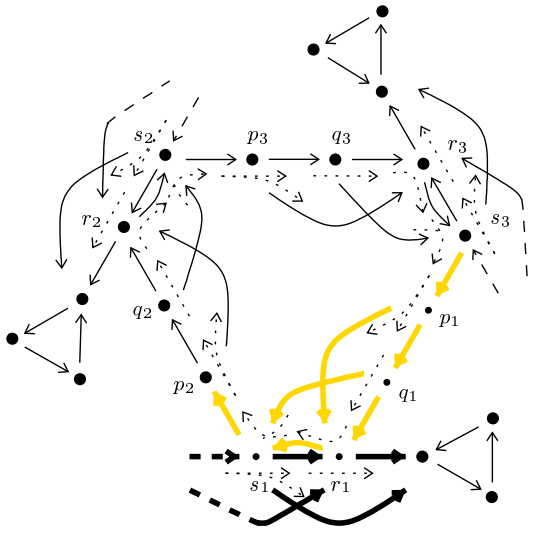

Figure 3.10: The singletons and the nontrivial bundle in a clause component induced by the solution V' to the OC-bundle recognition problem that is constructed from a fulfilling truth assignment for the given One-In-Three 3Sat instance in the proof of Theorem 3.2(a), for the case that the first of the three literals making up the clause is set to true. V' is shown in big black dots, the nontrivial bundle within the clause component is shown in thick yellow (grey) lines, and the nontrivial bundle in the arm of the variable component for the variable that is set to true is shown in thick black lines.

As with Theorems 3.1(a) and 3.1(b), we can again obtain a result corresponding to Theorem 3.2(a) where the optimization criterion is the number of induced singletons rather than the number of vertices in the vertex set V':

Theorem 3.2(b) *Given a directed graph $G = (V, A)$, a subset A'_{L} of its line graph edge set, and an integer K, it is NP-complete to decide whether there is a vertex set $V' \subseteq V$ that induces at most K singletons, and that solves the OC-bundle recognition problem.*

Proof: The proof uses the same ideas as the proof of Theorem 3.2(a) but with a slightly altered construction of the instance $(G = (V, A), A'_{\mathrm{L}}, K)$:

For every variable component with two or more arms, prolong the first arm as shown in Figure 3.11: Insert an extra white vertex e_1, add two edges (d_1, e_1) and (e_1, f_1), add two line graph edges $((c_1, d_1), (d_1, e_1))$ and $((d_1, e_1), (e_1, f_1))$, and replace the edge (c_1, f_1) by (c_1, e_1), so that $\{y_1, b_1, c_1, d_1, e_1\}$ forms an all-or-none-set.

For every variable component with $\ell \geq 3$ arms, add to every "middle" arm i with $2 \leq i < \ell$ one edge (b_i, f_i), and add one line graph edge $((a_i, b_i), (b_i, f_i))$ so

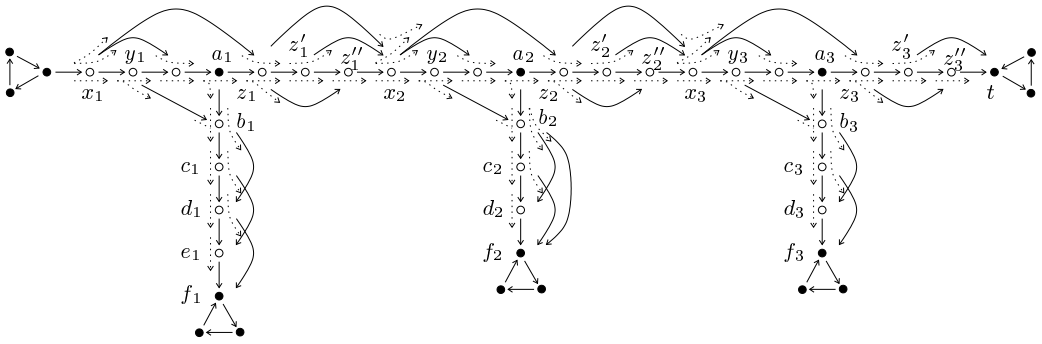

Figure 3.11: An example of a variable component as used for the proof of Theorem 3.2(b). The picture shows an example with three arms, the first and the second one of which are different in comparison to the variable component used for the proof of Theorem 3.2(a).

that $\{y_i, b_i, c_i, d_i\}$ is still an all-or-none-set (see again Figure 3.11). If a variable component has only one arm, then do the same thing to this arm.

If K_{F}, K_{M}, and K_{L} are the numbers of first, middle, and last arms, counted over all variable components, then set $K = 6 \cdot |U| + 20 \cdot |C| + 9 \cdot K_{\mathrm{F}} + 8 \cdot K_{\mathrm{M}} + 7 \cdot K_{\mathrm{L}}$.

Now follow the line of argument in the proof of Theorem 3.2(a). But note that while in the proof of Theorem 3.2(a), for any given variable component, the same number of white vertices had to be placed in a solution V' to the OC-bundle recognition problem no matter whether the corresponding variable was true or false, in the proof of Theorem 3.2(b) now, for any given variable component, the same number of singletons are induced, no matter whether the corresponding variable is true or false: Besides the $6 \cdot |U| + 9 \cdot |C|$ singletons consisting of an edge that is incident to two black vertices, there are nine singletons for every first arm, eight singletons for every middle arm, and seven singletons for every last arm of a variable component. Additionally, every clause component yields eleven singletons.

3.3 The OHC-Bundle Recognition Problem

Theorem 3.3(a) *Given a directed graph $G = (V, A)$, a subset A'_{L} of its line graph edge set, and an integer K, it is NP-complete to decide whether there is a vertex set $V' \subseteq V$ with $|V'| \leq K$ that solves the OHC-bundle recognition problem.*

Proof: First redefine again all-or-none-sets in the appropriate way, this time for OHC-bundle recognition problems: From now on, an *all-or-none-set* is a vertex set V'' such that for every solution V' to the OHC-bundle recognition problem, either $V'' \subseteq V'$, or $V'' \cap V' = \emptyset$.

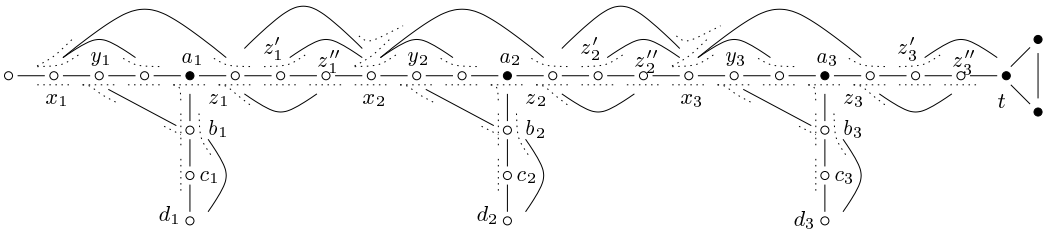

Figure 3.12: An example of a variable component as used for the proof of Theorem 3.3(a). If a variable appears ℓ times in clauses, then the corresponding variable component has ℓ *arms* and consists of $3+\ell$ black vertices, $1+9\ell$ white vertices, $6+34\ell$ edges, and $40\ell-2$ line graph edges. All edges and all line graph edges exist in both directions and are shown undirected. The picture shows an example with three arms. Every variable component is vertex and edge disjoint from every other variable component.

The OHC-bundle recognition problem is again in NP. As in the proofs in the previous section, we use a reduction from the following variation of One-In-Three 3Sat: Given a set U of Boolean variables and a set C of clauses over U where each clause contains exactly three positive literals, is there a truth assignment for the variables so that there is exactly one true literal in each clause ?

Given U and C, we now construct an instance $(G=(V,A), A'_{\mathrm{L}}, K)$ similarly to the one used in the proof of Theorem 3.2(a): For each variable in U the positive literal of which appears ℓ times in clauses of C, there is a *variable component* with ℓ *arms*. And for each clause c in C there is a *clause component* connecting three arms of variable components corresponding to the three literals making up c. The variable and clause components are defined by Figures 3.12 and 3.13, respectively. They differ from the variable and clause components used for the proof of Theorem 3.2(a). The connecting of arms of variable components through a clause component is now done as follows: Suppose u_i is the first literal appearing in a clause c. Then exactly one of the arms of the variable component corresponding to u_i, say arm number j, is connected to the clause component corresponding to c as follows: Identify vertices c_j and D_1 with each other, and identify vertices d_j and C_1 with each other, and then eliminate the two multiple edges that result, leaving again a simple graph. Similarly, proceed for the second and third literal of c. Note that G is a directed graph, in which every edge exists in both directions, and the line graph edges in A'_{L} also exist in both directions. Set $K = 3\cdot|U| + 20\cdot|C|$.

We will make use of the following lemmas:

Lemma 3.3.1 *In this instance of the OHC-bundle recognition problem, there are* $3|U|+4|C|$ *vertices (the black vertices of the variable components and of the*

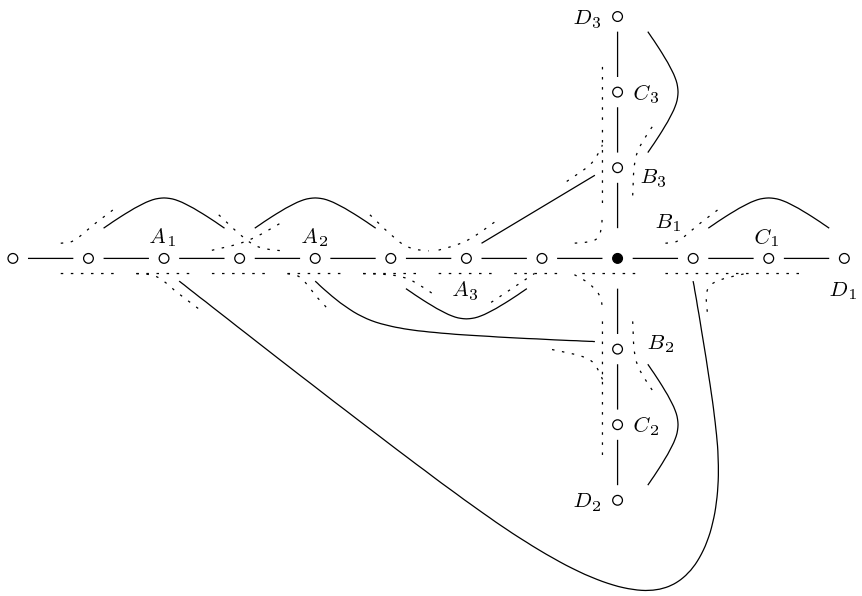

Figure 3.13: For each clause, there is a clause component connecting three arms of the variable components corresponding to the literals making up the clause. A clause component contains one black vertex, 17 white vertices, 52 edges, and 62 line graph edges. Every clause component is vertex and edge disjoint from every other clause component.

clause components) that every solution to the OHC-bundle recognition problem must contain.

The proof for the $3|U|+3|C|$ black vertices of variable components is analogous to the proof of Lemma 3.2.2. The $|C|$ black vertices of the clause components must be in every solution to the OHC-bundle recognition problem. For if a black vertex o of a clause component is not contained in a solution, there would be two edge sets in the induced edge partition cotaining edges incident to o, and these edge sets would not contain a Hamilton path.

Lemma 3.3.2 *If a variable component has ℓ arms, then the vertex set $\{x_1, z_1, z'_1, z''_1, \ldots, x_\ell, z_\ell, z'_\ell, z''_\ell\}$ forms an all-or-none-set.*

The proof is analogous to the proof of Lemma 3.2.3.

Lemma 3.3.3 *If arm number j of a variable component is connected to the vertices D_k and C_k of a clause component, then $\{y_j, b_j, c_j = D_k, d_j = C_k, B_k, A_k\}$ forms an all-or-none-set.*

To see this, first observe that $\{y_j, b_j\}$ is an all-or-none-set as in Lemma 3.2.4. Similarly, $\{A_k, B_k\}$ is also an all-or-none-set. Finally, $\{b_j, c_j = D_k, d_j = C_k, B_k\}$ forms an all-or-none-set as well, since $b_j \in V'$ implies $d_j \in V'$, $d_j = C_k$, $C_k \in V'$ implies $B_k \in V'$, $B_k \in V'$ implies $D_k \in V'$, $D_k = c_j$, and $c_j \in V'$ implies $b_j \in V'$.

To show now that there is a truth assignment for the variables in U fulfilling the One-In-Three 3Sat instance if and only if there is a solution to the OHC-bundle recognition problem with at most K vertices, first assume there is a fulfilling truth assignment. Construct a solution V' to the OHC-bundle recognition problem as follows. Let every one of the $3|U| + 4|C|$ black vertices be in V'. For each variable that is set to true, let the all-or-none-set containing x_1 of the corresponding variable component be in V'. For every variable that is set to false, let the all-or-none-sets of the arms of the corresponding variable component be in V'. In either case, this places $12|C|$ white vertices of variable components in V'. And since exactly one literal is true for every clause, it additionally places exactly four white vertices of clause components in V', namely for a given clause the vertices A_i, B_i, A_j, B_j if $i, j \in \{1, 2, 3\}$, $i \neq j$, are the two false literals of a clause. This set V' contains exactly K vertices, and it is a solution to the OHC-bundle recognition problem. In particular, every clause component contains edges of exactly two nontrivial bundles in the induced edge partition as illustrated in Figure 3.14, and these two bundles are opposite of each other.

Conversely, assume now that we are given a vertex set V' with at most K vertices that solves the OHC-bundle recognition problem. We have the following lemmas:

Lemma 3.3.4 *If the all-or-none-set containing x_1 of a variable component is not in V', then every all-or-none-set corresponding to an arm of the variable component must be in V'. Thus, there are at least $12 \cdot |C|$ white vertices of variable components in V'.*

The proof is identical to the proof of Lemma 3.2.6.

Lemma 3.3.5 *For every clause component, every solution to the OHC-bundle recognition problem must contain at least the vertices A_i, A_j, B_i, B_j for two distinct indices i and j, where $i, j \in \{1, 2, 3\}$.*

To see this, consider the directed path P from A_1 to the black vertex in the clause component in Figure 3.13 consisting of six horizontally drawn edges. Assume V' is a solution to the OHC-bundle recognition problem. If $A_i \notin V'$ for some $i \in \{1, 2, 3\}$, then all the unlabeled white vertices between A_i and the black vertex lying on P must not belong to V' either, because otherwise the edge set containing (A_i, B_i) in the induced edge partition would not contain a Hamilton path. Therefore, if additionally $A_j \notin V'$ for some $j \in \{1, 2, 3\}$, $i \neq j$, then (A_i, B_i) and (A_j, B_j) belong to the same edge set in the induced edge partition,

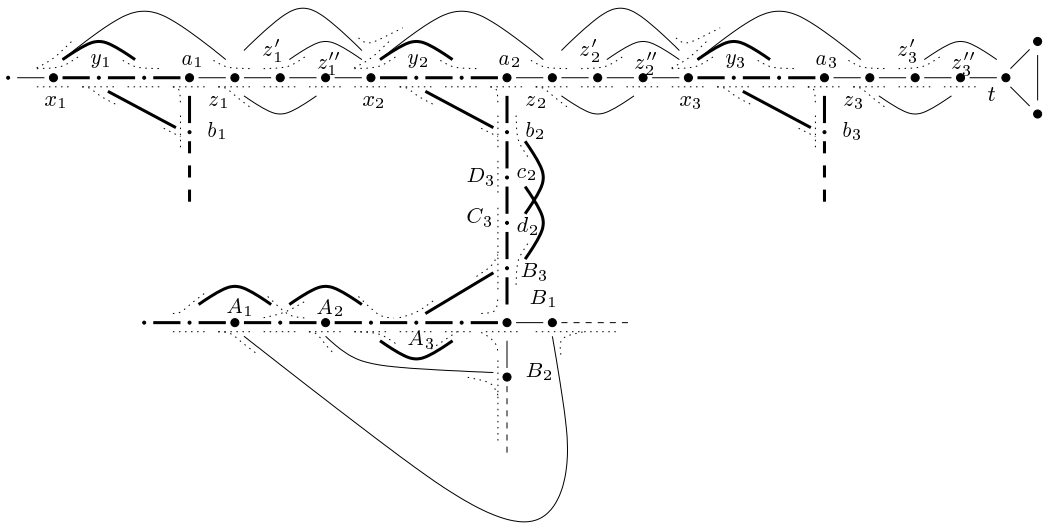

Figure 3.14: A variable component with three arms and the clause component connected to its second arm for the proof of Theorem 3.3(a). If a fulfilling truth assignment for the given One-In-Three 3Sat instance sets the variable corresponding to the variable component to true, then the solution V' to the OHC-bundle recognition problem constructed from the truth assignment (shown in big black dots) induces two nontrivial bundles in the clause component that are opposite of each other and that extend into the arm of the variable component the clause component is connected to. The two other arms of the variable component are also involved in nontrivial bundles with other clause components. The nontrivial bundles are shown in thick lines.

but this edge set does not contain a Hamilton path. So V' must contain at least the vertices A_i, A_j for two distinct indices i and j, where $i, j \in \{1, 2, 3\}$. Because of Lemma 3.3.3, B_i and B_j must then also belong to V'.

The minimum number of vertices that V' must contain according to Lemmas 3.3.1, 3.3.4, and 3.3.5 is $3 \cdot |U| + 4 \cdot |C| + 12 \cdot |C| + 4 \cdot |C| = K$. Therefore, if for a variable component its all-or-none-set containing x_1 belongs to V', then no all-or-none-sets corresponding to arms of the variable component may be in V', and we have a truth assignment for the variables by setting a variable to true if and only if the all-or-none-set containing x_1 belongs to V'. This truth assignment fulfills the One-In-Three 3Sat instance because due to the limit of K vertices for V', no clause component can have all three of its sets $\{A_i, B_i\}$ belong to V'. This concludes the proof of Theorem 3.3(a).

As with Theorems 3.1(a) and 3.1(b), and with Theorems 3.2(a) and 3.2(b), we can again obtain a result corresponding to Theorem 3.3(a) where the optimization criterion is the number of induced singletons rather than the number of vertices in the vertex set V':

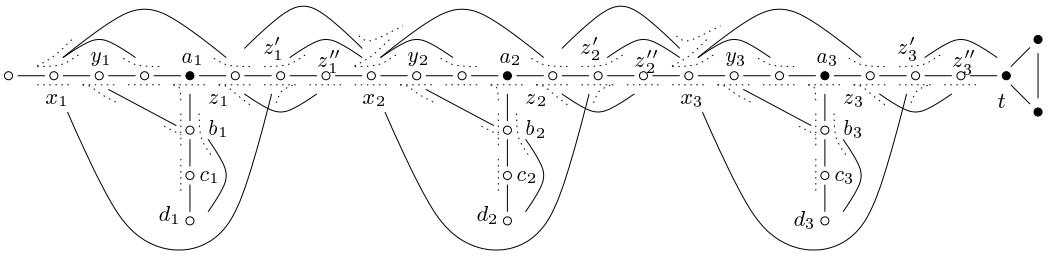

Figure 3.15: An example of a variable component as used for the proof of Theorem 3.3(b). The picture shows an example with three arms.

Theorem 3.3(b) *Given a directed graph $G = (V, A)$, a subset A'_{L} of its line graph edge set, and an integer K, it is NP-complete to decide whether there is a vertex set $V' \subseteq V$ that induces at most K singletons, and that solves the OHC-bundle recognition problem.*

Proof: The proof uses the same line of argument as the proof of Theorem 3.3(a) but with a slightly altered construction of the instance of the OHC-bundle recognition problem:

For every variable component, for each arm i, add the two directed edges (x_i, z'_i) and (z'_i, x_i), as well as the two line graph edges $((x_i, z'_i), (z'_i, z''_i))$ and $((z''_i, z'_i), (z'_i, x_i))$ as shown in Figure 3.15. Set $K = 6 \cdot |U| + 54 \cdot |C|$ and observe that Lemmas 3.3.1 through 3.3.5 still hold. Again, first assume that there is a fulfilling truth assignment for the One-In-Three 3Sat instance. Construct a solution to the OHC-bundle recognition problem exactly as in the proof of Theorem 3.3(a) and observe that it induces exactly K singletons. Conversely, assume there is a solution V' to the OHC-bundle recognition problem inducing at most K singletons. If for each variable component, either its all-or-none-set containing x_1 or the all-or-none-sets corresponding to its arms are part of V', and if for each clause component, four white vertices A_i, A_j, B_i, B_j according to Lemma 3.3.5 are in V', then V' induces an edge partition with at least K singletons. Therefore, for no variable component, both its all-or-none-set containing x_1 *and* any of the all-or-none-sets corresponding to its arms may be part of V', and for no clause component $\{A_i, B_i\} \subset V'$ for all $i \in \{1, 2, 3\}$, for then the number of induced singletons would be greater than K. We can therefore construct a fulfilling truth assignment for the One-In-Three 3Sat instance again exactly as in the proof of Theorem 3.3(a).

Chapter 4

Heuristic Scheme

In Chapter 3, we have seen that the bundle recognition problem (in fact, several variations of it) is intractable. Therefore, we will now turn our attention to a heuristic to solve it. Chapter 5 will then provide a computational study based on this heuristic.

4.1 Overview

The main idea for the heuristic scheme described in this chapter is that, given a vertex set V' (that may or may not solve the bundle recognition problem), we may try to locally guess whether an individual vertex v that is currently in V' is actually not needed in V' by examining the edge sets of the current edge partition that are incident to v. Taking v out of V' will usually result in uniting formerly separate edge sets to form one new, larger edge set. The goal is, of course, for these edge sets to be bundles.

Before growing bundles, we construct a vertex set V', that, even if it does not solve the bundle recognition problem, is a reasonable starting set for the bundle growing process.

Besides eliminating vertices from V' to grow bundles, we also add vertices to V' in order to repair violations of the bundle conditions in Definition 2.5, i.e. we augment V'. Recall that $V' = V$ is a feasible, albeit practically useless, solution to the bundle recognition problem, so augmenting V' until the bundle conditions are fulfilled will always yield a feasible solution. This guarantees that the result of the heuristic does indeed solve the bundle recognition problem.

Together, these ideas lead to the heuristic scheme formulated in Figure 4.1 and explained in detail in the remainder of this chapter.

Various ways of choosing an initial vertex set V' are discussed in Section 4.2. As a concrete example, consider the European time tables in the Summer of 1997. Their induced time table graph contains 28 280 vertices and 82 594 (directed) edges. Choosing vertices whose vertex degree in the underlying undirected graph

Initial V': Guess an initial set $V' \subseteq V$ and determine the edge partition it induces.

First augment: For each induced edge set that is not a DAG or that is a DAG but does not contain a Hamilton path, add all end vertices of its edges to V'. Also, for each vertex not in V' that is common to two non-opposite edge sets, add that vertex to V'. Determine the edge partition induced by the augmented V'.

Repeat

reduce: If the edge sets incident to a vertex $v \in V'$ can be united to form new, larger edge sets that will likely form bundles, then delete v from V' and update the edge partition accordingly.

unite-singleton: If there is a singleton $\{(s,t)\}$ and exactly one other bundle whose Hamilton path begins with s and ends with t, then unite the singleton with this bundle.

until no more changes in V' occur. Note that so far it is not guaranteed that the resulting vertex set V' is a solution to the bundle recognition problem.

Final augment: Perform the augment step again, if needed repeatedly, so that the resulting vertex set V' is a solution to the bundle recognition problem.

Final unite-singleton: Perform unite-singleton again one more time.

Figure 4.1: The heuristic scheme for the bundle recognition problem

is larger than that of at least 60 percent of their neighbors yields an initial vertex set with cardinality 4 679. The part of this time table graph showing the South West corner of Germany and the adjoining parts of Switzerland (with Basel, Schaffhausen, and Konstanz being at the border between the two countries) is displayed in Figure 4.4, with vertices in V' in large dots and all edges shown undirected. We will use this region as an example throughout the whole chapter.

Of course this initial vertex set does not solve the bundle recognition problem. Some edge sets in the induced partition do form DAGs with Hamilton paths, and the Hamilton paths and transitive edges of these edge sets are indicated in black and grey, respectively, in Figure 4.4. But many edge sets violate the bundle conditions of Definition 2.5: In Figure 4.4, no cyclic edge sets occur in the induced initial partition, but edge sets that do not contain a Hamilton path are indicated in red. Also, whenever edge sets that are not opposite have a vertex outside V' in common, this vertex is indicated in green. A legend for the remaining figures

in this section is also given in Figure 4.2.

Figure 4.5 then shows the induced partition after the *first augment* step, where all green vertices and all vertices of red edge sets in Figure 4.4 have been added to V' and the resulting induced edge partition has been determined. Violations of the bundle conditions are still possible at this stage. In the given example, there are edge sets between Singen, Tuttlingen and Donaueschingen without Hamilton path. We will examine this situation in more detail at the end of this section.

The result of the bundle growing process that follows the *first augment* step in Figure 4.5 is shown in Figure 4.6. Observe that for example a bundle stretching from Tuttlingen to Sigmaringen was grown, and that six vertices now belonging to this bundle have been eliminated from V' in the process. The bundle growing process is described in Section 4.3.

Note that violations of the bundle conditions are still possible after the bundle growing process. But the *final augment* step always yields a solution V' to the bundle recognition problem, in our example depicted in Figure 4.7. After a solution V' to the bundle recognition problem has been found with the completion of the *final augment*, there are often singletons such as (Singen, Schaffhausen) in Figure 4.7 where there is a unique bundle with the end points of its Hamilton path equal to the end points of the singleton. In this case, we assume that the singleton is a transitive edge, and that the trains traveling along it actually travel along the Hamilton path of the bundle. Therefore, we perform a *final unite-singleton* after the *final augment* to take care of all such singletons with unique corresponding bundles. The *final unite-singleton* corresponds to step 3. of the edge partitioning procedure in Section 2.1. Figure 4.8 shows the results for the given example. For the European time table graph in the example, 4 126 edges became transitive edges through the *final unite-singleton*, leaving 16 066 singletons in the final partition.

Altogether, the heuristic has determined 10 146 nontrivial bundles containing 52 973 Hamilton path edges and 13 555 transitive edges for the European time table graph with the initial vertex set as chosen in this example.

Resulting from the original edge classification problem, as well as from observations with time table graphs, we introduced as an additional element the addition of artificial line graph edges: It occurs in time table graphs that a vertex v should according to intuition be an inner vertex of a bundle but remains in V' because all trains visiting v begin or end there (see Figure 4.19 on page 62 for an example). When such an absence of line graph edges is detected in a *reduce* step, artificial line graph edges are added to the instance of the bundle recognition problem so that in the induced edge partition, bundles incident to v are united to form larger bundles. Section 4.4 discusses in detail when the addition of artificial ine graph edges is helpful.

The addition of artificial line graph edges in the *reduce* step implies that the vertex set determined by the heuristic after the *final augment* step is strictly speaking not a solution to the bundle recognition problem given by a time table

graph and the subset of line graph edges induced by the trains making up the time tables. Also, applying *final unite-singleton* means that the edge partition at the end of the heuristic is not the partition induced by the set V' and by the (real and artificial) line graph edges. While the heuristic without *final unite-singleton* and without artificial line graph edges can serve to find solutions to instances of the bundle recognition problem, the actual goal in devising a heuristic is to solve the original edge classification problem. And it is this goal in mind that motivates the artificial line graph edges and the *final unite-singleton* in the heuristic scheme presented in this chapter.

Finally, we examine the situation between Singen, Tuttlingen and Donaueschingen in more detail in order to illustrate by this example how there may be edge sets violating the bundle conditions in Definition 2.5 even after *first augment*. Consider Figure 4.3 which shows again the initial partition between Singen, Tuttlingen and Donaueschingen, but this time including the line graph edges induced by the train time tables: Initially, the vertices Singen, Tuttlingen and Donaueschingen are in V'. With the line graph edges induced by the trains traveling between these cities, three nontrivial edge sets are induced: The first one consists of three edges that also form its Hamilton path (Singen, Mühlhausen, Welschingen, Engen). The second edge set consists of eight edges, three of which are transitive, and its Hamilton path is (Singen, Engen, Tuttlingen, Immendingen, Geisingen, Donaueschingen). The third one consists of eleven edges and has the Hamilton path (Donaueschingen, Geisingen, Immendingen, Tuttlingen, Engen, Welschingen, Mühlhausen, Singen). If additionally there were the line graph edge ((Welschingen, Engen), (Engen, Immendingen)), then the first and the second edge set listed above would be united to form one edge set in which the edge (Singen, Engen) would be transitive. But as it is, Engen is a vertex that is common to two edge sets (the first and the second one, for example) that are not opposite. So in the *first augment* step, Engen is added to V'. Figure 4.5 shows the resulting edge partition. Now however, the edge sets containing Immendingen do not contain a Hamilton path anymore, so in the *final augment* step, all their vertices are added to V' (see Figure 4.7).

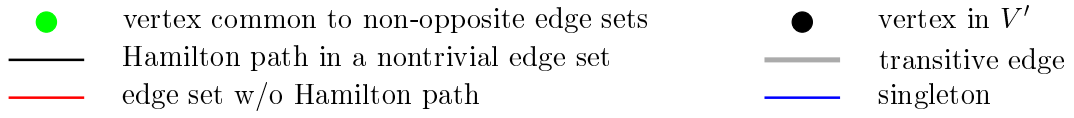

Figure 4.2: Legend for Figures 4.3 through 4.8

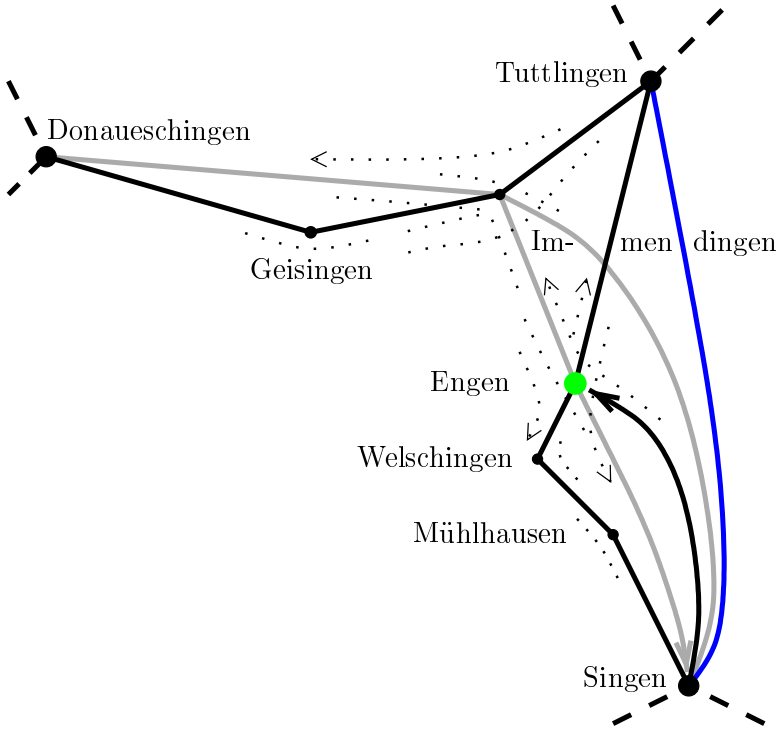

Figure 4.3: An initial partition: Detail around Immendingen. Initially, the vertices Singen, Tuttlingen and Donaueschingen are in V'. Edges (solid) and line graph edges (dotted) shown undirected exist in both directions.

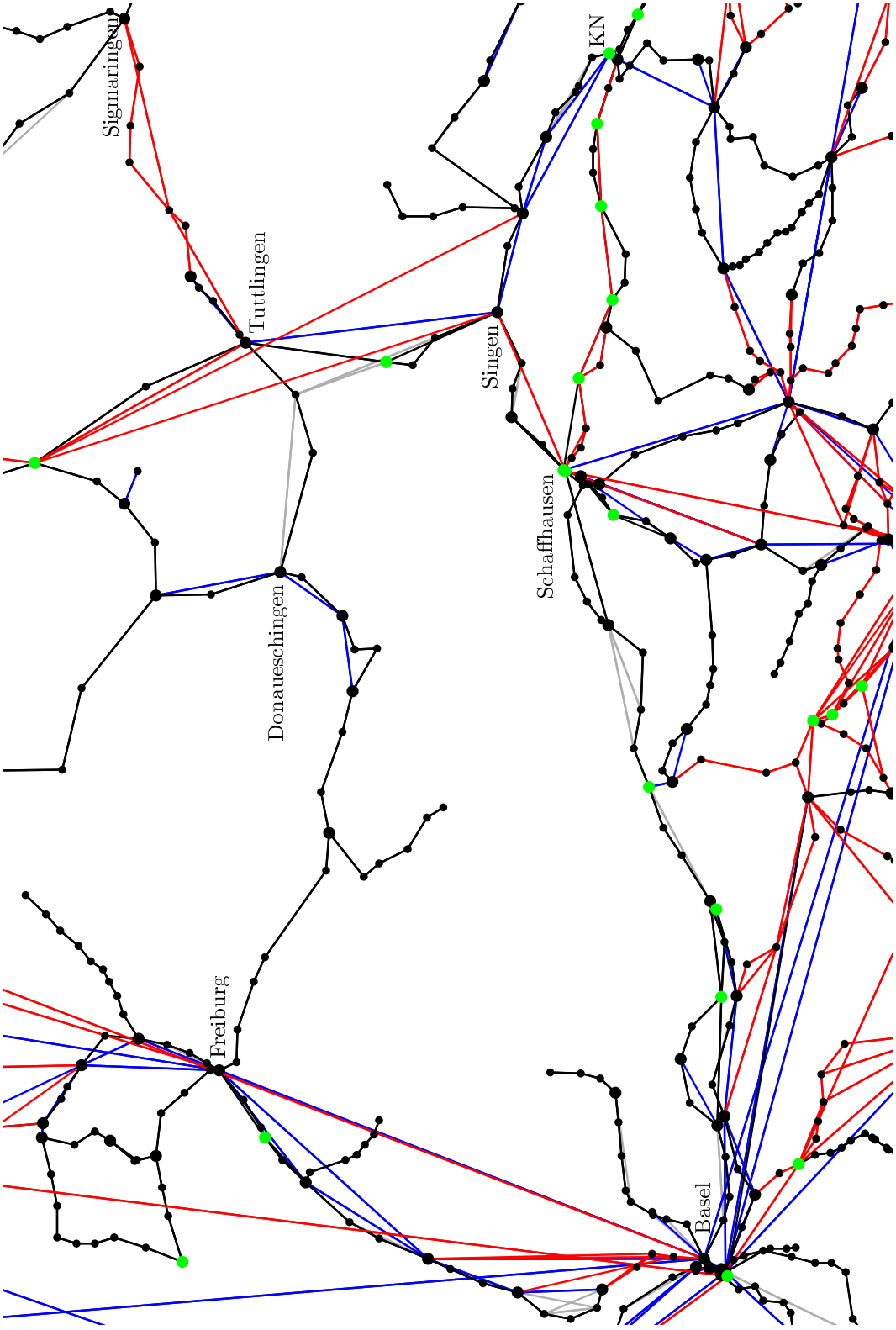

Figure 4.4: An initial partition. KN is Konstanz, Germany.

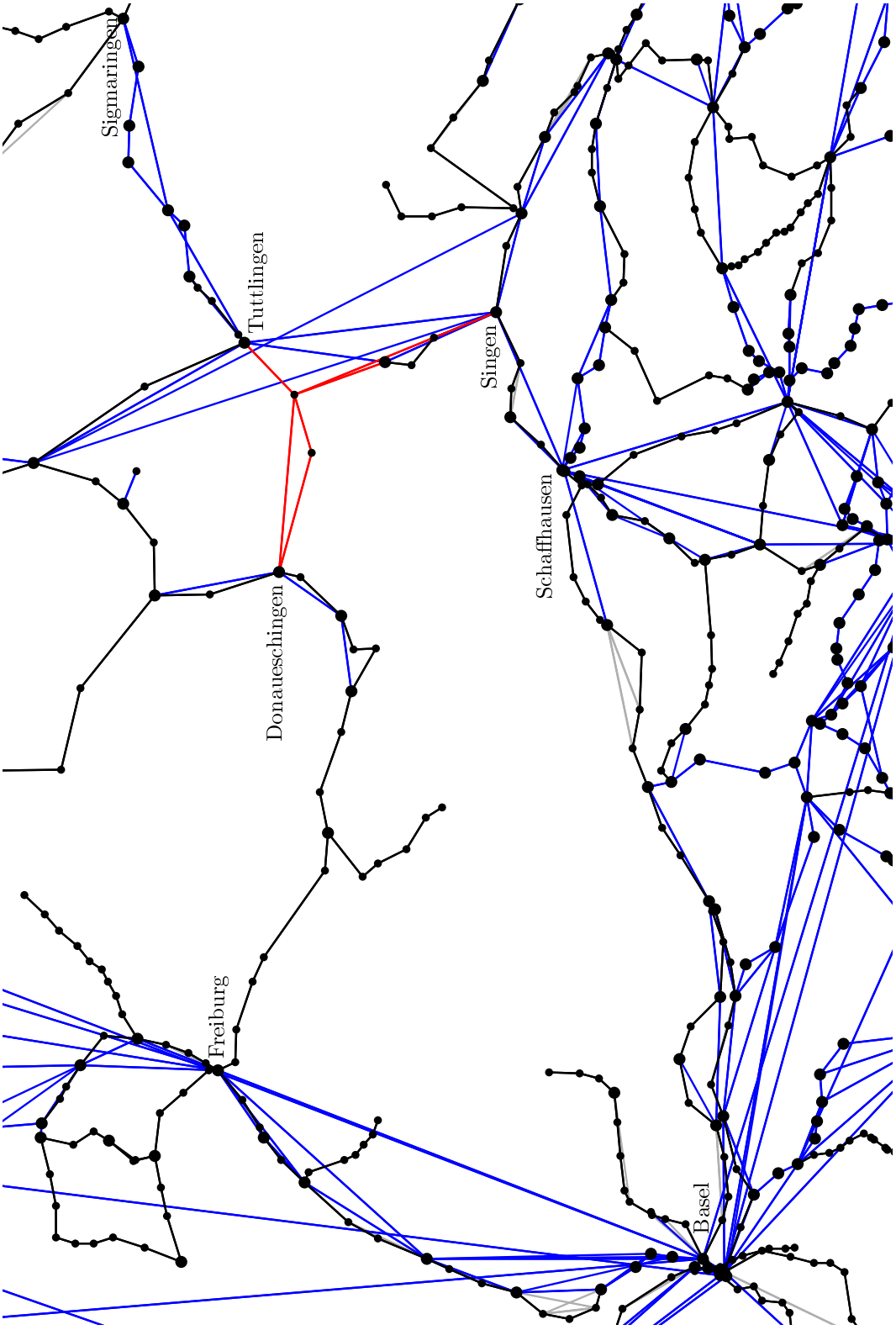

Figure 4.5: Partition after the *first augment* step

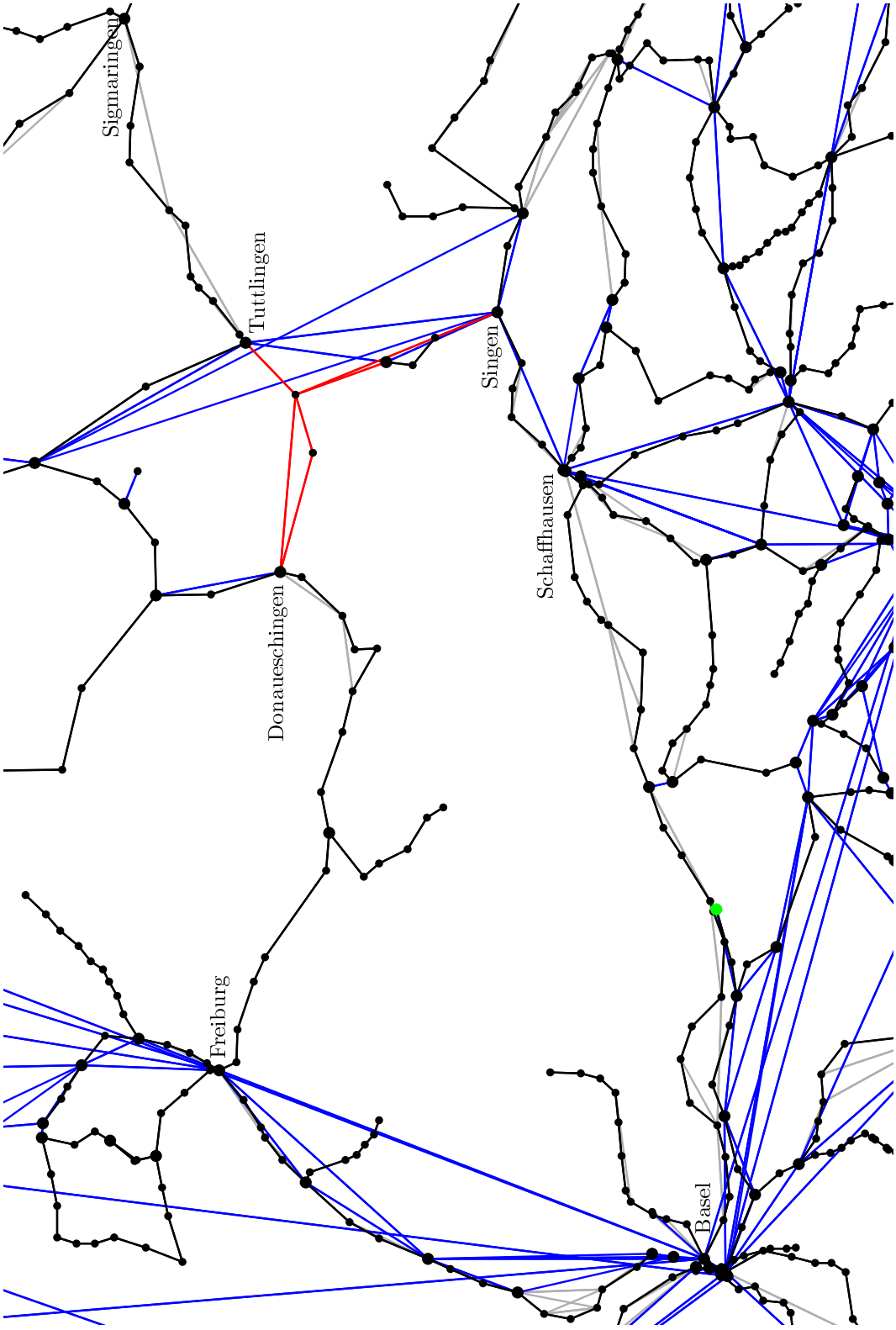

Figure 4.6: Partition before *final augment*

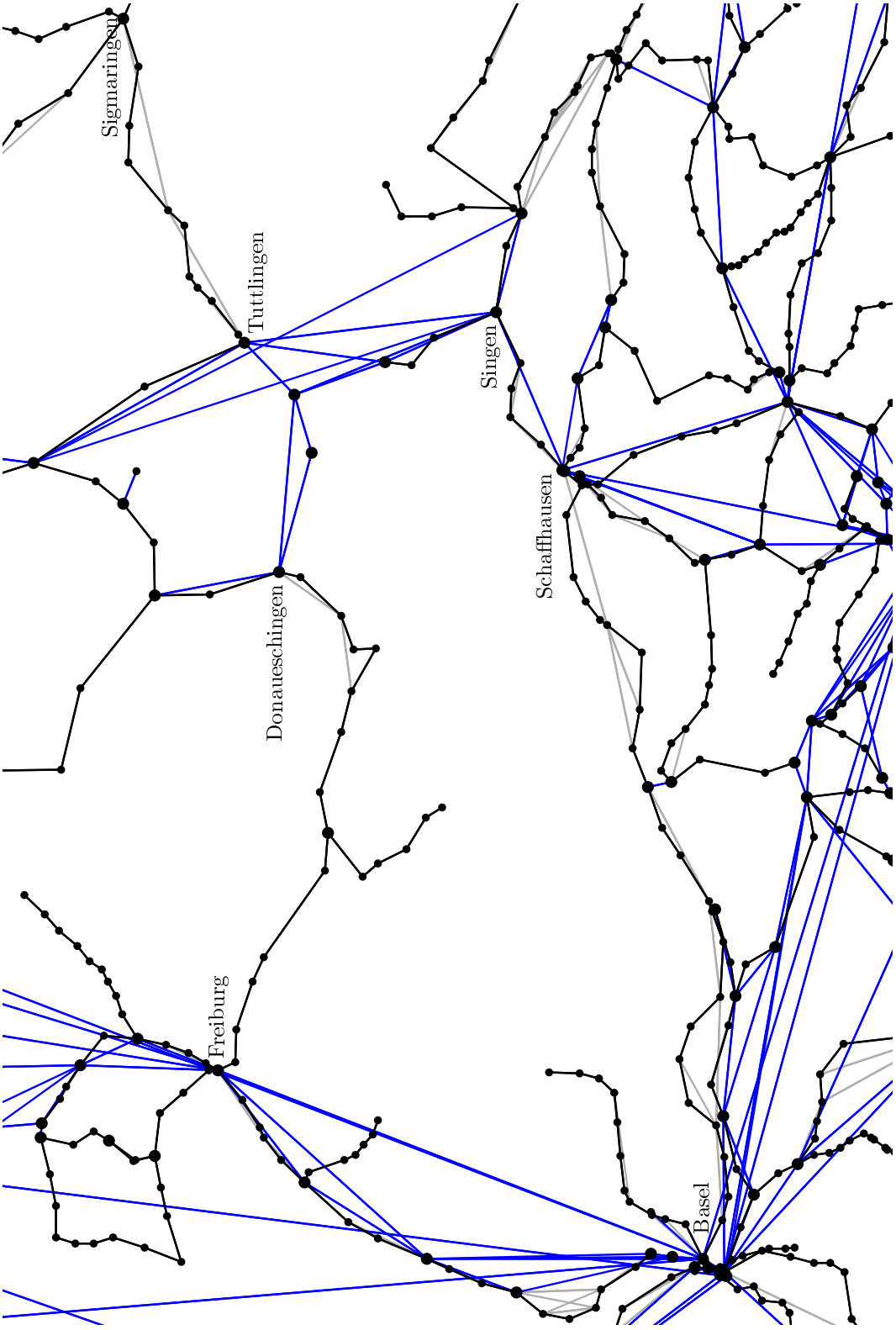

Figure 4.7: Partition after the *final augment* step

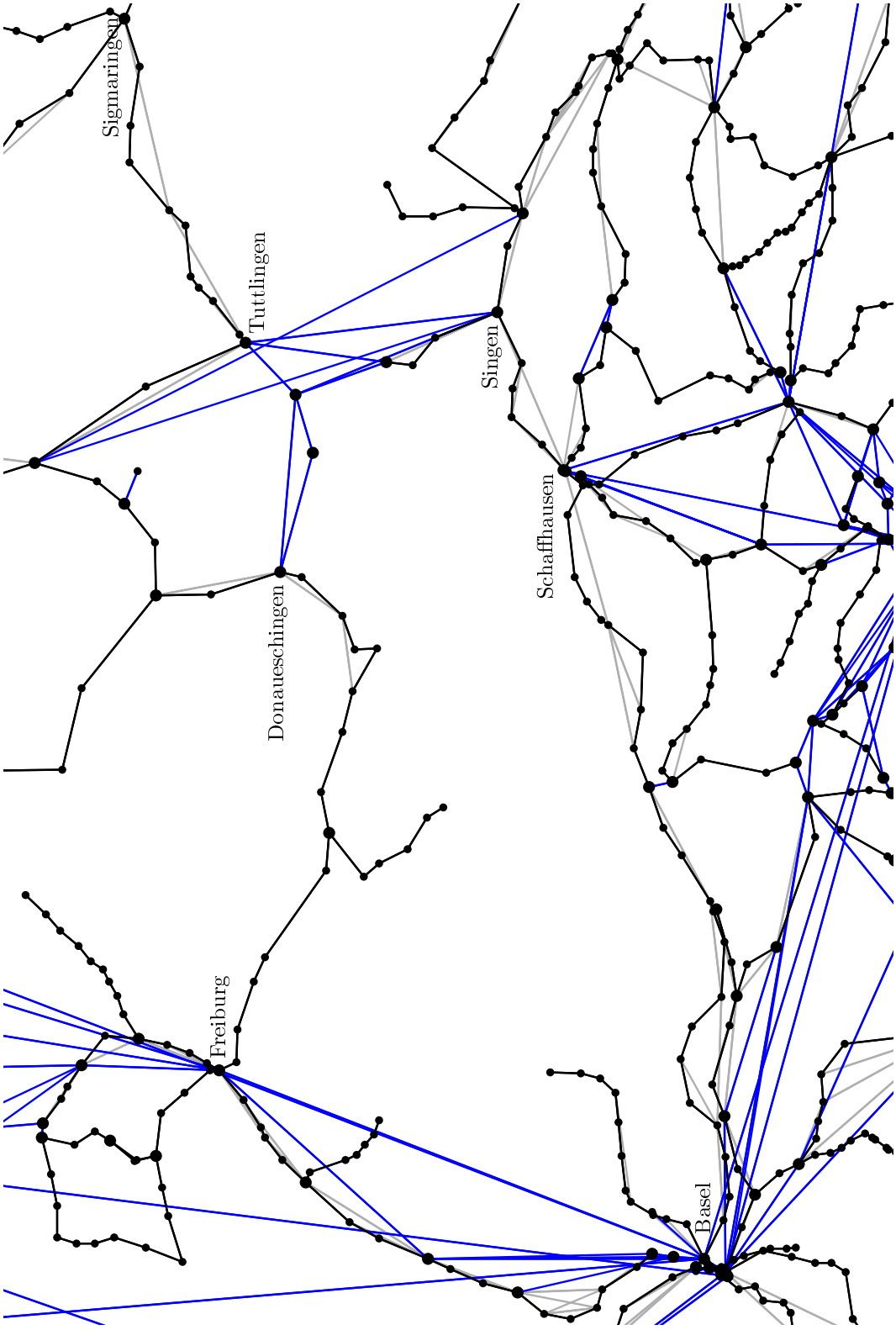

Figure 4.8: Final partition

4.2 Guessing an Initial Vertex Set

An initial vertex set is good if the heuristic, when starting with this vertex set and working on time table graphs induced by train time tables collected by the *TLC*, leaves few edges as singletons. We will present various choices for an initial vertex set in this section. Chapter 5 contains computational results for the heuristic started with these initial vertex sets and with combinations of these sets.

One possibility for an initial vertex set is to take the whole vertex set V and to rely entirely on the bundle growing process as described in Section 4.3. But Section 4.3 also illustrates that unfortunately, the bundle growing process does not always work well, and the computational results in Chapter 5 will confirm that V is not a good choice for an initial vertex set. So in this section we will present ideas how to guess an initial vertex set that (hopefully) works better with the heuristic. When suggesting initial vertex sets, we have in mind that branching points of the physical railroad network are intuitively useful for solving the bundle recognition problem on time table graphs.

4.2.1 Vertex Degree

The exploration of time table graphs suggests that vertices that are branching points of the physical railroad network (and thus candidates for being bundle end points) have a relatively high vertex degree. Therefore it is sensible to consider the vertex degree when guessing an initial vertex set of potential bundle end points for the heuristic. Clearly, the absolute vertex degree itself being high or low does not indicate potential bundle end points. Compare for example Titisee and Löffingen in Figure 4.10 on page 52: Their vertex degree is identical, but one is a bundle end point and the other is not. This is only one of many such examples. Rather, we consider the vertex degree of a vertex in comparison to the vertex degree of the vertex's neighbors. Intuitively, branching points of the physical railroad network will tend to have a vertex degree that is larger than that of (most of) their neighbors. We will use the vertex degree in the underlying undirected graph.

Since there are often edges like (Singen, Schaffhausen) connecting branching points, we cannot expect the branching vertices to have a vertex degree higher than that of *all* of their neighbors. In Section 4.1, vertices whose vertex degree in the undirected graph underlying the time table graph is larger than that of at least 60 percent of their neighbors were used for an initial vertex set. Figure 4.10 shows vertices whose degree is larger than that of at least 50 percent of their neighbors (shown in tiny black dots like Waldshut), 60 percent of their neighbors (small black dots like Löffingen), 70 percent of their neighbors (medium sized black dots like Döggingen) or even 80 percent of their neighbors (big black dots like Donaueschingen). As the following table shows, there are 6 975, 4 679, 2 724,

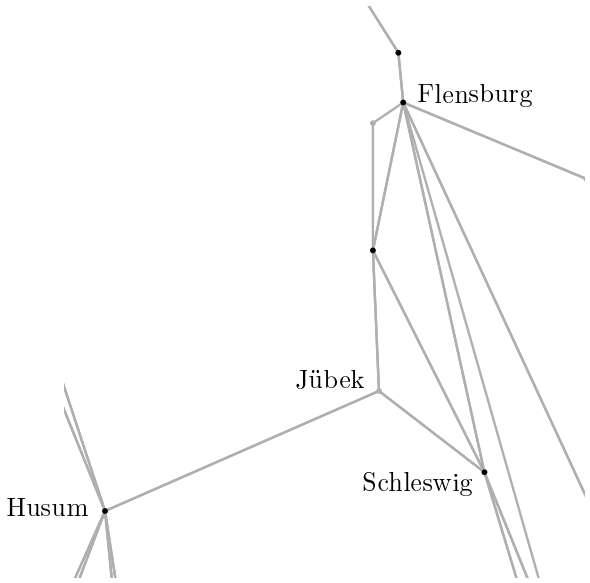

Figure 4.9: Vertices (in black) with (undirected) degree larger than that of at least one of their neighbors

and 1 820 such vertices, respectively, in the time table graph of the summer of 1997:

%	10	20	30	40	50	60	70	80	90	100
#	10 091	10 066	9 443	7 338	6 975	4 679	2 724	1 820	874	664

For each percentage x (top row), the number of vertices whose (undirected) vertex degree is higher than that of at least x percent of their neighbors is given (bottom row). The total number of vertices in the time table graph is 28 280.

None of the four vertex sets shown in Figure 4.10 corresponds exactly to the bundle end points we seek. Each contains vertices that *are not* bundle end points, and at the same time misses vertices that *are* bundle end points: For example, the branching point Immendingen does not even have vertex degree larger than that of 50 percent of its neighbors, while Erzingen has vertex degree larger than that of at least 80 percent of its neighbors but is not an intuitive bundle end point. It is even possible that a branching point has a vertex degree that is smaller than that of all its neighbors, and Figure 4.9 contains Jübek in Northern Germany as an example. Section 4.1 showed how a vertex set with high vertex degrees in comparison to the vertex's neighbors can nonetheless be a useful initial vertex set for the heuristic, and this is confirmed in the computational results of Chapter 5.

4.2.2 Branches in Minimum Spanning Forests

The intuition that the edges of the physical railroad network tend to be short in comparison with transitive edges leads to the idea that a minimum spanning forest (MSF for short) would tend to be made up of real edges. In particular then, the branching vertices of an MSF (i.e. the vertices with vertex degree at least three within the MSF) would also be branching vertices of the physical railroad network. We will use the branching vertices within minimum spanning forests of the underlying undirected graph as initial vertex sets for the heuristic.

Two natural measures for the length of an (undirected) edge come to mind: The Euclidean length of the edge, and the minimum travel time of trains along the edge. For the underlying undirected graph of the time table graph of the summer of 1997, we determine a minimum spanning forest based on Euclidean edge lengths and call this particular undirected forest MSF_e. Out of its 28 280 vertices, 2 395 are branching vertices (i.e. vertices that have degree at least three within the forest).

Similarly we determine a minimum spanning forest based on travel times and call it MSF_t. It has 2 358 branching vertices, and the union of MSF_e and MSF_t has 3 105 vertices that have degree at least three (within the union). Figure 4.11 shows the two forests and their union for the example used throughout this chapter.

Notice that in Figure 4.11, vertices such as Donaueschingen and Singen that are branching points of the physical railroad network would not be included in the initial vertex set if we took only the branching vertices within MSF_e. Similarly, Immendingen and Waldshut would not be included if we took only the branching vertices within MSF_t. And Rottweil does not even have degree three or more within the union of the two minimum spanning forests, even though it is a branching vertex of the physical railroad network. So many intuitive bundle end points are missing from initial vertex sets based on minimum spanning forests, and Chapter 5 confirms that these initial vertex sets do not produce very good results for the heuristic, even though they are better than the trivial initial vertex set V. However, Figure 4.11 suggests that the branching vertices of MSF_e and MSF_t and even the vertices with degree three or more within their union tend to also be branching points of the physical railroad network. In Chapter 5, we will study combinations of initial vertex sets introduced here, and for these combinations, the initial vertex sets based on minimum spanning forests will again be useful.

4.2.3 Angles

The previous section used a geometric argument (namely that real edges tend to have small Euclidean edge lengths in comparison with transitive edges) as a basis for identifying branching points of the physical railroad network. Another geometric indication for such branching points may be the angles between the

edges incident to a vertex. So for each vertex v, determine the angles between consecutive edges incident to v and call the three largest angles α_1, α_2, and α_3, respectively.

For a non-branching vertex along a line of railroad tracks like Beuron in Figure 4.12, its incident edges will tend to point in two main directions, so α_1 and α_2 will be large. Conversely, the edges incident to a branching vertex like Freiburg will tend to point in various directions, so that α_1 and α_2 are only moderate or even small. Following this observation, we identify vertices as branching vertices (and therefore as potential bundle end points) for which $\alpha_1+\alpha_2$ is "not too large". Figure 4.12 shows vertices (in black), for which $\alpha_1 + \alpha_2 \leq 315°$ (tiny dots), 300° (small), 285° (medium), or even 270° (large dots). In the time table graph for the summer of 1997, there are 4 073, 3 227, 2 539, and 1 932 such vertices, respectively.

Similarly, a vertex will tend to be a branching vertex if α_3 is "not too small". Figure 4.13 shows vertices for which $\alpha_3 \geq 20°$ (tiny dots), 30° (small), 40° (medium), or even 50° (large dots). In the time table graph for the summer of 1997, there are 5 661, 4 197, 3 076, and 2 200 such vertices, respectively.

Of course these criteria can again err in two ways: Not only will Titisee not be considered a branching vertex in Figure 4.12, but for Emmendingen, which is not a branching vertex, we even have $\alpha_1 + \alpha_2 \leq 270°$. And for Sissach, which is not a branching vertex either, we have $\alpha_3 \geq 50°$ in Figure 4.13. To illustrate that we cannot expect to formulate an angle condition that will recognize precisely the branching points of the physical railroad network, notice that the angle situation at Titisee is very similar to the one at Löffingen, and Titisee is a branching vertex while Löffingen is not. As with the initial vertex sets based on minimum spanning forests, the initial vertex sets based on angles will turn out to be useful for the heuristic in combinations with other sets, rather than by themselves.

4.2.4 Terminal Stations

Another idea for finding potential bundle end points is to use the terminal vertices of trains, i.e. vertices where trains begin or end, as an initial vertex set for the heuristic, assuming that usually, trains do not start or end in the middle of a line of railroad tracks, but rather at its endpoints.

In the time table graph for the summer of 1997, trains begin or end at 5 942 out of the 28 280 vertices, and for 5 368 of them, or 90 percent, there is at least one train that starts *and* at least one train that ends at this train station. Figure 4.14 shows vertices where trains begin or end. In the example shown in Figure 4.14, intuitive bundle end points are indeed mostly found as terminal vertices (with a few exceptions such as Sulgen), but many vertices such as Hinterzarten where trains begin or end are not intuitive bundle end points. Chapter 5 will show that the terminal vertices form a useful initial vertex set for the heuristic, similar to the vertices whose undirected degree is larger than that of at least 60 % of their neighbors.

4.2.5 Importance Number

In connection with another project, the *TLC* assigns a natural number to each train station that reflects in some way its "importance". We will investigate how useful this "importance number" is for finding potential bundle end points as an initial vertex set for the heuristic, hoping that the "important" vertices roughly correspond to end points of railway lines and thus to the bundle end points that we seek.

In the time table graph for the summer of 1997, more than half of the 28 280 vertices have importance number 1, and six percent of them have the highest importance number that is used. Figure 4.15 illustrates the distribution of importance numbers for our example of a time table graph. It shows black vertices in four different sizes: The tiny black vertices have an importance number that is larger than that of 60 percent of all the vertices. The small black vertices (such as Hinterzarten) have an importance number that is even larger than the importance number of 70 percent of all vertices, and similarly for the medium sized black vertices (such as Titisee) and 80 percent of all vertices. Finally, the large black vertices (such as Neustadt) have an importance number that is larger than that of 90 percent of all vertices. As an initial vertex set for the heuristic, the vertices with importance number larger than that of 80 % of all vertices in Figure 4.15 seem intuitively most useful among the four sets shown, and the computational results in Chapter 5 will show that this set is indeed a very good initial vertex set for the heuristic. But notice that (again) the bundle end point Sulgen would not belong to this set, while the vertex set also contains vertices such as Albbruck and Neustadt that are not intuitive bundle end points.

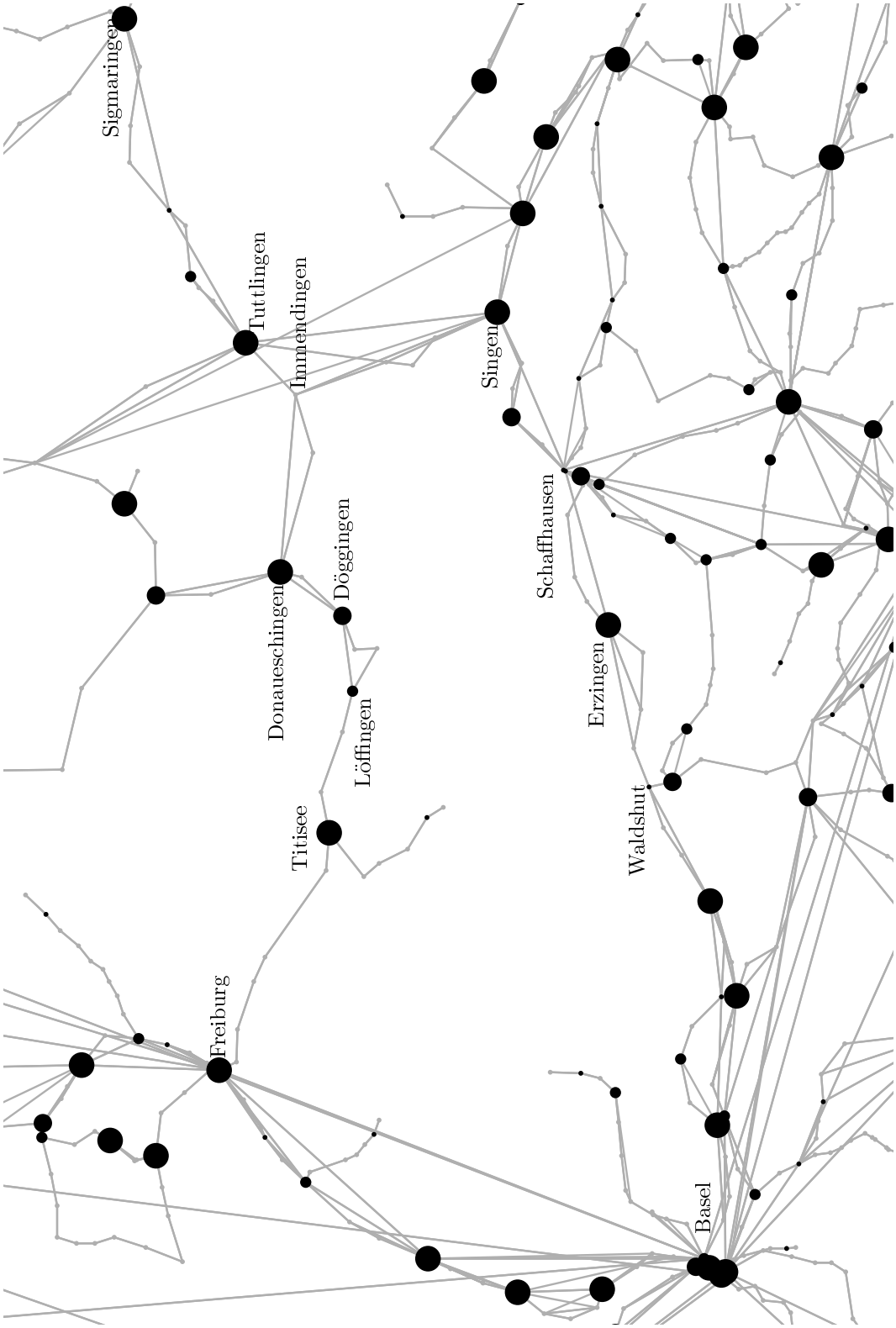

Figure 4.10: Vertices (in black) with (undirected) vertex degree larger than that of at least 50 (tiny), 60 (small), 70 (medium), or even 80 % (large) of their neighbors

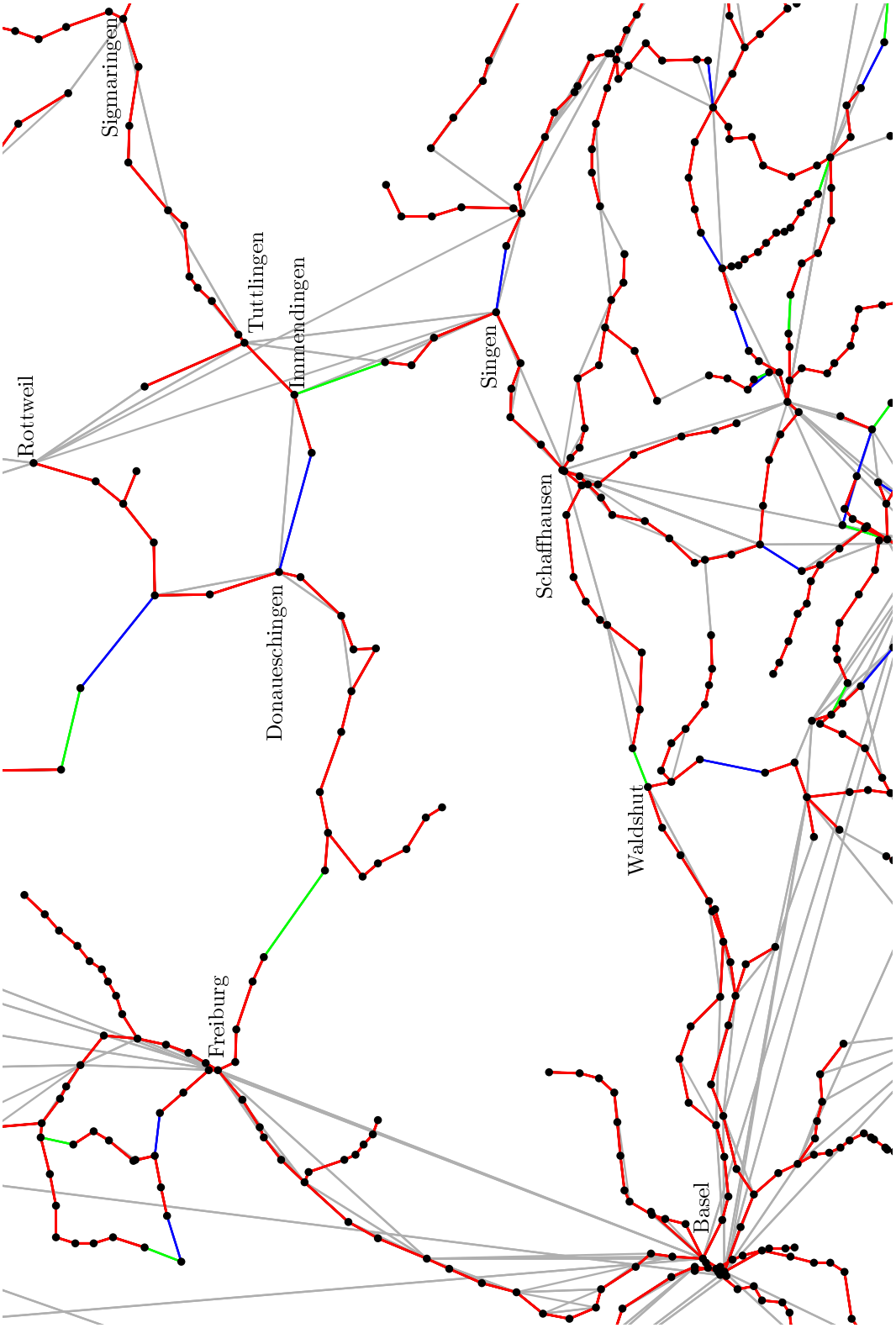

Figure 4.11: Edges of MSF_e, a minimum spanning forest based on Euclidean lengths, (green), of MSF_t, a minimum spanning forest based on travel times (blue), and edges belonging to both minimum spanning forests (red)

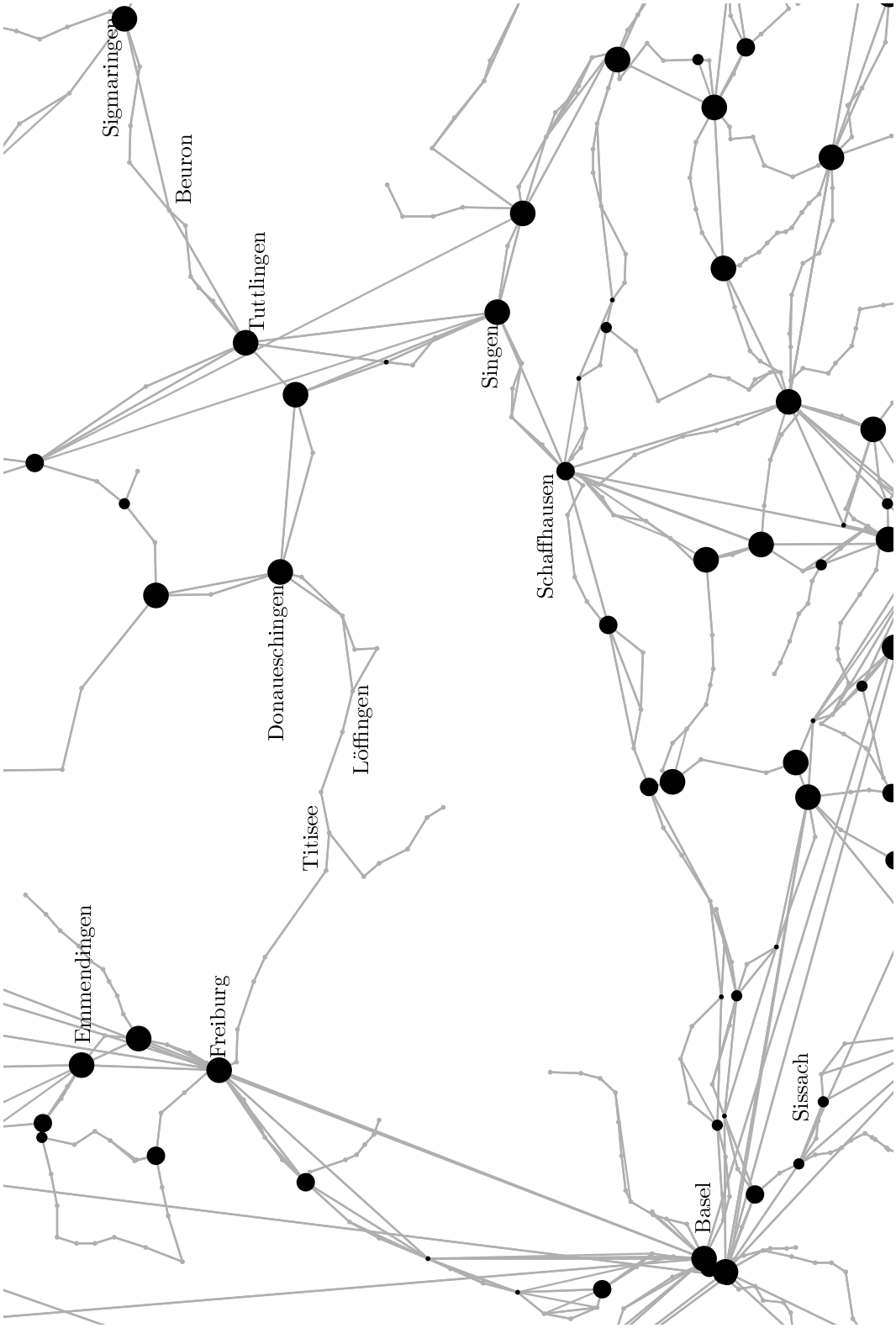

Figure 4.12: Vertices (in black) with $\alpha_1 + \alpha_2 \leq 315°$ (tiny), 300° (small), 285° (medium), or even 270° (large)

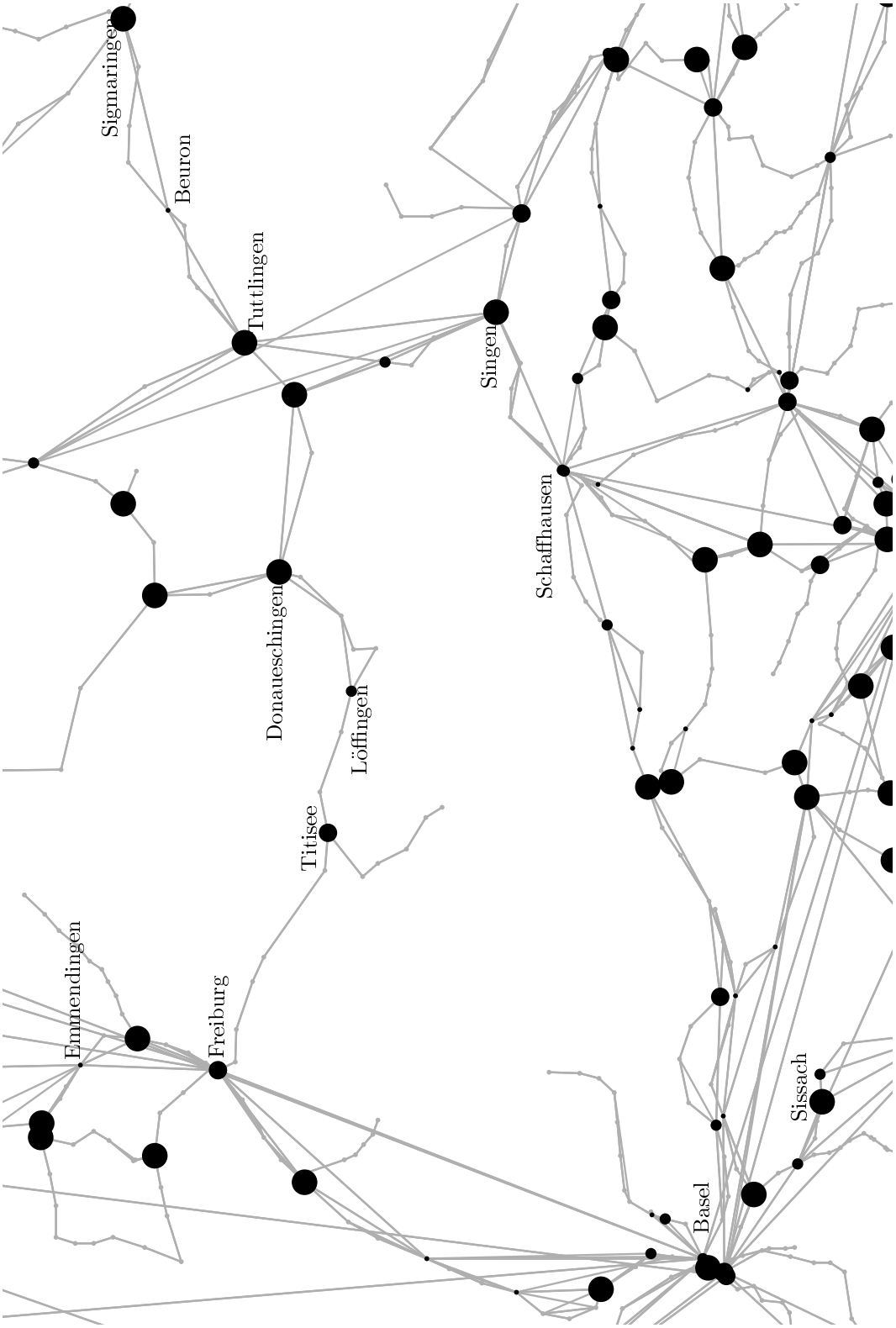

Figure 4.13: Vertices (in black) with $\alpha_3 \geq 20°$ (tiny), 30° (small), 40° (medium), or even 50° (large)

Figure 4.14: Vertices (in black) where trains begin or end

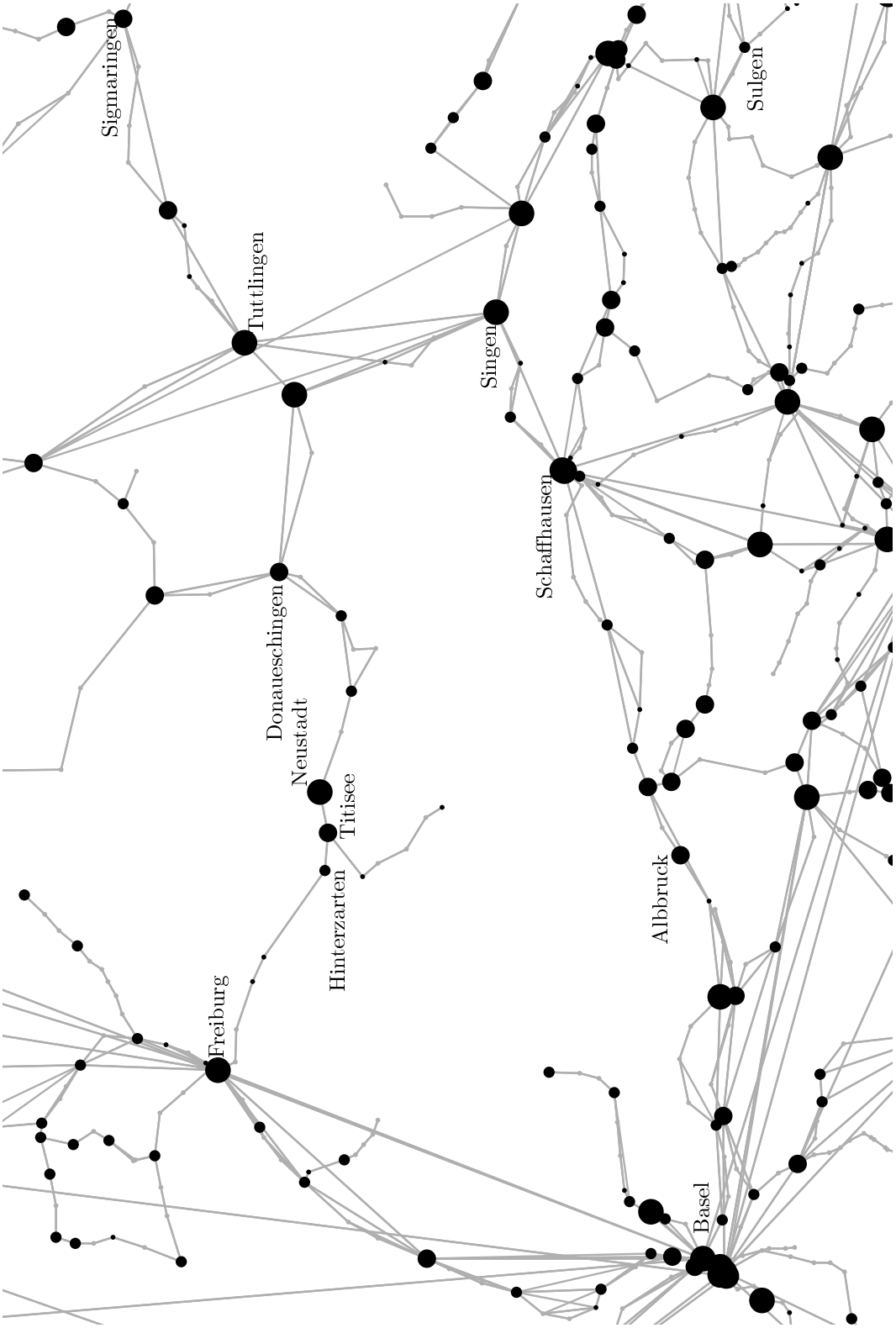

Figure 4.15: Vertices (in black) with importance number higher than that of 60 (tiny), 70 (small), 80 (medium), or even 90 % (large) of all vertices

4.3 Growing Bundles

Given a vertex subset V' for a time table graph and its induced edge partition (where V' may or may not solve the bundle recognition problem), the goal of the bundle growing process is to find a new vertex set $V'' \subset V'$ so that in the edge partition induced by V'', some formerly separate edge sets are united to form new larger edge sets, and so that these edge sets are bundles. The approach is to locally guess for each individual vertex v whether or not v is actually needed in V'. This is done by examining the edge sets of the current edge partition that are incident to v. Consider for example the vertex $q \in V'$ in the bottom left diagram of Figure 4.16: In the edge partition induced by $V' = \{o, q, r, t\}$, q is incident to four edge sets A_{in1}, A_{out1}, A_{in2}, and A_{out2}, say $(p, q) \in A_{\text{in1}}$, $(q, r) \in A_{\text{out1}}$, $(r, q) \in A_{\text{in2}}$, and $(q, p) \in A_{\text{out2}}$. The edge sets are acyclic, and A_{out2} and A_{in1} are opposite, and A_{out2} and A_{in1} are opposite as well. Thus, q does not appear to be a branching vertex but rather a vertex in the middle of a line of railroad tracks, and we eliminate q from V' and obtain the top right diagram in Figure 4.16 with $V' = \{o, r, t\}$. A general description of this operation is the following:

reduce: For every vertex $v \in V'$, if all edge sets incident to v are acyclic, then:

> **1.** If there is an edge set A_{in1} incident to v containing an edge (i_1, v) for some $i_1 \in V$ and another edge set A_{out1} incident to v containing an edge (v, o_1) for some $o_1 \in V$, if A_{in1} and A_{out1} are not opposite, and if there are no other edge sets incident to v except possibly a set A_{out2} that contains an edge (v, o_2) for some $o_2 \in V$ and that is opposite to A_{in1}, and a set A_{in2} that contains an edge (i_2, v) for some $i_2 \in V$ and that is opposite to A_{out1}, then v appears to be a vertex in the middle of a line of railroad tracks and is deleted from V'.
>
> **2.** If there is only one edge set incident to v, or if there are exactly two edge sets A_1 and A_2 incident to v and A_1 and A_2 are opposite and there are no line graph edges from an edge in A_1 to an edge in A_2 or vice versa, then v appears to be the end vertex of a line of railroad tracks without being a branching vertex and is eliminated from V'.

When a vertex v is deleted from V' in case 1. of a *reduce* operation, then A_{in1} and A_{out1} (as well as their opposites if they exist) may be united in the new induced edge partition. This of course depends on the existing line graph edges at v (but see Section 4.4 for the possible addition of artificial line graph edges). In Figure 4.14, Schopfheim is an example for such a vertex v.

The deletion of a vertex from V' in case 2. of a *reduce* operation will not cause any edge sets to be united in the new induced edge partition. In Figure 4.14, Zell is an example for such a vertex.

Now consider again the top right diagram in Figure 4.16: The vertex $r \in V'$ would not be eliminated from V' through a *reduce* step because it is incident to

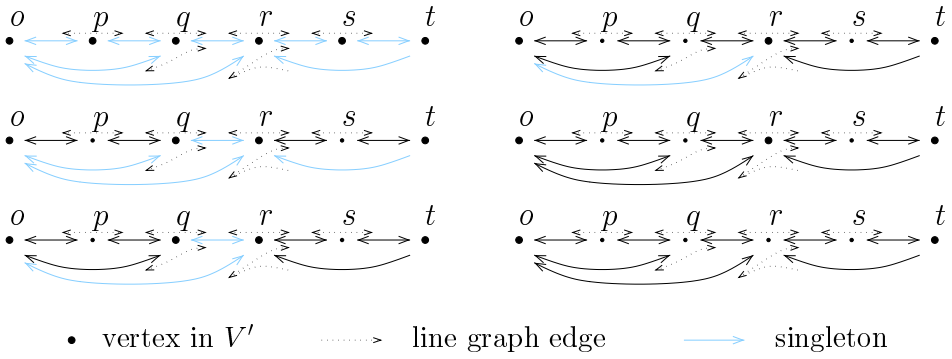

Figure 4.16: Growing bundles by alternately applying *reduce* and *unite-singleton*: Suppose we are starting with the instance in the top left diagram with 6 vertices, 15 edges and 13 line graph edges. The initial vertex set $V' = V$ induces a partition into singletons. *reduce* yields the diagram below with 4 nontrivial bundles (two in each direction; in black) and 7 singletons. *unite-singleton* yields again the diagram below (bottom left), where 3 singletons where united with nontrivial bundles. *reduce* then eliminates q from V', leaving still 4 nontrivial bundles but only two singletons (top right). Another *unite-singleton* (middle right) and *reduce* (bottom right) yield $V' = \{o, t\}$, two nontrivial bundles, and no singletons for this instance.

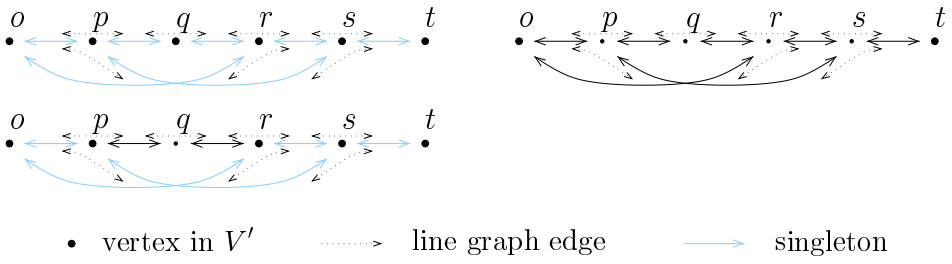

Figure 4.17: Where bundle growing does not work: Suppose we are starting with the instance in the top left diagram with 6 vertices, 14 edges and 14 line graph edges. The initial vertex set $V' = V$ induces a partition into singletons. *reduce* yields the diagram below with 2 nontrivial bundles (one in each direction; in black) and 10 singletons. No further reduction of the number of bundles by *reduce* or *unite-singleton* steps is possible. But if we start with $V' = \{o, t\}$ (top right), only two bundles (one in each direction) are induced.

six edge sets. But observe that for the singleton $\{(o, r)\}$, there is exactly one edge set A the Hamilton path of which starts with o and ends with r, and if we unite this singleton with A (and similarly for the singleton $\{(r, o)\}$), we have as a result (see the middle right diagram) that r is only incident to four edge sets. A *reduce* step can now eliminate r from V' (bottom right diagram). We formulate as the second ingredient of the bundle growing process besides *reduce* as

unite-singleton: For each singleton $\{(s, t)\}$ in the current edge partition:

If there is exactly one edge set with a Hamilton path that starts with s and ends with t, then unite $\{(s, t)\}$ with this edge set.

and conclude that an alternate application of *reduce* and *unite-singleton* as illustrated in Figure 4.16 makes for a bundle growing process:

Repeat

reduce

unite-singleton

until no more changes in V' occur.

Notice that incidentally, the bundles between Eberbach and Neckarelz in Figures 1.2 and 2.1 are very similar to the bundles grown in Figure 4.16, and that in the time table graph considered in the present chapter, there were for example two bundles between Tuttlingen and Sigmaringen that were formed through the bundle growing process (compare Figures 4.5 and 4.6). The bundle growing process worked so well in the examples considered so far, because the edges that later turned out to be transitive are nicely "nested". Figure 4.17, however, shows a situation without this nesting where the bundle growing process gets stuck. Figure 5.7 on page 92, which is based on $V_i' = V$, shows such a constellation in the German time table graph north of Basel: The bundles depicted north of Basel in Figure 5.6 were not grown in Figure 5.7.

So starting with a large initial vertex set and applying the bundle growing process may leave many unnecessary singletons in the resulting edge partition where bundles were not grown, and together with the concept of growing bundles, guessing a good initial vertex set where the induced edge partition already contains many bundles is therefore important for the success of the heuristic scheme.

Besides non-nested transitive edges, missing line graph edges may be another hindrance for the success of growing bundles. The next Section will deal with this problem.

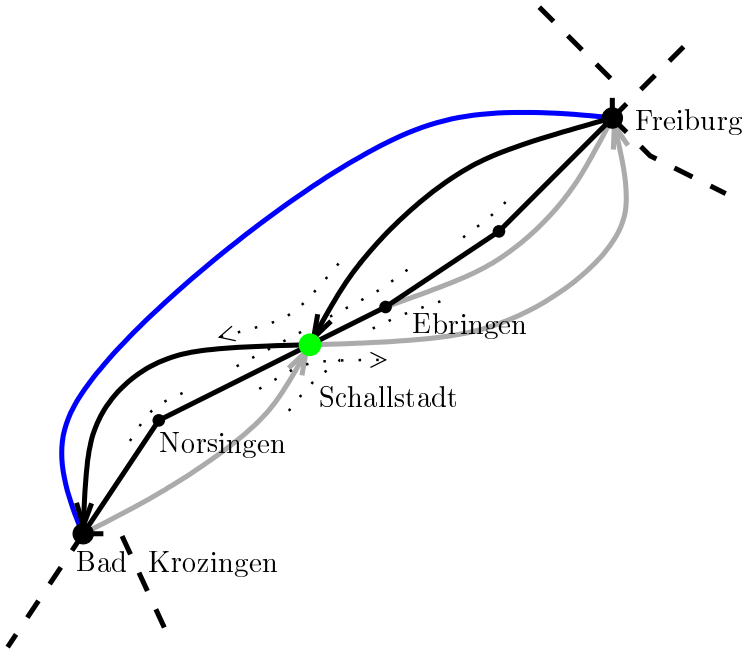

Figure 4.18: An initial partition: Detail around Schallstadt. The vertices Freiburg and Bad Krozingen are in V'. Edges (solid) and line graph edges (dotted) between Bad Krozingen and Freiburg that are shown undirected exist in both directions. Since there is no line graph edge ((Ebringen, Schallstadt), (Schallstadt, Bad Krozingen)) or ((Freiburg, Schallstadt),(Schallstadt, Norsingen)), the edges between Bad Krozingen and Freiburg are partitioned into five sets: Two singletons (blue), one edge set with Hamilton path (black) of length five from Bad Krozingen to Freiburg and with three transitive edges (grey), one edge set with the reverse Hamilton path of length five from Freiburg to Bad Krozingen and with one transitive edge, and one edge set with Hamilton path of length two from Freiburg to Bad Krozingen. Schallstadt (green) is the only vertex outside V' that is common to two edge sets that are not opposite of each other.

4.4 Artificial Line Graph Edges

Back in Section 2.2, situations like the one on the left side of Figure 2.4 on page 11 motivated the third condition in definition 2.5 of the bundle recognition problem:

3. *Two distinct edge sets do not have vertices outside V' in common, except possibly if the edge sets are opposite.*

Figure 4.18 gives a concrete example around Schallstadt taken from the time table graph used throughout this chapter. It shows how in the initial partition shown in Figure 4.4, through "missing" line graph edges, two edges that should belong to another edge set as transitive edges form the Hamilton path of their

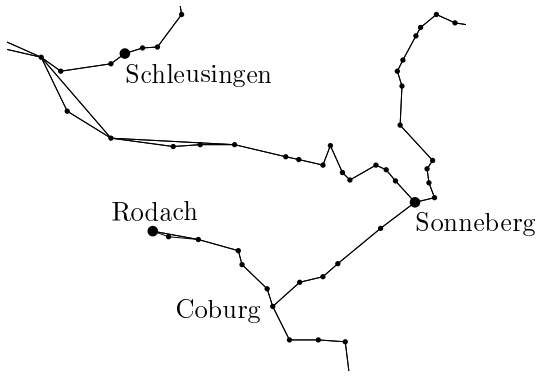

Figure 4.19: Detail of Figure 4.20 in the Thuringian Forest in Germany.

own edge set and would therefore wrongly be classified as real if the third bundle condition were not enforced. In *first augment* shown in Figure 4.5, Schallstadt is detected as a "common" vertex and added to V'. In the bundle growing process, a *unite-singleton* will unite the four blue edges that are incident to Schallstadt with one edge set each. Then *reduce* (case 1. on page 58) will take Schallstadt out of V' again. If *reduce* now were to add the extra line graph edge ((Ebringen, Schallstadt), (Schallstadt, Bad Krozingen)), for example, then in the induced edge partition, the edges between Bad Krozingen and Freiburg would form two opposite bundles, each with a Hamilton path of length five and with three transitive edges. These two bundles would not violate the third bundle condition anymore, and more edges could be classified than when Schallstadt is in V'.

Similarly, consider Figure 4.19: At big vertices (Rodach, Sonneberg, and Schleusingen), trains only begin or end, whereas for every small vertex, there is at least one train that induces a line graph edge at that vertex. Rodach is the end of a railroad line, and Rodach and Coburg should be recognized as bundle end points. Since no train induces a line graph edge at Rodach, it is irrelevant for the induced partition whether the heuristic places Rodach in V' or not. Sonneberg is a branching point of the railroad network, and the heuristic will insert it in V'. But Schleusingen should intuitively be an inner vertex of a bundle. If Schleusingen does not belong to the initial V', then it will be added to V' in *first augment*, again because of the third bundle condition. In the bundle growing process, *reduce* (case 1.) will take Schleusingen back out of V'. If *reduce* now were to add extra line graph edges at Schleusingen, then in the induced edge partition, the edges incident to Schleusingen would form two opposite bundles, the edge partition would not violate the third bundle condition anymore, and Schleusingen would not have to belong to V'. Out of the 28 280 vertices in the European time table graph for the summer of 1997, there are 1 217 without any line graph edges induced by trains (*non-intermediate* vertices). Close inspection

of pictures like Figure 4.20 indicates that most of them are like Rodach, some are like Sonneberg, and only few are like Schleusingen.

Motivated by cases like Schallstadt in Figure 4.18 and Schleusingen in Figure 4.19, we add artificial line graph edges as follows during the bundle growing process: If in *reduce* (case 1.) there are more than two edge sets incident to the vertex $v \in V'$ under consideration, and if v is intermediate or if both A_{in1} and A_{out1} contain at least two edges, and if $A_{\text{in1}} \cup A_{\text{out1}}$ is acyclic, then we assume that A_{in1} and A_{out1} are actually part of the same intuitive bundle and add artificial line graph edges $((i, v), (v, o))$ for all edges $(i, v) \in A_{\text{in1}}$ and for all edges $(v, o) \in A_{\text{out1}}$ besides taking v out of V'. The same is applied to A_{in2} and A_{out2}, if they exist. The conditions required for adding artificial line graph edges are intended to avoid accidentally adding artificial line graph edges at a vertex that is the end point of a line of railroad tracks.

As for Schallstadt from Figure 4.18, Figure 4.6 shows that Schallstadt was deleted from V' during the bundle growing process. Because of the artificial line graph edges added at Schallstadt, the edge sets incident to Schallstadt in the induced edge partition after the bundle growing process do not violate the bundle conditions, so that Schallstadt is not added to V' in *final augment* and the edges between Bad Krozingen and Freiburg form two opposite bundles in the final partition shown in Figure 4.8.

If the heuristic is applied to the same time table graph and with the same initial vertex set as in Section 4.1 but without adding artificial line graph edges, then the number of Hamilton path and transitive edges decreases, while the number of singletons increases:

	nontrivial bundles	Hamilton path edges	transitive edges	singletons
with a.l.g.e.	10 146	52 973	13 555	16 066
without a.l.g.e.	11 336	52 863	13 004	16 727

The computational results given in Chapter 5 will always reflect the results of the heuristic including the addition of artificial line graph edges as described in this section.

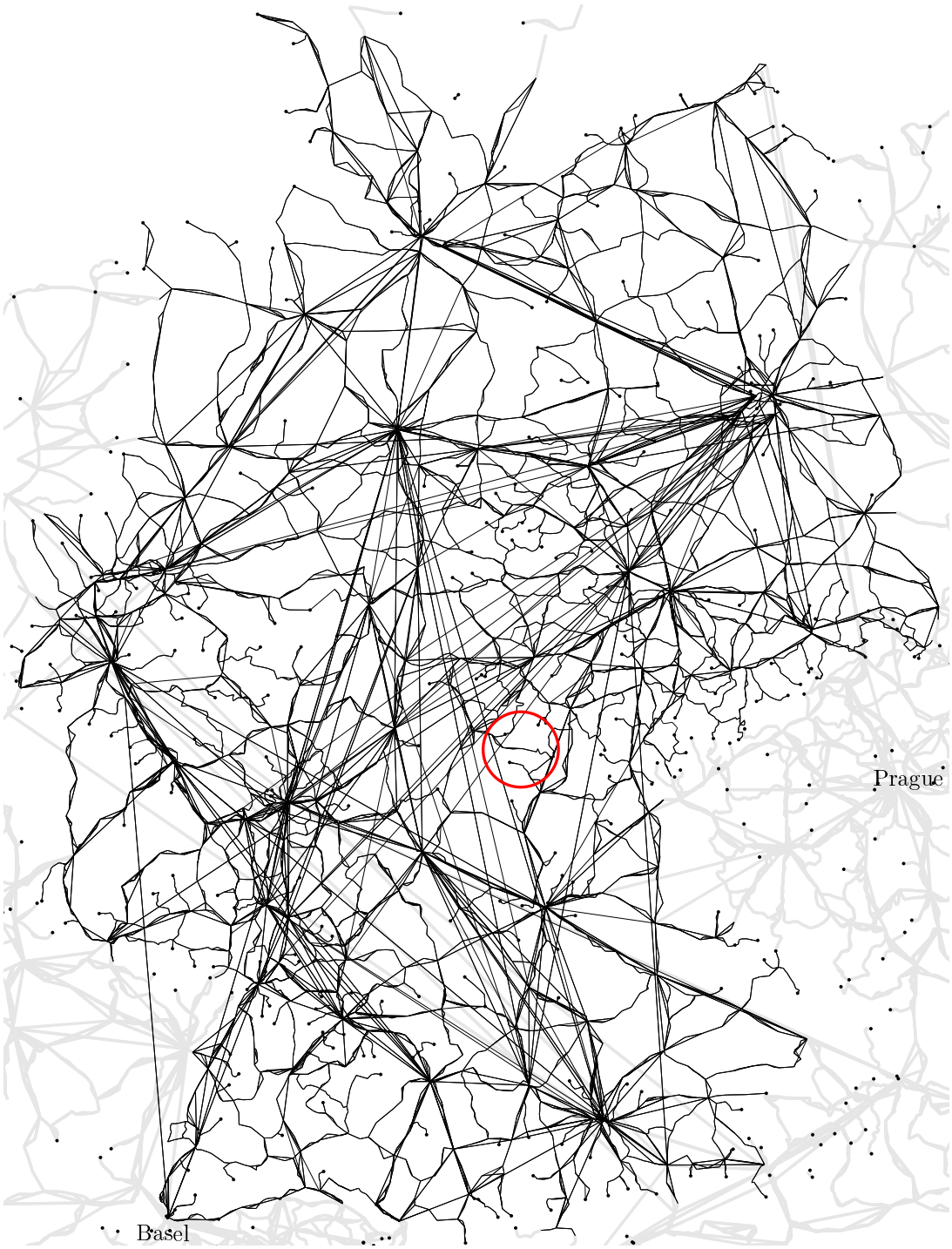

Figure 4.20: Non-intermediate vertices (in black). Edges within Germany are black. Figure 4.19 shows the encircled area in detail.

Chapter 5

Computational Study

The purpose of this chapter is to evaluate the performance of the bundle recognition heuristic for the initial vertex sets discussed in Section 4.2 and combinations of them on the one hand, and for different sets of input data drawn from the time tables of summer 1997 on the other hand. Section 5.1 reports results for the heuristic for the whole European time table graph of the summer of 1997 for various initial vertex sets. The results are broken down for 13 European countries, and it turns out that the performance of the heuristic varies considerably not only from initial vertex set to initial vertex set but also from country to country.

Section 5.2 then reports results on running the heuristic on four time table graphs induced only by the trains traveling entirely within one country. It turns out that the countries yielding good or bad results in Section 5.1 behave similarly when the heuristic is run on their "national" time table graphs in Section 5.2. Furthermore, the choice of initial vertex set yielding the best results for one country in Section 5.1 also yields the best results for the "national" time table graph of this country in Section 5.2.

Recall that the output of the heuristic is a vertex set inducing an edge partition of the time table graph into singletons and nontrivial bundles. The edges forming the singletons remain unclassified, but within the nontrivial bundles, the edges forming the Hamilton path are classified as *real*, and the other edges of the bundle are *transitive*.

Since no quality criterion for the correctness of the edge classifications resulting from the bundle recognition heuristic is available except for studying the visualization of the results with a human eye, and since this visualization suggests that the classifications resulting from the bundle recognition heuristic are conservative in that they do not classify incorrectly but rather leave edges unclassified, calculating the percentage of classified edges is a reasonable measure for the success of the bundle recognition heuristic. In the remainder of this chapter, the results of the heuristic will be evaluated for different initial vertex sets and for different input data sets in terms of the percentage of classified edges. It will turn out that the percentage of edges classified through bundle recognition

actually indicates to what extent the bundle structure is present in different time table graphs.

In summary, 81 percent of the European time table graph edges can be classified as *real* or *transitive* as a result of the bundle recognition heuristic. Counted separately for 13 countries, the highest percentage of classified edges (93) is reached for Slovakia, and the lowest (62) is obtained for Great Britain. The percentages for the other 11 countries are spread between 75 and 78 for five of them, and between 85 and 91 for the remaining six.

Running the heuristic for one initial vertex set on the European time table graph for the summer of 1997 with its 28 280 vertices and 82 594 edges takes roughly 15 minutes on a Sun Microsystems Enterprise 4000/5000 server with 1024 MB main memory and an 336 MHz UltraSPARC-II CPU.

5.1 The European Time Table Graph

Starting on page 73, Tables 5.1 through 5.11 list results of the heuristic for the time table graph induced by the European train time tables of the summer of 1997. Columns for Poland, the Czech Republic, Slovakia, Great Britain, Sweden, Germany, Austria, Italy, The Netherlands, Switzerland, Denmark, France and Belgium indicate the counts for vertices and edges that lie in each country, respectively. These countries have between 375 (The Netherlands) and 6 505 (Germany) vertices, while European countries not listed have less than 150 vertices in the European time table graph. Their vertices and edges only appear in the column *Europe*. Also, edges crossing a country border are only accounted for in this column. Abbreviations used for the table rows are explained in Figure 5.1 on page 73.

The results of the heuristic are reported for 34 different initial vertex sets. The simplest way to guess an initial vertex set is of course to take the whole vertex set of the time table graph: $V_i' = V$. Using it for the heuristic means to rely entirely on the bundle growing process as described in Section 4.3. In the resulting edge partition as reported in Table 5.1, 77 percent of the edges of the European time table graph can be classified as *real* or *transitive*, while 23 percent (or 18 928 edges out of 82 594) remain unclassified. All percentages of classified edges reported in this chapter are rounded down to the nearest integer.

Recalling from Section 4.3 that the bundle growing process does not always work well, one might expect that $V_i' = V$ yields rather poor results in comparison with other initial vertex sets, and this is indeed so: Table 5.1 shows that for the European time table graph as a whole as well as for 11 out of the 13 countries accounted for individually, this choice of V_i' yields the lowest percentage of classified edges among all 34 initial vertex sets considered.

5.1.1 Criteria for an initial vertex set as discussed in Section 4.2

We will now examine the results of the heuristic with the choices of initial vertex sets suggested in Section 4.2, starting with vertices of high vertex degree in relation to that of their neighbors that were discussed in Section 4.2.1. Table 5.2 reports the results for the heuristic with the four initial vertex sets depicted in Figure 4.10 on page 52. Each of the four sets allows the classification of 79 or 80 % of the edges of the European time table graph, and among the four sets, vertices with (undirected) degree larger than that of at least 60 % of their neighbors (i.e. initial vertex set number 3) yields the highest number of classified edges.

Table 5.3 then shows the results for the initial vertex sets taken as branching vertices in minimum spanning forests or their union as suggested in Section 4.2.2 and illustrated in Figure 4.11 on page 53. These initial vertex sets lead to the classification of 79 % of the edges, and the set of vertices with degree at least three in the union of the two (undirected) minimum spanning forests (initial vertex set number 8) is a little bit better than the set of branching vertices of one of the minimum spanning forests alone.

In Section 4.2.3, vertices fulfilling angle conditions in the straight line embedding of a time table graph are suggested for initial vertex sets. Table 5.4 reports the results of the heuristic for the vertex sets illustrated in Figure 4.12 on page 54, where the sum of the two largest angles incident to a vertex may not be too large. Similarly, Table 5.5 shows the results for the initial vertex sets illustrated in Figure 4.13, where the third largest angle incident to a vertex may not be too small. These sets yield percentages of 79 to 80, and 78 to 79 of classified edges, respectively. The best results within each table are achieved for vertices with the sum of their two largest incident angles not exceeding 300° (initial vertex set number 10), and for vertices with the third largest angle measuring at least 30° (initial vertex set number 14), respectively. Assuming that each of these two vertex sets probably misses some potential bundle end points, the union of the two vertex sets is considered as initial vertex set number 17 in Table 5.6, and the union of initial vertex sets 11 and 15 is also considered in Table 5.6. The numbers of edges classified as a result of these unions lie in between the results for vertex sets number 10 and 11 on the one side, and the results for vertex sets number 14 and 15 on the other side, so nothing was gained by considering these set unions. But in Section 5.1.2 we will find examples where combining two good guesses of an initial vertex set may yield an even better one.

The set of terminal vertices of trains suggested in Section 4.2.4 and illustrated in Figure 4.14 on page 56 is the initial vertex set number 19. As Table 5.6 shows, 80 % of the edges can be classified using this initial vertex set. In particular, the number of classified edges is the highest achieved for any initial vertex set so far.

The fifth and last criterion to guess an initial vertex set suggested in Section 4.2 is the "importance number" of vertices, a value that is assigned to each

train station by the *TLC*. Figure 4.15 on page 57 shows vertices with high importance numbers, and Table 5.7 reports the results of the heuristic using the four initial vertex sets illustrated in Figure 4.15. It turns out that the vertices with importance number higher than that of 80 % of all vertices (i.e. initial vertex set number 22) form a very good initial vertex set that allows the classification of 81 % of all edges, superseding even the set of terminal vertices discussed in the previous paragraph.

In summary, the three best single criteria for an initial vertex set were the vertices with importance number higher than that of 80 % of all vertices as best choice, followed by the terminal vertices that leave 417 more edges unclassified than the best choice does, and by the vertices with vertex degree larger than that of at least 60 % of their neighbors, reducing the number of classified edges again by 117. It is an interesting outcome of this comparative study that the vertex set based on a classification of important vertices by humans actually yields better results than any of the criteria that are trying to formalize the intuition of what a branching vertex is by considering vertex degrees, minimum spanning forests or angles. And also the second best choice for an initial vertex set, the terminal vertices, is directly determined by decisions of humans at train companies where to let trains begin or end, instead of by some property derived from the structure of the time table graph.

Considering the choices of initial vertex sets that involve numerical parameters, i.e. the sets based on vertex degree, angles or importance numbers, it is of course rather likely that for a given parameter, a higher number of classified edges could be achieved using a parameter value not tried out yet. One would expect such an effect for a parameter value close to the one that yielded the best result so far for this parameter, but even with a completely different parameter value, a (maybe even large) increase in classified edges is theoretically possible. But firstly, for the four choices of initial vertex sets involving numerical parameters (Tables 5.2, 5.4, 5.5, and 5.7), the parameter values for which results are reported were chosen in such a way that the parameter value yielding the best result for the particular parameter lies somewhere in the middle of the parameter range considered. And secondly, for each of the four choices involving numerical parameters, the sequence of the cardinalities of the initial vertex sets is monotonic as the parameter increases. Thirdly, if we consider a parameter value at the end of a parameter range as for initial vertex sets number 5, 12, 16, or 23 at the bottom of Tables 5.2, 5.4, 5.5, and 5.7, where the initial vertex set contains considerably less vertices than the initial vertex set that yielded better results for this parameter, extending the parameter range beyond the bottom of the table would further decrease the cardinality of the initial vertex set. The heuristic would then tend to perform even worse because already for these initial vertex sets 5, 12, 16, and 23, there are probably many bundle end points missing from the vertex set. But many vertices missing from the initial vertex set leads to high numbers of vertices that are not bundle end points being added in *first augment*, and to many

of these vertices remaining in the vertex set because the bundle growing process is not able to eliminate them. Likewise, each of the initial vertex sets number 2, 9, 13, and 20 at the other end of the four parameter ranges contains considerably more vertices than the initial vertex set that yielded better results for this one parameter. Extending the parameter ranges beyond the top of Tables 5.2, 5.4, 5.5, and 5.7 would further increase the cardinality of the initial vertex set. Again, the heuristic would then tend to perform worse because already the initial vertex sets number 2, 9, 13, and 20 apparently contain too many vertices that the bundle growing process is unable to eliminate. Together, these three observations give rise to the hope that the parameter ranges for the initial vertex sets based on vertex degrees, angles, and importance numbers are well chosen in the sense that there probably is no value of the parameter outside the range considered in Tables 5.2, 5.4, 5.5, and 5.7 that will yield better results than the best results found for each particular parameter so far.

Finally, since the variation of the percentages of classified edges within each of the four parameter ranges considered lies within only two percentage points between the best initial vertex set for this parameter and the two adjacent parameter values considered, there is no real need to fine tune the parameter values further, but we can be content with those four parameter values found that yield the highest numbers of classified edges per parameter.

5.1.2 Combining criteria for initial vertex sets

In order to still improve the number of classified edges, one might combine the initial vertex sets considered so far in numerous ways. To keep the number of combinations of vertex sets to be tested within a practical range, we combine each of the three best initial vertex sets so far (i.e. sets number 3, 19, and 22) with the initial vertex sets based on minimum spanning forests (i.e. sets number 6, 7, and 8). The reason for using these latter vertex sets is the assumption that while they might not make very good initial vertex sets by themselves because they are missing many bundle end points, they will tend to not contain many vertices that are not bundle end points. So combining them with another good initial vertex set will tend to only add bundle end points to this already good set and to not add any unnecessary vertices that the heuristic might not be able to eliminate in the bundle growing process.

Uniting the set of vertices with degree larger than that of at least 60 % of their neighbors (initial vertex set number 3) with one of the three vertex sets based on minimum spanning forests yields the three initial vertex sets for which the results of the heuristic are reported in Table 5.8. Unfortunately, the positive effect of combining vertex sets that we were hoping for is not very drastic: For the two unions with the vertices of degree at least three in one of the minimum spanning forests (vertex sets 24 and 25), the numbers of classified edges are higher than the numbers of classified edges with initial vertex sets 6 and 7 alone, and they

are slightly higher (less than 100 edges) than the number of classified edges with initial vertex set 3 alone, but they are several hundred edges below the number of classified edges achieved by the best initial vertex set so far (number 22). And the union of vertex set 3 with vertex set 8 even yields slightly less classified edges than vertex set 3 alone.

Table 5.9 shows that combining terminal vertices (initial vertex set number 19) with one of the sets 6, 7, and 8, respectively, yields a higher number of classified edges than any single one of the four initial vertex sets by itself. And for the union of vertex set 19 with vertex set 8, the heuristic even comes close to the best result so far.

The effect of combining vertices with importance number higher than that of 80 % of all vertices (initial vertex set number 22) with vertex sets 6, 7, and 8 is reported in Table 5.10. The outcome is similar to the combinations with vertex set 3: For the two unions with the vertices of degree at least three in one of the minimum spanning forests (vertex sets 30 and 31), the numbers of classified edges are higher than the numbers of classified edges with initial vertex sets 6 and 7 alone, and they are slightly higher (roughly 100 edges) than the number of classified edges with initial vertex set 22 alone. But the union of vertex set 22 with vertex set 8 yields less classified edges than vertex set 22 alone.

In summary, the combinations of vertices with importance number higher than that of 80 % of all the vertices with vertices that have degree at least three in a minimum spanning forest form the two vertex sets with highest numbers of classified edges so far, and the minimum spanning forest based on travel times (set 31) yields even slightly better results than the Euclidean minimum spanning forest (set 30).

Taking the combination of initial vertex sets even further, Table 5.11 shows the results of combining more than two criteria for initial vertex sets with the idea that already motivated the combination sets numbered 24 through 32: Unite several vertex sets, each of which either performs very well by itself, or tends to add only vertices that are actually bundle end points.

To use the angles incident to vertices in the straight line embedding of the time table graph to characterize branching points of the physical railroad network, observe that a branching in three evenly spread directions would induce an angle of 120° between directions. Allowing a slightly uneven spreading of the directions, we can hope that vertices with the sum of their two largest incident angles not exceeding 270°, and vertices with their third largest incident angle being at least 90° wide will all be branching points of the physical railroad network. Vertex set number 33 combines these two criteria with vertex set number 31. The resulting number of classified edges becomes the second best among all initial vertex sets considered, exceeding that of vertex set number 30, but not that of vertex set number 31. With vertex set number 34 we try out whether it helps to restrict the vertex sets united to form set 33, but the number of classified edges using this last initial vertex set does not come within reach of the results achieved by

vertex sets 30, 31, and 33.

Of course numerous other combinations of initial vertex sets could be tested, and it is very likely that a higher number of classified edges than was achieved with vertex set number 31 is possible. But since we know from Section 2.1 that time table graphs do contain edges that will not be classified by the bundle recognition approach, it is well possible that the number of classified edges for the European time table graph considered in this chapter can actually not be increased drastically. And notice that not only the initial vertex sets number 22, 28, 30, 31, and 33 each yield 81 % classified edges, but that the fifteen initial vertex sets numbered 3, 4, 9, 10, 11, 19, 21, 24, 25, 26, 27, 28, 29, 32, and 34 each lead to a classification of still 80 % of the edges, and that in particular, all the vertex set combinations that were considered lead to a classification of at least 80 % of the edges. So there are initial vertex sets based on a diverse set of sensible criteria that all lead to the classification of 80 or 81 % of the edges of the European time table graph, and it is this uniformity in behaviour of vertex sets constructed in very different ways that suggests that we have indeed found reasonably good initial vertex sets for the bundle recognition heuristic.

5.1.3 Breaking down the results for 13 countries

Tables 5.1 through 5.11 report the results for each of the 34 initial vertex sets for each one of 13 countries by counting only the vertices and edges within each country. The results vary considerably from country to country: Not only is the gap between 93 % classified edges for Slovakia and 62 % classified edges for Great Britain rather wide. Also notice that the difference between the highest and the lowest percentage of classified edges for a single country varies greatly between one percentage point for Poland and 12 percentage points for Italy.

Germany contributes more than one fifth of the edges to the European time table graph, which makes it by far the largest country. This is not surprising, since the *TLC* which collects the train time tables that induce the time table graph is a German company. The next largest countries in terms of $|A|$ are France, Poland, Great Britain, and Italy, in that order.

So Great Britain with its exceptionally low percentage of 62 % classified edges is among these five largest countries. In Section 5.2, we will see what might make Great Britain such a difficult case for the bundle recognition heuristic. And since the second lowest percentage of classified edges is already 75 (for Belgium), the 78 % classified edges achieved for Italy and France must also be considered a low result. The 85 % classified edges for Germany are a medium result, and on the high side, the heuristic classifies 89 % of the Polish time table graph edges.

Also the five smallest countries The Netherlands, Denmark, Sweden, Belgium, and Slovakia display low percentages of classified edges (Belgium, The Netherlands, and Sweden with 75, 76, and 77 percent classified edges, respectively), a medium percentage of 86 % for Denmark, and then the highest result of 93 %

classified edges for Slovakia.

The set of vertices with importance number larger than that of 80 % of all vertices and of vertices that have degree at least three in the minimum spanning forest based on travel times (vertex set number 31) achieves not only best results for the whole European time tables graph as well as for five of the 13 countries considered, but for two of the remaining countries, it actually yields the same percentage of classified edges as the best choice for these countries (namely 89 % for Poland and 88 % for Switzerland). And for Great Britain, Sweden, Germany, and Italy, the percentages of classified edges using initial vertex set number 31 are only one percentage point below the best percentages reached for these countries. Finally, for The Netherlands and Denmark, the percentages of classified edges reached with this initial vertex set are two percentage points below the best percentages reached for these countries. So all in all, the initial vertex set number 31 seems to be a universally good initial vertex set for the bundle recognition heuristic, just as the first initial vertex set $V_i' = V$ is universally bad.

But the initial vertex set number 28, which leads to the classification of 80 % of the European time table graph edges and achieves best classification results for six countries, leads to classification percentages for Austria and Denmark that are 3 percentage points below the best results achieved for these countries. And the set of terminal vertices (initial vertex set number 19) even achieves best results for two countries (Sweden and Switzerland) and at the same time yields the lowest number of classified edges for another country (The Netherlands). So we conclude that an initial vertex set is not necessarily universally good or bad for all countries.

$\|V\|$	:	numbers of vertices
$\|A\|$	:	numbers of edges
$\|V_i'\|$	:	numbers of vertices in the initial vertex set under consideration
$\|V_f'\|$	:	numbers of vertices in the final vertex set, i.e. after *final augment*
s	:	numbers of singletons in the final edge partition, i.e. after *final unite-singleton*
t	:	numbers of transitive edges in the final partition
H	:	numbers of Hamilton path edges in the final partition
b	:	numbers of nontrivial bundles in the final partition
$\underline{\%}$	:	In Tables 5.1 through 5.11, the lowest percentage of edges classified through the bundle recognition heuristic taken over all 34 initial vertex sets is reported for each column. The corresponding numbers of transitive and Hamilton path edges for this worst choice of V_i' are also underlined in the tables. In Table 5.12, the lowest percentage of edges classified is taken over the four initial vertex sets considered in this table only.
%	:	Analogously, the highest percentages of edges classified are also given, and the corresponding numbers of transitive and Hamilton path edges are marked in bold face also.

Figure 5.1: Legend for Tables 5.1 through 5.12

	PL	**CZ**	**SL**	**GB**	**S**	**D**	**A**	**I**	**NL**	**CH**	**DK**	**F**	**B**	**Europe**
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
1) Choose the initial vertex set $V_i' = V$:														
$\|V_f'\|$	478	464	126	962	153	1 270	274	838	117	280	102	1 031	147	6 661
s	987	872	212	3 581	391	3 115	649	2 743	300	591	235	3 381	462	18 928
t	1 216	1 199	391	997	173	2 159	719	1 564	140	635	132	2 138	254	12 107
H	6 655	5 298	1 796	3 933	1 042	12 519	3 095	3 889	668	3 360	850	5 613	952	51 559
b	1 029	974	275	1 003	228	2 577	448	914	185	590	144	1 238	225	10 363
$\underline{\%}$	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.1: Computational results with $V_i' = V$

	PL	**CZ**	**SL**	**GB**	**S**	**D**	**A**	**I**	**NL**	**CH**	**DK**	**F**	**B**	**Europe**
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
2) V_i' contains the vertices with degree larger than that of at least 50 % of their neighbors:														
$\|V_i'\|$	655	637	214	650	140	1 503	416	713	97	401	119	955	147	6 975
$\|V_j'\|$	468	434	111	899	149	1 153	245	660	120	267	84	885	145	6 028
s	965	758	163	3 307	380	2 694	559	2 056	306	562	184	2 818	450	16 570
t	1 230	1 249	412	1 126	178	2 352	763	1 906	138	650	155	2 410	258	13 234
H	6 663	5 362	1 824	4 078	1 048	12 747	3 141	4 234	664	3 374	878	5 904	960	52 790
b	1 019	977	272	1 017	228	2 554	436	899	185	578	140	1 236	226	10 298
3) V_i' contains the vertices with degree larger than that of at least 60 % of their neighbors:														
$\|V_i'\|$	412	425	149	434	103	1 001	260	512	74	266	87	626	92	4 679
$\|V_j'\|$	461	424	107	879	146	1 132	228	630	120	267	79	847	139	5 858
s	939	729	150	3 237	381	2 642	525	1 981	306	561	169	2 681	433	16 066
t	1 250	1 268	**419**	1 162	177	2 388	791	1 956	138	650	**160**	2 503	267	13 555
H	6 669	5 372	**1 830**	4 112	1 048	12 763	3 147	4 259	664	3 375	**888**	5 948	968	52 973
b	1 011	968	270	1 012	221	2 532	409	864	185	576	140	1 205	224	10 146
4) V_i' contains the vertices with degree larger than that of at least 70 % of their neighbors:														
$\|V_i'\|$	241	231	77	252	60	613	153	307	42	149	49	362	48	2 724
$\|V_j'\|$	461	430	109	893	146	1 167	235	601	118	282	80	844	142	5 909
s	932	758	154	3 292	381	2 762	560	1 924	298	616	169	2 699	439	16 319
t	1 244	1 255	415	1 141	177	2 330	776	2 007	144	634	**160**	2 514	263	13 487
H	6 682	5 356	1 830	4 078	1 048	12 701	3 127	4 265	666	3 336	**888**	5 919	966	52 788
b	1 024	967	273	1 007	221	2 543	402	813	183	572	142	1 172	228	10 073
5) V_i' contains the vertices with degree larger than that of at least 80 % of their neighbors:														
$\|V_i'\|$	164	163	52	173	38	409	102	198	27	113	32	224	29	1 820
$\|V_j'\|$	471	448	117	918	147	1 195	234	679	115	285	80	912	142	6 139
s	959	831	175	3 417	381	2 866	539	2 213	287	614	169	2 951	441	17 170
t	1 229	1 220	410	1 073	177	2 278	783	1 851	151	632	**160**	2 400	267	13 064
H	6 670	5 318	1 814	4 021	1 048	12 649	3 141	4 132	670	3 340	**888**	5 781	960	52 360
b	1 032	964	273	1 003	221	2 551	415	840	181	580	142	1 172	223	10 118
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.2: Computational results with V_i' based on vertex degrees as suggested in Section 4.2.1 and illustrated in Figure 4.10 on page 52

	PL	CZ	SL	GB	S	D	A	I	NL	CH	DK	F	B	Europe
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
6) V_i' contains vertices that have degree at least three in the Euclidean MSF:														
$\|V_i'\|$	221	241	73	262	63	575	100	176	47	135	34	266	57	2 395
$\|V_f'\|$	456	434	111	912	148	1 170	243	673	113	265	87	901	140	6 053
s	931	776	160	3 386	387	2 745	551	2 204	284	560	190	2 893	433	16 848
t	1 251	1 243	415	1 104	175	2 343	772	1 838	148	654	155	2 404	269	13 192
H	6 676	5 350	1 824	4 021	1 044	12 705	3 140	4 154	676	3 372	872	5 835	966	52 554
b	1 006	967	271	988	222	2 550	433	852	183	574	140	1 192	225	10 123
7) V_i' contains vertices that have degree at least three in the MSF based on travel times:														
$\|V_i'\|$	222	219	67	264	65	595	89	176	44	135	35	224	57	2 358
$\|V_f'\|$	454	434	116	910	148	1 182	236	625	111	263	83	882	138	5 984
s	923	774	184	3 387	387	2 803	558	2 059	270	552	185	2 817	423	16 686
t	1 253	1 247	403	1 101	175	2 311	773	1 922	154	656	152	2 457	271	13 287
H	6 682	5 348	1 812	4 023	1 044	12 679	3 132	4 215	684	3 378	880	5 858	974	52 621
b	1 006	966	270	983	222	2 553	409	815	185	574	140	1 181	227	10 048
8) V_i' contains vertices that have degree at least three in the union of the two MSFs:														
$\|V_i'\|$	269	311	101	373	79	678	135	243	51	165	49	349	80	3 105
$\|V_f'\|$	460	436	117	900	145	1 173	235	598	111	262	80	905	140	5 965
s	941	778	180	3 329	380	2 781	527	1 931	270	553	169	2 898	433	16 535
t	1 246	1 247	405	1 131	178	2 319	786	1 985	154	655	**160**	2 401	269	13 346
H	6 671	5 344	1 814	4 051	1 048	12 693	3 150	4 280	684	3 378	**888**	5 833	966	52 713
b	1 011	964	273	992	220	2 550	427	824	185	570	142	1 196	225	10 101
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.3: Computational results with V_i' based on minimum spanning forests as suggested in Section 4.2.2 and illustrated in Figure 4.11 on page 53

	PL	CZ	SL	GB	S	D	A	I	NL	CH	DK	F	B	Europe
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
9) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 315°$:														
$\|V_i'\|$	317	413	113	450	88	813	226	396	73	202	54	601	83	4 073
$\|V_f'\|$	460	435	115	890	146	1 150	221	611	113	259	83	905	140	5 932
s	933	768	173	3 250	373	2 714	487	1 934	278	538	185	2 930	433	16 367
t	1 241	1 249	408	1 164	181	2 362	802	1 970	148	658	152	2 391	269	13 400
H	6 684	5 352	1 818	4 097	1 052	12 717	3 174	4 292	682	3 390	880	5 811	966	52 827
b	1 022	970	275	1 019	226	2 531	419	865	187	576	140	1 186	225	10 163
10) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 300°$:														
$\|V_i'\|$	262	321	86	361	74	667	155	309	57	163	40	462	68	3 227
$\|V_f'\|$	458	439	117	884	144	1 162	228	566	115	259	83	905	142	5 901
s	931	778	177	3 243	370	2 747	508	1 807	292	539	185	2 913	437	16 275
t	1 243	1 245	404	1 161	182	2 347	797	2 045	142	665	152	2 409	265	13 475
H	6 684	5 346	1 818	4 107	1 054	12 699	3 158	4 344	674	3 382	880	5 810	966	52 844
b	1 018	972	279	1 012	223	2 535	421	822	185	570	140	1 181	229	10 105
11) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 285°$:														
$\|V_i'\|$	213	251	67	282	58	541	113	241	43	130	35	349	55	2 539
$\|V_f'\|$	458	437	114	878	146	1 174	238	581	115	262	87	901	142	5 932
s	930	766	164	3 245	376	2 784	551	1 852	292	536	197	2 896	437	16 363
t	1 244	1 247	411	1 165	180	2 332	784	2 034	142	669	146	2 424	265	13 469
H	6 684	5 356	1 824	4 101	1 050	12 677	3 128	4 310	674	3 381	874	5 812	966	52 762
b	1 016	976	277	993	224	2 539	411	816	185	574	140	1 171	229	10 073
12) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 270°$:														
$\|V_i'\|$	177	183	52	212	46	417	79	186	35	96	27	257	44	1 932
$\|V_f'\|$	460	443	114	896	149	1 201	247	626	115	279	89	912	142	6 073
s	946	783	164	3 304	380	2 862	578	2 004	287	583	203	2 921	437	16 795
t	1 244	1 238	411	1 140	178	2 294	775	1 958	147	656	146	2 418	265	13 296
H	6 668	5 348	1 824	4 067	1 048	12 637	3 110	4 234	674	3 347	868	5 793	966	52 503
b	1 008	980	277	997	228	2 550	410	837	183	572	138	1 177	229	10 105
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.4: Computational results with V_i' based on $\alpha_1 + \alpha_2$ being "not too large" as suggested in Section 4.2.3 and illustrated in Figure 4.12 on page 54

	PL	CZ	SL	GB	S	D	A	I	NL	CH	DK	F	B	Europe
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
13) V_i' contains vertices with angle condition $\alpha_3 \geq 20°$:														
$\|V_i'\|$	425	542	169	619	112	1 113	348	590	102	262	85	863	104	5 661
$\|V_f'\|$	464	456	114	937	149	1 183	250	705	117	273	97	940	140	6 231
s	942	845	173	3 474	380	2 839	595	2 248	300	562	212	3 082	433	17 458
t	1 236	1 210	408	1 054	178	2 287	744	1 812	140	642	155	2 299	269	12 832
H	6 680	5 314	1 818	3 983	1 048	12 667	3 124	4 136	668	3 382	850	5 751	966	52 304
b	1 026	974	273	1 000	228	2 547	429	896	185	594	136	1 196	225	10 242
14) V_i' contains vertices with angle condition $\alpha_3 \geq 30°$:														
$\|V_i'\|$	342	410	118	445	91	871	246	385	80	196	58	602	80	4 197
$\|V_f'\|$	462	451	117	910	146	1 177	251	657	117	265	83	943	140	6 121
s	934	808	177	3 348	373	2 824	574	2 119	300	548	185	3 089	433	17 071
t	1 240	1 225	404	1 118	181	2 304	757	1 875	140	650	152	2 286	269	13 007
H	6 684	5 336	1 818	4 045	1 052	12 665	3 132	4 202	668	3 388	880	5 757	966	52 516
b	1 026	986	279	1 004	226	2 532	434	866	185	586	140	1 210	225	10 228
15) V_i' contains vertices with angle condition $\alpha_3 \geq 40°$:														
$\|V_i'\|$	255	301	88	328	77	676	162	269	56	153	42	388	67	3 076
$\|V_f'\|$	469	461	117	920	149	1 185	240	661	115	265	83	951	147	6 174
s	966	858	177	3 397	380	2 836	556	2 136	292	554	185	3 113	462	17 304
t	1 224	1 203	404	1 093	178	2 294	779	1 868	142	650	152	2 301	254	12 944
H	6 668	5 308	1 818	4 021	1 048	12 663	3 128	4 192	674	3 382	880	5 718	952	52 346
b	1 024	979	278	1 002	228	2 544	416	866	185	582	140	1 187	225	10 178
16) V_i' contains vertices with angle condition $\alpha_3 \geq 50°$:														
$\|V_i'\|$	180	209	67	228	62	496	106	176	46	113	34	275	54	2 200
$\|V_f'\|$	469	463	119	951	149	1 207	251	724	115	279	83	961	147	6 324
s	966	864	188	3 539	380	2 921	587	2 361	292	577	185	3 145	462	17 856
t	1 224	1 201	399	1 017	178	2 255	758	1 748	142	641	152	2 277	254	12 647
H	6 668	5 304	1 812	3 955	1 048	12 617	3 118	4 087	674	3 368	880	5 710	952	52 091
b	1 024	978	277	1 002	228	2 544	427	884	185	594	140	1 199	225	10 222
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.5: Computational results with V_i' based on α_3 being "not too small" as suggested in Section 4.2.3 and illustrated in Figure 4.13 on page 55

	PL	**CZ**	**SL**	**GB**	**S**	**D**	**A**	**I**	**NL**	**CH**	**DK**	**F**	**B**	**Europe**
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
17) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 300°$ or $\alpha_3 \geq 30°$:														
$\|V_i'\|$	351	437	122	487	93	898	260	422	84	206	64	647	85	4 432
$\|V_f'\|$	462	442	117	895	146	1 160	242	632	117	267	83	929	140	6 034
s	934	785	177	3 284	373	2 749	551	2 015	300	560	185	3 033	433	16 736
t	1 240	1 240	404	1 148	181	2 341	770	1 935	140	646	152	2 322	269	13 194
H	6 684	5 344	1 818	4 079	1 052	12 703	3 142	4 246	668	3 380	880	5 777	966	52 664
b	1 026	976	279	1 007	226	2 533	427	858	185	582	140	1 200	225	10 194
18) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 285°$ or $\alpha_3 \geq 40°$:														
$\|V_i'\|$	285	342	99	385	82	742	187	319	64	171	49	460	74	3 489
$\|V_f'\|$	460	447	117	898	149	1 159	230	634	115	256	83	901	140	5 996
s	940	812	177	3 304	380	2 744	524	2 027	292	523	185	2 915	433	16 632
t	1 240	1 227	404	1 138	178	2 346	797	1 931	142	669	152	2 401	269	13 302
H	6 678	5 330	1 818	4 069	1 048	12 703	3 142	4 238	674	3 394	880	5 816	966	52 660
b	1 016	973	278	1 005	228	2 533	409	856	185	576	140	1 177	225	10 123
19) V_i' contains the terminal vertices:														
$\|V_i'\|$	451	607	161	628	175	1 433	292	438	103	441	94	624	118	5 942
$\|V_f'\|$	455	423	112	892	143	1 155	227	574	120	250	89	799	138	5 787
s	930	727	165	3 262	369	2 742	523	1 817	314	513	203	2 570	430	15 949
t	1 244	1 266	414	1 155	**183**	2 336	790	2 048	138	**677**	146	2 580	272	13 648
H	6 684	5 376	1 820	4 094	**1 054**	12 715	3 150	4 331	656	**3 396**	868	5 982	966	52 997
b	1 012	971	270	1 011	222	2 533	407	827	179	567	138	1 143	221	10 029
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.6: Computational results with V_i' containing vertices fulfilling at least one of two angle conditions, or containing terminal vertices as suggested in Section 4.2.4 and illustrated in Figure 4.14 on page 56

	PL	CZ	SL	GB	S	D	A	I	NL	CH	DK	F	B	Europe
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
20) V_i' contains vertices with importance number larger than that of 60 % of all vertices:														
$\|V_i'\|$	917	831	270	1 122	235	2 564	560	1 197	172	591	174	1 718	221	11 142
$\|V_f'\|$	473	429	121	930	151	1 219	252	758	117	272	101	1 005	147	6 377
s	977	761	202	3 466	391	2 963	584	2 459	300	574	233	3 287	462	18 027
t	1 219	1 248	395	1 045	173	2 229	749	1 690	140	640	134	2 184	254	12 505
H	6 662	5 360	1 802	4 000	1 042	12 601	3 130	4 047	668	3 372	850	5 661	952	52 062
b	1 026	968	271	1 010	226	2 554	439	912	185	586	141	1 234	225	10 303
21) V_i' contains vertices with importance number larger than that of 70 % of all vertices:														
$\|V_i'\|$	745	591	188	812	220	2 016	392	828	139	439	124	1 263	122	8 423
$\|V_f'\|$	456	416	112	902	151	1 176	234	669	115	260	85	851	136	5 964
s	924	704	167	3 347	391	2 788	521	2 101	292	538	183	2 722	415	16 457
t	1 246	1 277	412	1 103	173	2 316	786	1 873	146	662	158	2 476	273	13 307
H	6 688	5 388	1 820	4 061	1 042	12 689	3 156	4 222	670	3 386	876	5 934	980	52 830
b	1 020	969	270	1 012	226	2 551	431	907	183	576	138	1 201	228	10 239
22) V_i' contains vertices with importance number larger than that of 80 % of all vertices:														
$\|V_i'\|$	521	432	129	573	174	1 280	239	513	90	221	79	819	71	5 657
$\|V_f'\|$	454	409	108	882	146	1 161	194	587	112	265	85	761	136	5 697
s	912	682	150	3 254	373	2 765	409	1 848	278	544	191	2 361	422	15 532
t	1 256	1 291	**419**	1 160	181	2 329	848	2 027	152	666	148	2 686	274	13 858
H	6 690	5 396	**1 830**	4 097	1 052	12 699	3 206	4 321	678	3 376	878	6 085	972	53 204
b	1 016	963	271	1 002	226	2 534	397	846	183	574	142	1 168	223	10 068
23) V_i' contains vertices with importance number larger than that of 90 % of all vertices:														
$\|V_i'\|$	197	126	65	475	95	489	102	227	64	84	65	393	52	2 824
$\|V_f'\|$	472	454	119	887	149	1 223	223	630	114	263	85	896	142	6 058
s	965	828	182	3 284	380	2 942	505	2 017	287	536	191	2 947	441	16 861
t	1 231	1 219	403	1 146	178	2 254	806	1 954	151	670	148	2 395	267	13 231
H	6 662	5 322	1 814	4 081	1 048	12 597	3 152	4 225	670	3 380	878	5 790	960	52 502
b	1 024	977	280	997	228	2 557	404	833	179	574	142	1 145	223	10 092
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.7: Computational results with V_i' based on importance numbers as suggested in Section 4.2.5 and illustrated in Figure 4.15 on page 57

	PL	CZ	SL	GB	S	D	A	I	NL	CH	DK	F	B	Europe
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
24) V_i' contains the vertices with degree larger than that of at least 60 % of their neighbors and vertices that have degree at least three in the Euclidean MSF:														
$\|V_i'\|$	439	472	160	528	111	1 098	282	548	85	286	91	696	107	5 177
$\|V_f'\|$	459	432	110	871	146	1 126	228	628	115	267	79	846	139	5 844
s	929	766	163	3 190	381	2 614	517	1 963	292	561	169	2 674	433	15 981
t	1 254	1 249	412	**1 189**	177	**2 402**	793	1 964	142	650	**160**	2 502	267	13 587
H	6 675	5 354	1 824	**4 132**	1 048	**12 777**	3 153	4 269	674	3 375	**888**	5 956	968	53 026
b	1 011	966	271	1 015	221	2 537	415	870	185	576	140	1 209	224	10 170
25) V_i' contains the vertices with degree larger than that of at least 60 % of their neighbors and vertices that have degree at least three in the MSF based on travel times:														
$\|V_i'\|$	437	457	160	530	114	1 109	275	545	84	286	91	670	102	5 151
$\|V_f'\|$	456	423	107	879	146	1 136	220	622	113	267	79	844	137	5 835
s	921	729	150	3 231	381	2 676	495	1 942	278	561	169	2 669	423	15 988
t	1 256	1 268	**419**	1 164	177	2 370	803	1 973	148	650	**160**	2 507	269	13 576
H	6 681	5 372	**1 830**	4 116	1 048	12 747	3 165	4 281	682	3 375	**888**	5 956	976	53 030
b	1 009	966	270	1 016	221	2 529	409	868	187	576	140	1 205	226	10 150
26) V_i' contains the vertices with degree larger than that of at least 60 % of their neighbors and vertices that have degree at least three in the union of the two MSFs:														
$\|V_i'\|$	457	504	174	600	121	1 138	298	578	86	296	98	739	118	5 528
$\|V_f'\|$	459	436	113	877	146	1 137	226	623	113	267	79	853	139	5 874
s	931	781	170	3 213	381	2 676	517	1 937	278	561	169	2 705	433	16 110
t	1 252	1 244	409	1 176	177	2 370	793	1 976	148	650	**160**	2 485	267	13 518
H	6 675	5 344	1 820	4 122	1 048	12 747	3 153	4 283	682	3 375	**888**	5 942	968	52 966
b	1 011	964	272	1 018	221	2 533	411	872	187	576	140	1 209	224	10 170
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.8: Computational results with V_i' containing vertices with relatively high vertex degree and branching vertices in MSFs

	PL	**CZ**	**SL**	**GB**	**S**	**D**	**A**	**I**	**NL**	**CH**	**DK**	**F**	**B**	**Europe**
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
27) V_i' contains terminal vertices and vertices that have degree at least three in the Euclidean MSF:														
$\|V_i'\|$	513	638	176	728	193	1 580	315	480	119	467	106	729	137	6 617
$\|V_f'\|$	454	430	107	882	143	1 140	226	577	117	250	86	797	136	5 754
s	923	759	150	3 231	369	2 675	513	1 808	302	513	187	2 557	422	15 790
t	1 251	1 252	**419**	1 171	**183**	2 367	790	2 048	140	**677**	154	2 570	274	13 695
H	6 684	5 358	**1 830**	4 109	**1 054**	12 751	3 160	4 340	666	**3 396**	876	6 005	972	53 109
b	1 010	966	270	1 007	222	2 534	417	843	181	567	140	1 156	223	10 067
28) V_i' contains terminal vertices and vertices that have degree at least three in the MSF based on travel times:														
$\|V_i'\|$	514	622	171	729	195	1 588	306	480	119	471	105	692	138	6 578
$\|V_f'\|$	449	423	107	886	143	1 140	228	553	115	250	87	773	134	5 697
s	906	727	150	3 255	369	2 677	529	1 736	288	513	197	2 455	412	15 595
t	**1 256**	1 266	**419**	1 156	**183**	2 365	784	**2 085**	146	**677**	146	2 635	**276**	13 793
H	**6 696**	5 376	**1 830**	4 100	**1 054**	12 751	3 150	**4 375**	674	**3 396**	874	6 042	**980**	53 206
b	1 008	971	270	1 005	222	2 531	409	829	183	567	140	1 148	225	10 037
29) V_i' contains terminal vertices and vertices that have degree at least three in the union of the two MSFs:														
$\|V_i'\|$	542	670	193	793	198	1 623	339	512	121	482	116	785	153	7 009
$\|V_f'\|$	458	434	113	891	143	1 145	226	570	115	251	84	811	136	5 785
s	935	774	170	3 269	369	2 696	513	1 768	288	515	181	2 587	422	15 863
t	1 246	1 247	409	1 153	**183**	2 356	790	2 066	146	675	154	2 554	274	13 653
H	6 677	5 348	1 820	4 089	**1 054**	12 741	3 160	4 362	674	3 396	882	5 991	972	53 078
b	1 013	964	272	1 006	222	2 536	417	847	183	567	142	1 168	223	10 093
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.9: Computational results with V_i' containing terminal vertices and vertices based on MSFs

	PL	**CZ**	**SL**	**GB**	**S**	**D**	**A**	**I**	**NL**	**CH**	**DK**	**F**	**B**	**Europe**
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
30) V_i' contains vertices with importance number larger than that of 80 % of all vertices and vertices that have degree at least three in the Euclidean MSF:														
$\|V_i'\|$	547	475	144	645	187	1 382	257	538	98	262	84	870	92	6 106
$\|V_f'\|$	458	413	107	872	146	1 141	197	595	112	257	82	782	136	5 695
s	928	713	150	3 215	373	2 666	416	1 857	278	530	175	2 445	422	15 511
t	1 252	1 278	**419**	1 183	181	2 376	843	2 019	152	670	160	2 627	274	13 855
H	6 678	5 378	**1 830**	4 113	1 052	12 751	3 204	4 320	678	3 386	882	6 060	972	53 228
b	1 014	952	269	999	226	2 537	401	860	183	570	140	1 182	223	10 081
31) V_i' contains vertices with importance number larger than that of 80 % of all vertices and vertices that have degree at least three in the MSF based on travel times:														
$\|V_i'\|$	550	461	139	643	190	1 387	250	531	95	268	83	841	93	6 064
$\|V_f'\|$	452	411	107	882	146	1 150	191	583	112	257	83	759	134	5 665
s	904	696	150	3 266	373	2 716	394	1 823	278	530	185	2 353	412	15 427
t	1 258	**1 285**	**419**	1 154	181	2 350	**853**	2 038	152	670	152	**2 688**	**276**	**13 896**
H	6 696	**5 388**	**1 830**	4 091	1 052	12 727	**3 216**	4 335	678	3 386	880	**6 091**	**980**	**53 271**
b	1 016	959	269	998	226	2 534	399	850	183	570	140	1 170	225	10 061
32) V_i' contains vertices with importance number larger than that of 80 % of all vertices and vertices that have degree at least three in the union of the two MSFs:														
$\|V_i'\|$	564	500	159	703	193	1 415	277	563	98	282	93	916	110	6 420
$\|V_f'\|$	458	421	113	885	146	1 153	198	595	112	258	80	799	136	5 758
s	921	742	170	3 260	373	2 729	419	1 844	278	532	169	2 512	422	15 723
t	1 254	1 267	409	1 162	181	2 343	840	2 024	152	668	**160**	2 590	274	13 741
H	6 683	5 360	1 820	4 089	1 052	12 721	3 204	4 328	678	3 386	**888**	6 030	972	53 130
b	1 017	950	271	999	226	2 537	401	864	183	570	142	1 186	223	10 099
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.10: Computational results with V_i' containing vertices with importance number larger than that of 80 % of all vertices and vertices based on MSFs

	PL	**CZ**	**SL**	**GB**	**S**	**D**	**A**	**I**	**NL**	**CH**	**DK**	**F**	**B**	**Europe**
$\|V\|$	3 382	2 719	919	2 537	587	6 505	1 667	2 383	375	1 756	476	3 304	534	28 280
$\|A\|$	8 858	7 369	2 399	8 511	1 606	17 793	4 463	8 196	1 108	4 586	1 217	11 132	1 668	82 594
33) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 270°$ or $\alpha_3 \geq 90°$, or that have importance number larger than that of 80 % of all vertices, or that are branching vertices in an MSF based on travel times:														
$\|V_i'\|$	557	480	141	665	191	1 397	264	539	96	275	85	862	98	6 184
$\|V_f'\|$	450	415	107	878	146	1 147	206	589	112	257	83	767	134	5 688
s	898	708	150	3 250	373	2 703	445	1 836	278	530	185	2 378	412	15 489
t	1 258	1 277	**419**	1 160	181	2 355	832	2 030	152	670	152	2 669	276	13 851
H	6 702	5 384	**1 830**	4 101	1 052	12 735	3 186	4 330	678	3 386	880	6 085	980	53 254
b	1 018	962	269	998	226	2 537	401	856	183	570	140	1 176	225	10 087
34) V_i' contains vertices with angle condition $\alpha_1 + \alpha_2 \leq 260°$ or $\alpha_3 \geq 110°$, or that have importance number larger than that of 90 % of all vertices, or that are branching vertices in an MSF based on travel times:														
$\|V_i'\|$	321	302	106	580	124	862	158	317	75	193	72	514	83	4 133
$\|V_f'\|$	452	434	109	880	146	1 155	211	567	107	260	83	825	138	5 764
s	916	756	154	3 269	376	2 714	471	1 787	259	540	185	2 654	423	15 848
t	1 256	1 251	415	1 151	180	2 350	818	2 071	**161**	664	152	2 543	271	13 701
H	6 686	5 362	1 830	4 091	1 050	12 729	3 174	4 338	**688**	3 382	880	5 935	974	53 045
b	1 008	978	273	995	224	2 546	401	819	181	572	140	1 146	227	10 033
%	88	88	91	57	75	82	85	66	71	86	80	69	72	77
%	**89**	**90**	**93**	**62**	**77**	**85**	**91**	**78**	**76**	**88**	**86**	**78**	**75**	**81**

Table 5.11: Computational results with V_i' consisting of vertices fulfilling angle conditions or having a high importance number or having vertex degree at least three in an (undirected) MSF based on travel times

5.2 National Time Table Graphs

We will now study the performance of the bundle recognition heuristic on time table graphs that are induced by trains traveling entirely within one country. We will consider these national time table graphs for the countries yielding the highest and lowest percentage of classified edges in Section 5.1 (Slovakia and Great Britain) and for the two countries contributing the most edges to the European time table graph (Germany and France). From the initial vertex sets considered in Section 5.1, we first use $V_i' = V$ (choice number 1), the initial vertex set that yielded the lowest percentage of classified edges for each of the four countries. We also include choice number 31, because it yielded the best result for the whole European time table graph. In order to include for each of the four countries the choice of initial vertex set that worked best for the country, we add choice number 24 to our considerations, having been best for Great Britain and Germany, while number 31 had yielded best results for Slovakia and France. In order to include another initial vertex set for comparison that is good but is constructed from different criteria than number 24 and 31, we use the terminal vertices (choice number 19). Table 5.12 lists the results obtained by the heuristic for each of the four initial vertex sets and each of the four countries.

First observe that for each of the four countries considered, the best and worst choice of initial vertex set did not change from Section 5.1. Also, the highest and lowest percentage of classified edges stayed the same or increased by one percentage point for Slovakia, Great Britain and France. But note that both percentages increased by three percentage points for Germany.

The countries differ from each other in that for Great Britain, the numbers of transitive and Hamilton path edges are only a few edges below the corresponding numbers of Section 5.1, for each of the four initial vertex sets under consideration, whereas for Germany, this is true for the numbers of Hamilton path edges but not for the numbers of transitive edges. France is similar to Germany in this respect — only for initial vertex set 31 does the number of Hamilton path edges not agree with the corresponding number in Section 5.1. For Slovakia, no such statement can be made at all. So for Great Britain, the extra edges in the European time table graph stemming from international trains more or less all end up unclassified, while for Germany and France, international edges also become transitive.

Figures 5.2, 5.3, and 5.4 show the time table graphs for Great Britain, Germany, and France for the initial vertex set that yielded the best result in Table 5.12. The final vertex set V_f' and the nontrivial bundles in the final partition are shown in black, and the remaining singletons are shown in grey. Observe that the low percentages of classified edges in Great Britain and France reported in Table 5.12 correspond to a "large amount of grey" in Figures 5.2 and 5.4. In Great Britain and France, there are many very long edges that stem from high speed and overnight trains that travel long distances without stopping anywhere

in-between. Particularly in Great Britain, the percentage of such edges appears to be high, while the time table graph shows comparatively few areas with the bundle structure that the heuristic is trying to exploit. This explains why the heuristic achieves such a low percentage of classified edges in Great Britain.

This lack of bundle structure in the British time table graph contrasts sharply with three of the four time table graph excerpts of Germany that are also studied: The four areas *sparsest*, *sparse*, *dense*, and *densest* have an increasing number of vertices per area, and Figure 5.5 shows their geographic location within Germany. Table 5.12 shows results of the bundle recognition heuristic broken down for these four areas, where vertices and edges are counted separately for each area. The four initial vertex sets considered yield different percentages of classified edges for some of the areas but perform identically for the *dense* area. The best and worst results achieved for each of the regions are illustrated in Figures 5.6 and 5.7, respectively. The high percentages of classified edges achieved for the *sparsest*, *sparse*, and *dense* areas correspond to a bundle structure of these areas that is clearly visible in Figure 5.6, whereas the *densest* area with its clusters of large cities displays much less of a bundle structure and also yields considerably lower percentages of classified edges.

Note that in the *sparse* area in Figure 5.6, only 4 pairs of directed unclassified edges are visible. The fifth one is very short and obscured by a vertex in V'.

	SL	**GB**	**D**	**F**	*sparsest*	*dense*	*densest*	*sparse*
$\|V\|$	882	2 525	6 473	3 291	165	425	321	119
$\|A\|$	2 269	8 485	16 892	10 897	404	1 005	906	317
1) Choose the initial vertex set $V_i' = V$:								
$\|V_f'\|$	112	951	1 209	1 003	29	76	88	22
s	180	3 559	2 446	3 170	34	78	243	39
t	357	997	1 928	2 117	53	**93**	97	62
H	1 732	3 929	12 518	5 610	317	**834**	566	216
b	253	1 001	2 518	1 225	61	190	133	34
19) V_i' contains the terminal vertices:								
$\|V_i'\|$	157	620	1 429	620	42	100	85	30
$\|V_f'\|$	96	881	1 095	767	22	75	75	12
s	132	3 240	2 090	2 349	18	78	191	10
t	378	1 155	2 090	2 557	**61**	**93**	121	**75**
H	1 759	4 090	12 712	5 991	**325**	**834**	594	**232**
b	247	1 009	2 478	1 133	57	188	135	30
24) V_i' contains the vertices with degree larger than that of 60 % of their neighbors and vertices that have degree at least three in the Euclidean MSF:								
$\|V_i'\|$	153	525	1 104	694	31	70	71	24
$\|V_f'\|$	96	860	1 069	817	22	75	68	16
s	139	3 168	1 975	2 471	18	78	153	18
t	374	**1 189**	**2 151**	2 471	**61**	**93**	**141**	71
H	1 756	**4 128**	**12 766**	5 955	**325**	**834**	**612**	228
b	245	1 013	2 480	1 197	57	188	135	34
31) V_i' contains vertices with importance number larger than that of 80 % of all vertices and vertices that have degree at least three in the MSF based on travel times:								
$\|V_i'\|$	134	635	1 384	840	50	98	80	20
$\|V_f'\|$	92	871	1 084	743	22	75	70	12
s	123	3 244	2 032	2 215	18	78	163	10
t	**381**	1 154	2 126	**2 622**	**61**	**93**	137	**75**
H	**1 765**	4 087	12 734	**6 060**	**325**	**834**	606	**232**
b	245	996	2 471	1 153	57	187	133	30
%	92	58	85	70	91	92	73	87
%	**94**	**62**	**88**	**79**	**95**	**92**	**83**	**96**

Table 5.12: Computational results for input data restricted to trains traveling entirely within one of four selected countries. Within Germany, the results are additionally broken down for four geographical areas illustrated in Figure 5.5.

Figure 5.2: The time table graph of Great Britain showing the best result reported in Table 5.12

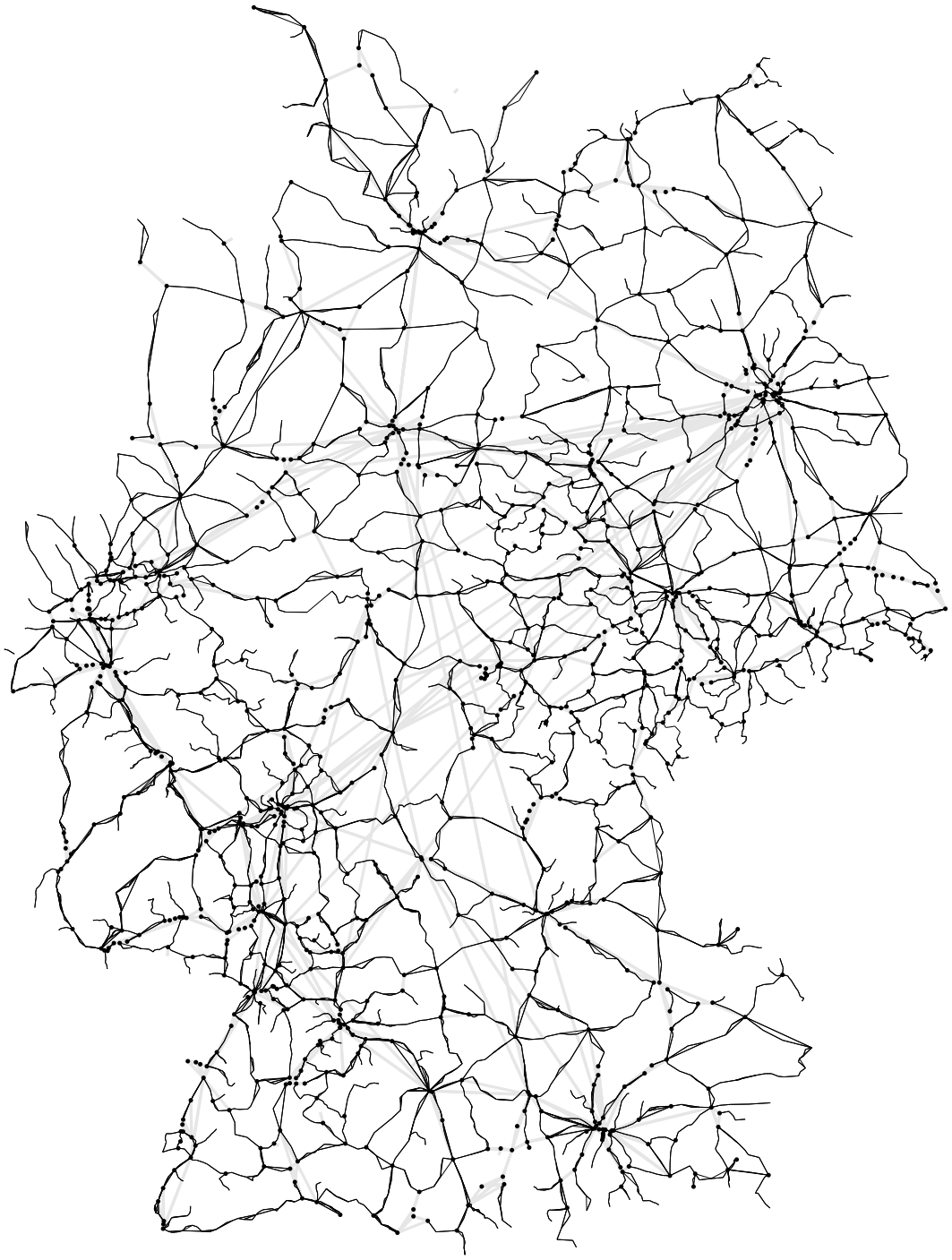

Figure 5.3: The time table graph of Germany showing the best result reported in Table 5.12

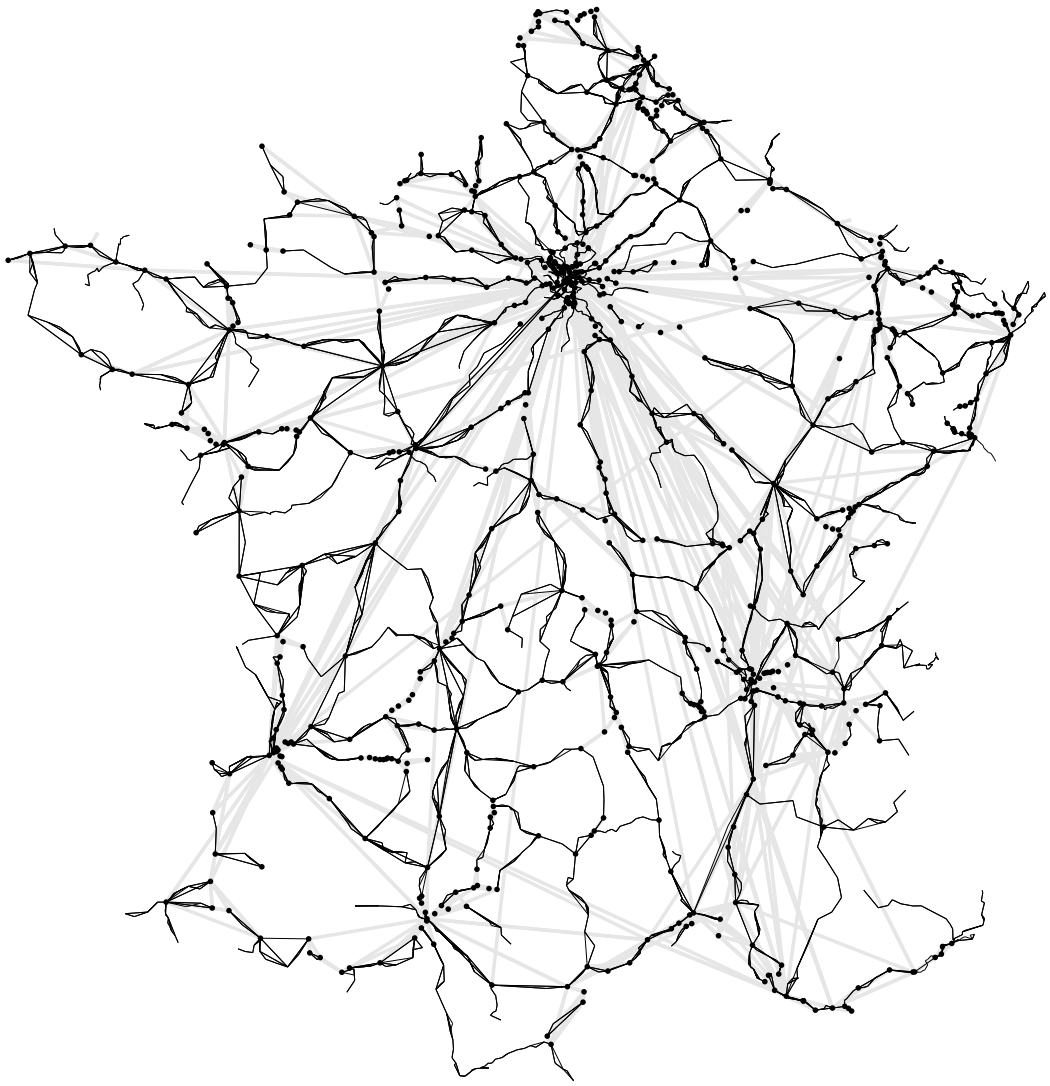

Figure 5.4: The time table graph of France (without Corsica) showing the best result reported in Table 5.12

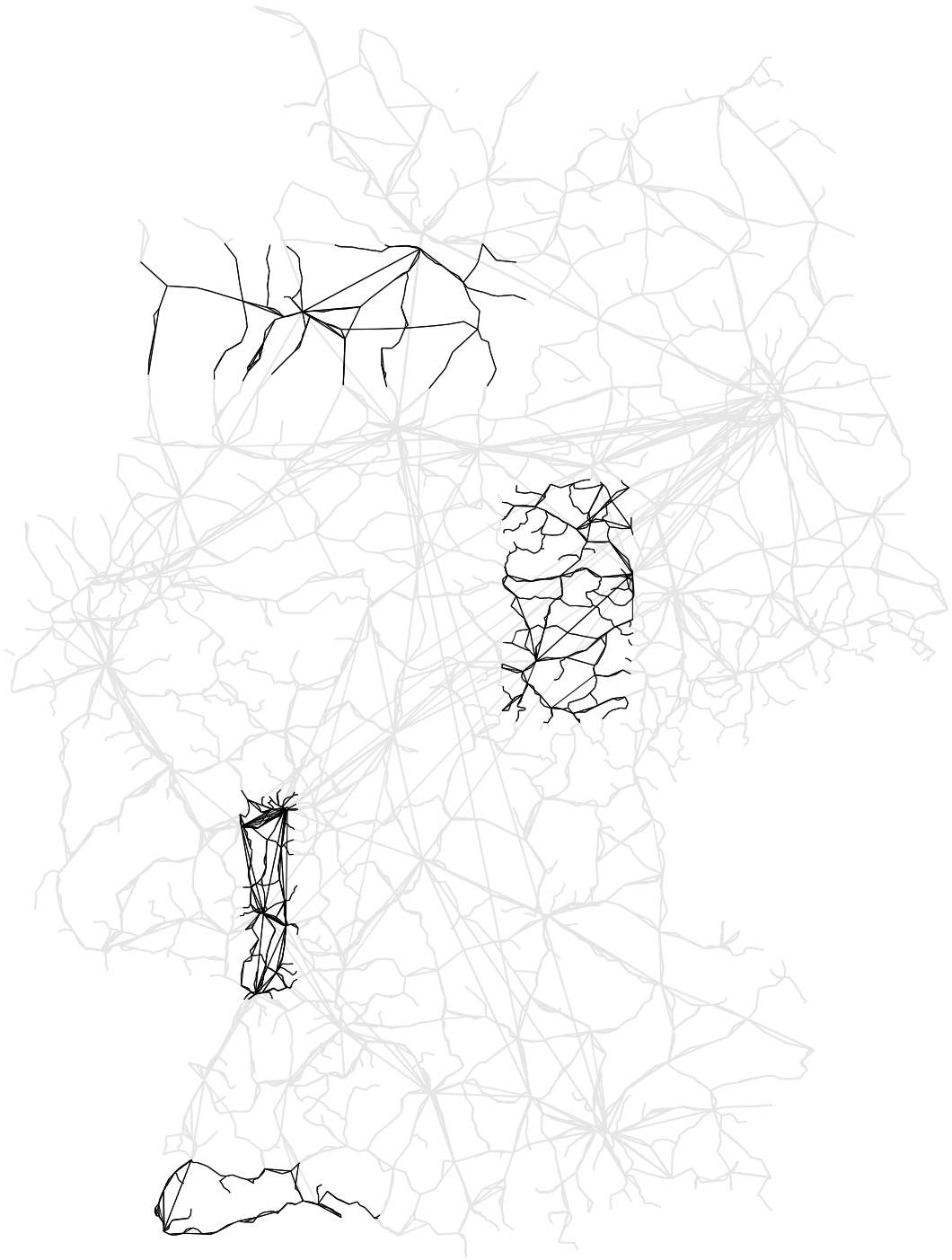

Figure 5.5: The time table graph of Germany with the four regions *sparsest*, *dense*, *densest*, and *sparse* (from north to south)

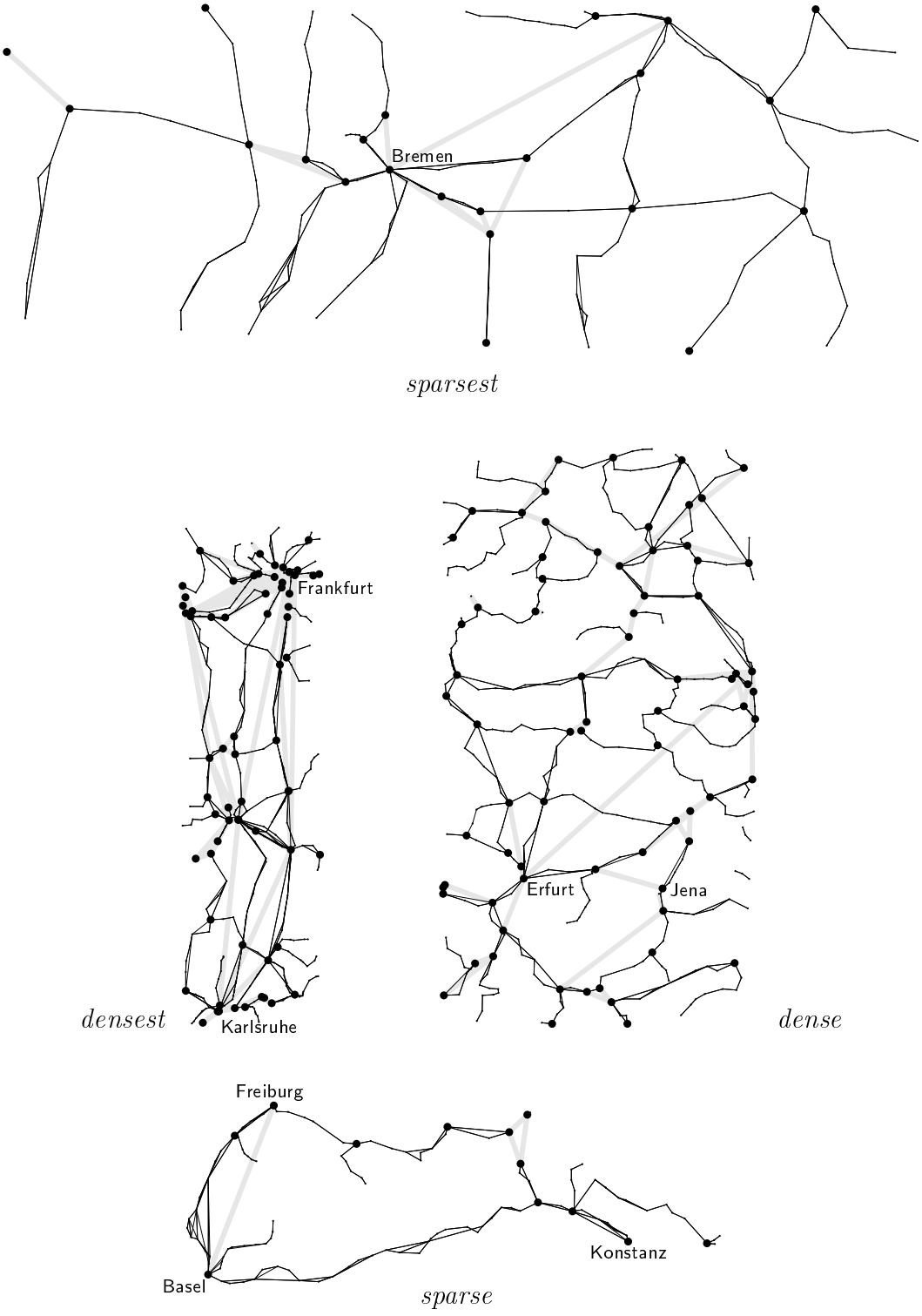

Figure 5.6: Excerpts of the German time table graph showing the best results reported in Table 5.12. The four excerpts are drawn to scale.

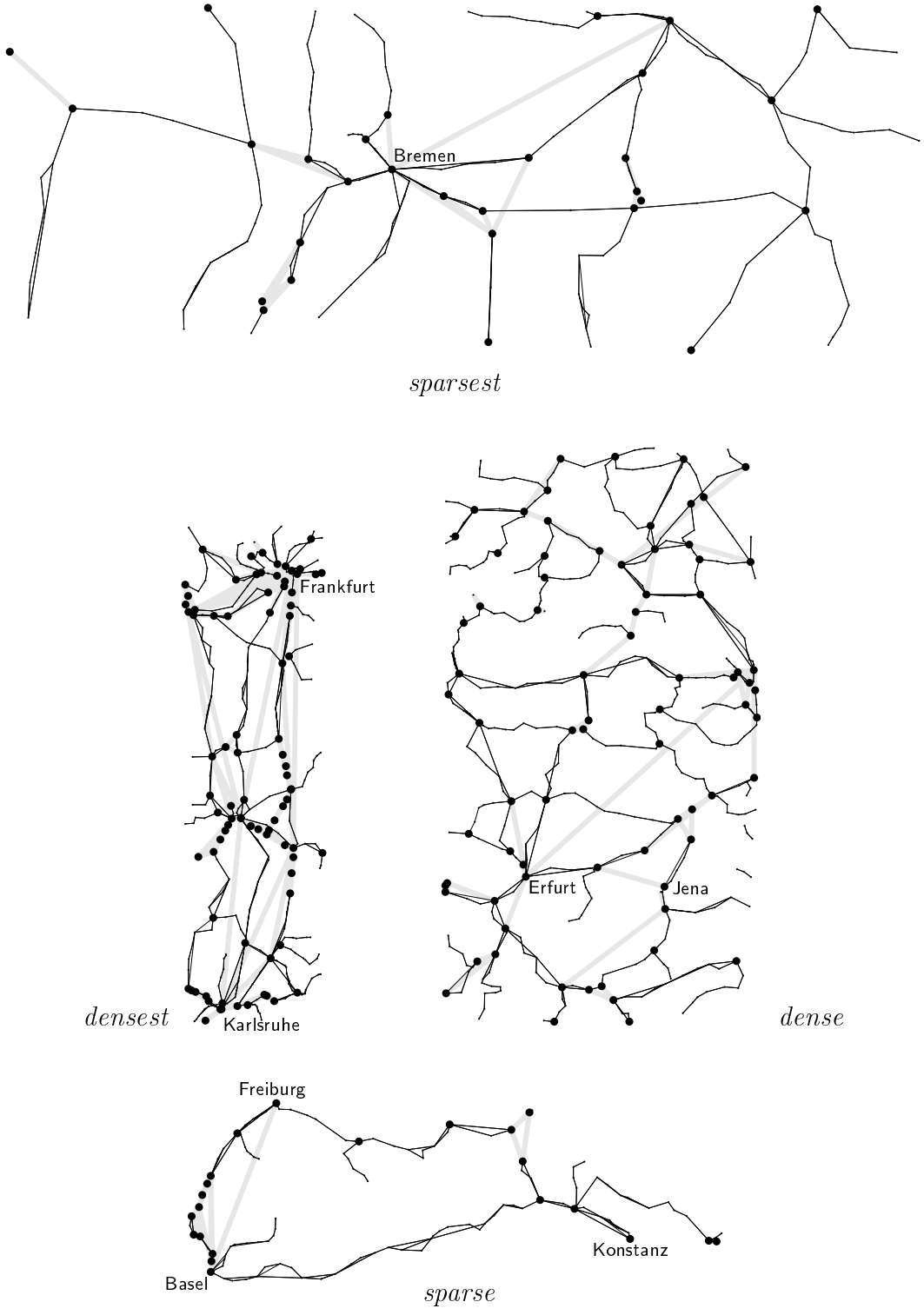

Figure 5.7: Excerpts of the German time table graph showing the worst results reported in Table 5.12. The four excerpts are drawn to scale.

Chapter 6

Conclusion

Considering an application problem from train time table graph analysis that entails classifying the edges of a time table graph as either belonging to the physical railroad network or as being transitive, we have investigated a structural approach. It is based on the apparent *bundle structure* of time table graphs, and the result of this modeling step is the *bundle recognition problem*.

Unfortunately, the corresponding graph theoretic optimization problem (and some variants of it) have turned out to be NP-hard. Therefore we have developed a heuristic and have evaluated it, and more generally the structural approach itself, by a computational study on European train time table data.

The results of the computational study show that our structural approach and the heuristic to implement it are able to solve the given application problem to a large degree, so that most of the time table graph edges are classified. The success of the heuristic actually depends on how much the assumed bundle structure is actually present in a time table graph. This could be observed by comparing the results for 13 European countries with each other.

Since already the modeling step had shown that we cannot expect to classify all time table graph edges using our approach, the obtained results actually fulfill our expectations, and we have shown our structural approach to be useful for the given application problem.

It remains open whether a further variant of the bundle recognition problem can be found that admits a polynomial time solution for the corresponding optimization problem.

List of Figures

List of Tables

Bibliography

[BM76] John A. Bondy and Uppaluri R.S. Murty. *Graph theory with applications*. North–Holland, 1976.

[BW00] Ulrik Brandes and Dorothea Wagner. Using Graph Layout to Visualize Train Connection Data. *J. of Graph Algorithms and Applications*, 4(3):135–155, 2000.

[BWZ97] M.R. Bussieck, T. Winter, and U.T. Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming*, 79:415–444, 1997.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw–Hill, 1990.

[Coo71] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, STOC'71*, pages 151–158, 1971.

[Doc99] Universität Konstanz, Lehrstuhl für Praktische Informatik I. *Creation and Analysis of Train Graphs: Software Documentation*, 1996 – 1999. 57 pages.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$–Completeness*. W H Freeman & Co Ltd, 1979.

[GKW98] Dieter Gluche, Dietmar Kühl, and Karsten Weihe. Iterators Evaluate Table Queries. *ACM SIGPLAN Notices*, 33(1):22–29, 1998.

[GM99] Michael T. Goodrich and C.C. McGeoch, editors. *International Workshop on Algorithm Engineering and Experimentation, ALENEX'99*. Springer-Verlag, Lecture Notes in Computer Science, vol. 1619, 1999.

[Kin95] Jeffrey H. Kingston. Collection of Computer Science Bibliographies: Bibliography on Practice and Theory of Automated Timetabling. http://liinwww.ira.uka.de/bibliography/Misc/timetabling.html, 1995.

[KW96] Dietmar Kühl and Karsten Weihe. Iterators and Handles for Nodes and Edges in Graphs. Konstanzer Schriften in Mathematik und Informatik 15, Universität Konstanz, 1996. http://www.inf.uni-konstanz.de/Preprints/preprints.html#015.

[KW97] Dietmar Kühl and Karsten Weihe. Data Access Templates. *C++ Report*, 9(7):15–21, 1997.

[LW] Annegret Liebers and Karsten Weihe. Recognizing Bundles in Time Table Graphs — A Structural Approach. In *Proceedings of the 4th Workshop on Algorithm Engineering, WAE 2000*. Springer-Verlag, Lecture Notes in Computer Science, to appear.

[LWW99] Annegret Liebers, Dorothea Wagner, and Karsten Weihe. On the Hardness of Recognizing Bundles in Time Table Graphs. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, *Proceedings 25th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'99*, pages 325–337. Springer-Verlag, Lecture Notes in Computer Science, vol. 1665, 1999.

[LWW00] Annegret Liebers, Dorothea Wagner, and Karsten Weihe. On the Hardness of Recognizing Bundles in Time Table Graphs. *International Journal of Foundations of Computer Science*, 11(3):467–484, 2000.

[MN99] Kurt Mehlhorn and Stefan Näher. *LEDA: A Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[Mor] Bernard M.E. Moret, editor. *2nd Workshop on Algorithm Engineering and Experiments, ALENEX 2000*.

[MS95] David R. Musser and Atul Saini. *STL Tutorial and Reference Guide*. Addison–Wesley Publishing Company, 1995.

[PTS] University of Kaiserslautern, Department of Mathematics. Bibliography in Public Transportation Systems (PTS). http://mispaix1.mathematik.uni-kl.de/lit-db/.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, STOC'78*, pages 216–226, 1978.

[Str97] Bjarne Stroustrup. *The C++ Programming Language*. Addison–Wesley Publishing Company, 3rd edition, 1997.

[SWW99] Frank Schulz, Dorothea Wagner, and Karsten Weihe. Dijkstra's Algorithm On–Line: An Empirical Case Study from Public Railroad Transport. In Jeffrey Scott Vitter and Christos D. Zaroliagis, editors, *Proceedings of the 3rd Workshop on Algorithm Engineering, WAE' 99*, pages 110–123. Springer-Verlag, Lecture Notes in Computer Science, vol. 1668, 1999.

[VZ99] Jeffrey Scott Vitter and Christos D. Zaroliagis, editors. *Proceedings of the 3rd Workshop on Algorithm Engineering, WAE' 99*. Springer-Verlag, Lecture Notes in Computer Science, vol. 1668, 1999.

[WBL+99] Karsten Weihe, Ulrik Brandes, Annegret Liebers, Matthias Müller-Hannemann, Dorothea Wagner, and Thomas Willhalm. Empirical Design of Geometric Algorithms. In Victor Milenkovic, editor, *Proceedings of the 15th ACM Symposium on Computational Geometry, SCG'99, Miami, Florida, USA, June 13–16, 1999*, pages 86–94, 1999.

[Wei97] Karsten Weihe. Reuse of Algorithms: Still a Challenge to Object-Oriented Programming. In *Proceedings of the 12th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA'97*, pages 34–48, 1997.

[Wei98] Karsten Weihe. Covering trains by stations or the power of data reduction. In *On-Line Proceedings of the 1st Workshop on Algorithms and Experiments, ALEX'98*, 1998. http://rtm.science.unitn.it/alex98/proceedings.html.

Index