Efficient survivable reconfiguration in SDH networks

Diplomarbeit bei
Prof. Dr. M. Grötschel

vorgelegt von
Thomas Thevis

Fachbereich Mathematik der TU Berlin,
Studiengang Wirtschaftsmathematik

Berlin, 17. Juli 2005

Zweitgutachter: Prof. Dr. R. Möhring
Betreuer: Dr. R. Wessäly

Abstract

This thesis deals with reconfiguration planning in telecommunication networks. Usually, telecommunication demands are routed on pre-configured paths through the network. However, the assignment of routing paths needs to be changed at times. For practical purposes, the extent of the applied changes does not have to be too complex. In this thesis, we present methods to reconfigure networks efficiently in this sense.

We develop mixed-integer programming models which abstract from specific reconfiguration tasks, such that the mathematical model and the presented solution methods can be applied to further reconfiguration tasks not explicitly discussed in this thesis. Based on the theory of Combinatorial Optimization, a branch-and-price framework is developed. The pricing problems for various reconfiguration tasks are examined in detail.

We have implemented the proposed branch-and-price framework and have tested the solution approach on different real world telecommunication networks for several reconfiguration tasks. The results of these tests are discussed in detail. It is of major practical interest to obtain small dimensioned reconfiguration results. Nevertheless, to our best knowledge, there are no solution approaches published in the existing literature covering a comparably large spectrum of different reconfiguration tasks.

The proposed methods support the reconfiguration of survivable networks, such that survivability restrictions of initial routings are respected in the reconfiguration process. Furthermore, it is possible to introduce survivability conditions to networks without initial protection mechanisms. Although the focus of this thesis is survivable reconfiguration in SDH networks, both the mathematical model and the developed solution methods support the integration of multi-layer aspects into the reconfiguration planning process.

Acknowledgements

My sincere thanks go to the people who provided me their help and support during the last months. I would like to thank many helpful members of the Optimization Department of ZIB and especially my advisor, Roland Wessäly, for the time he spent on discussing all kinds of mathematical and implementational questions with me. Furthermore, special thanks go to Sebastian Orlowski for offering and providing me his advice countless times and for proofreading drafts of this thesis. I do honestly appreciate the the personal commitment of So-Young Lee, Annika Poerschke, and Nader Razouk.

Contents

Chapter 1

Introduction

1.1 Outline of this thesis

In this section, we give an overview on the structure of this thesis. This chapter provides a brief introduction to modern telecommunication networks. We focus on network layout, dimensioning, and on network planning in general. Additionally, a simplified technical description of SDH technology and a characterization of protection mechanisms to ensure survivability is given. In Chapter 2, practically interesting reconfiguration tasks are presented in more detail. We have developed a very general mathematical model for the different reconfiguration tasks which is presented in Chapter 3. For the solution of the reconfiguration problems, we have developed a branch-and-price framework. The theoretical aspects of this algorithm are presented in detail in Chapter 4. To test the practical applicability of the branch-and-price approach, we have implemented the proposed algorithm. Implementational details are discussed in Chapter 5. The implementation of the algorithm has been tested on different telecommunication networks for several different reconfiguration tasks. In Chapter 6, we present the results of these tests and a detailed discussion of these results. Finally, conclusions are drawn in Chapter 7.

1.2 Telecommunication networks

In this section, we describe the basic functionality of modern telecommunication networks. Starting from different requirements, we provide some technical background information helping to understand the working methods of a telecommunication network. Since this thesis deals with the reconfiguration of SDH networks, SDH technology is described in more detail. Afterwards, we focus on security mechanisms protecting networks against an outage of provided services and ensure further data transmission even in the case of component failures, and discuss some difficulties which may arise in planning multi-layer networks.

1.2.1 Requirements

Modern telecommunication networks are a transport medium for many different services. Starting from voice telephony, over Internet access, television to video telephony, and so on. These services are provided based on the same infrastructure. They use the same physical equipment despite their different requirements w.r.t. data size and transmission time. Voice telephony for instance requires only a small bandwidth for data transmission, but a connection has to be continuously established over a certain time. On the other hand, there are services like facsimile or e-mail that require a certain amount of data transmission, but allow for a transportation of single data packages without the need for a continuous connection. Other services need both a huge bandwidth and a continuous connection. High quality video telephony is an example for such a service.

Another distinctive feature of telecommunication services is the balance of data transmission. Thinking of data transmission between an emitter and a receiver, there are services like telephony, in which both communication partners are emitter and receiver at the same time. The emitted amount of data is roughly the same. Other services, like database requests, are typically asynchronous in that a small data emission on one connection side leads to a relatively large transmission on the other side. All these different services have to be provided via the same infrastructure, i.e., the same physical connections and furthermore, they have to be provided concurrently.

1.2.2 Multiplexing

With multiplexing technologies, it is possible to transmit multiple independent data streams over a single physical connection concurrently. There are different multiplexing techniques like wavelength, code, or time multiplexing. The task is always the same: a number of small data streams have to be converted into a single data stream, and after transmitting the packed stream to another location, it has to be decomposed without loss of information. An important fact is that the multiplexing procedure may also be applied to already multiplexed data streams. That means, it is possible to embed small, low-level data streams into one multiplexed data stream and afterwards combine a number of these higher-level data streams to another one and so on. Due to the multiplexing technique, a telecommunication network is designed hierarchically. A more detailed description for the multiplexing procedure used in SDH networks is provided in Section 1.2.6.

1.2.3 Network

The network itself is composed of *locations* (or *nodes*) and *connections* (or *links*) between the nodes. A link may be an optical or an electrical fiber as well as a radio or any other possible connection to transmit signals between different locations. At locations, there are hardware installments to transmit and receive data streams and/or to forward signals. Some locations are equipped with hardware installments to multiplex

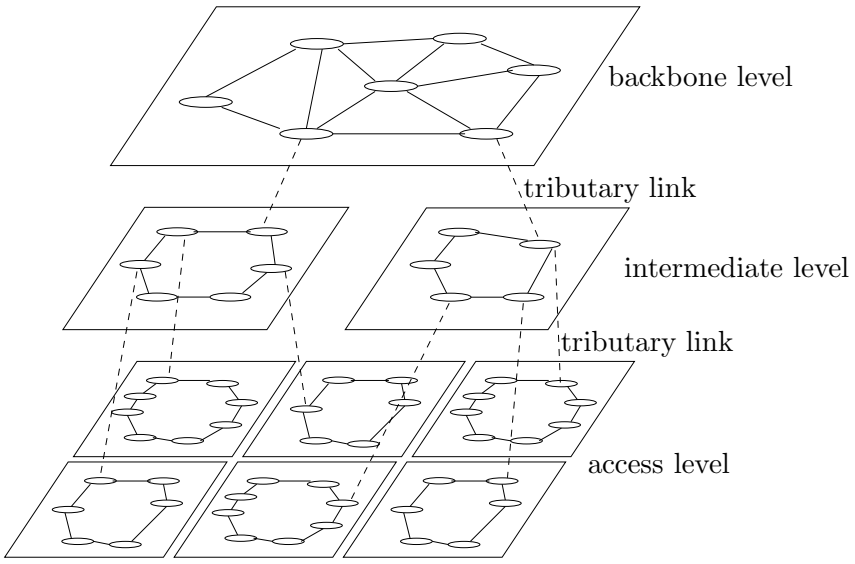

Figure 1.1: A typical multi-level network layout. On the top level a highly intermeshed backbone layer and below ring structured intermediate and access network levels. The connection between different hierarchies is provided by tributary links. The capacity of connections and the distances between locations decreases continously from backbone network level to access level.

and demultiplex data streams.

A typical network has a multi-level network structure. Figure 1.1 displays an exemplary network configuration. The lowest level subnetwork is called *access network*. The customers of the service providing company are usually connected to nodes in the access network. The access network is designed to cover only a small geographical area and allows a connection either between locations of this access network or between a single location of this subnetwork with the next level network to establish a connection to a location in another access network. Some location has to provide multiplexing capacity and therefore serves as a connection to the next level network. The highest-level network is called the *backbone network*. Typically, it consists of a highly intermeshed structure and provides large capacities on its links to transmit huge amounts of data over long distances. Its major task is to transmit data between the remote subnetworks which are connected to the nodes of the backbone network. Between the access layer and the backbone network there are several *intermediate layers*, which usually have a ring structure like the access network, but cover a larger area. The main reason for the ring layout structure of the lower layers is to provide at least basic protection mechanisms against the failure of connections or locations in the network (see Section 1.2.7). Inter-layer connections are called *tributary links*.

Remark 1.1. *This short sketch of a multi-level network layout should only serve as a basis for a general understanding of typical topologies and functionalities of mod-*

ern telecommunication networks. This topological multi-level structure is completely different from the technical multi-layer structure which will be described in detail in Section 1.2.6.

1.2.4 Communication demands and routing

Each service requested by a customer consumes hardware capacities for a certain time. Between nodes in the network, a virtual connection has to be established for data transmission. The virtual connection is realized by a physical path which consists of a sequence of physical network connections. The establishment of a virtual connection and the following data transmission between network locations consumes parts of the limited transmission capacity on the links which are contained in the path. The amount of required link capacity for data transmission is called the *(communication) demand*. The assignment of physical links that provide a connection between the locations of a certain demand are called *routing*. The sequence of physical links forming a virtual connection is called *routing path*.

1.2.5 Network planning

The planning process of a telecommunication network contains many different single planning steps. The most basic decision for an operating company is which services should be provided to customers and what transmission techniques should be applied. Furthermore, there are a lot of additional planning decisions. The following list of planning steps is ordered from more strategic decisions to more operational ones. Strategic decisions are long-term decisions, i.e., they are made once and are rarely changed, whereas operational ones are medium-term and short-term decisions, respectively. Usually, they last for a couple of weeks up to several months.

- **Topology**
  Topology planning describes the choice of locations and connections between the locations.Which geographical locations are suitable to set up network nodes? Which locations have to be directly linked with each other and what techniques can be applied for such a connection (bury a fiber, establish a radio connection, or lease capacities from other operating companies)? How many different layers should be set up? How to assign nodes and links to different subnetworks?

- **Dimensioning**
  Dimensioning can be seen as a link between topology planning and the more operational determination of demand routings. The capacity of links must be specified as well as the hardware installments at locations. To connect links to a location, switching devices have to be used. Special switching devices are *multiplexers*, which allow packing data streams, as described in Section 1.2.2 or more precisely for the SDH technology in Section 1.2.6.

- **Demand forecast and routing**
  Derived from the choice of provided services, there has to be an estimation of

demands between the locations. Starting from such a *demand forecast*, a demand routing is planned. For each demand, routing paths have to be found and to be integrated into the network. For multi-layer networks with many locations and links, this would lead to a huge amount of planning data. Instead of planning the routing for a single demand throughout the complete network over multiple layers, it is often possible to route a set of demands in common through the network. Demands with both end nodes equal can be routed together. If different layers are planned separately, the aggregation of low level demands can be seen as one induced single demand for the next level layer.

In the setup process of a new network, these planning steps should be performed as an integrated planning procedure, because of their close relationship. Decisions on any planning level have an impact on decision possibilities on other planning levels. Strictly speaking, a demand traffic forecast can only be done after the determination of locations. However, an appropriate estimation of traffic beforehand could ease the dimensioning planning.

However, a complete planning process, respecting all stages of the planning process, is rarely done. Often, it is necessary to adapt a given network to new circumstances. The decision to provide additional services or a changing of the amount of demands in the network might lead to a need for a new planning process. In such a case, the planning procedure is almost reduced to the demand planning steps. It may be possible to add locations or new links to the network or adopt the dimensioning of routing or switching capacities, but the main network layout will usually not be changed. In practice, network planners are mostly faced with the operational reconfiguration planning. Strategic decisions have to be implemented or the operational routing has to be adopted to modified demand forecasts.

Because of the practical importance of the reconfiguration planning, this thesis deals only with the operational planning steps of demand planning. Network configuration, dimensioning and demand forecast are assumed to be already determined beforehand. For all different reconfiguration scenarios, described in detail in Chapter 2, the general planning task is essentially the same: Possibilities for a reconfiguration of the demand routing have to be found which do not imply changes to the hardware configuration.

1.2.6 SDH networks

The abbreviation SDH denotes the *Synchronized Digital Hierarchy* technology. SDH is a world-wide standard data transmission system which replaces the old *Plesiochronous Digital Hierarchy* (PDH) system. The main advantage of SDH in contrast to PDH is the more transparent multiplexing procedure. It is possible to decouple a low-level data signal from the highest aggregated data stream. This is in contrast to PDH networks, where the complete multiplexing hierarchy has to be passed through in order to demultiplex a low-level data stream from a high-level one. Thus, the SDH system

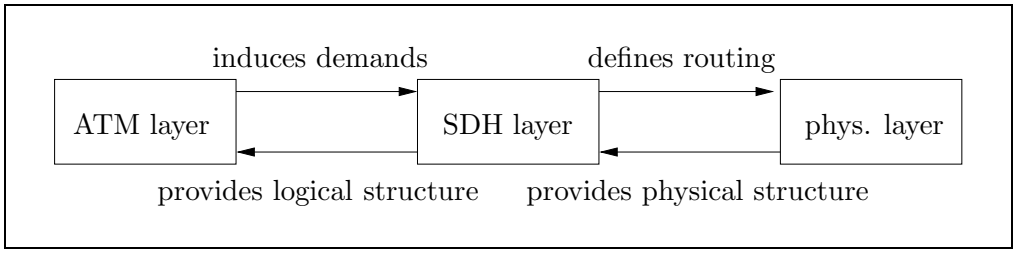

Figure 1.2: The role of the SDH layer as an interface between a higher level network layer (in this case an ATM layer) and the physical network layer.

is more flexible and cheaper in maintenance, since the number of multiplexing devices at network locations can be reduced.

SDH serves as transporting system for telecommunication services and for other transmission technologies. The *Broadband Integrated Services Digital Network* (B-ISDN) for instance specifies the *Asynchronous Transportation Module* technique (ATM) as transmission service. ATM itself specifies interfaces to different transmission layers. One of them is the SDH technology. Strongly simplified, in this thesis we deal with the following network hierarchy: From a superordinate layer (ATM for example), demand specifications are given. These have to be routed in an underlying physical network. The SDH layer is therefore another interface between the communication demands and the electrical or optical fiber network layer. Its main task is to translate the demands from the higher layer into a feasible routing in the physical network (see Figure 1.2). Note that multi-layer aspects in the planning process refer to this hierarchy of data transmission technologies and not to the multi-level network topology.

Simplified technical structure

Most information of this section is taken from [Kya93] and [Sie93]. As mentioned before, modern telecommunication networks provide the infrastructure for various services with different demands of data size transmission. The conversion of different demands into a unified transmission structure leads obviously to a trade-off between the different requirements. Each of the following frame and container size specifications must be seen as such a trade-off between physical conditions and various technical requirements.

The capacity of a fiber is defined as the amount of data that is passed through within a given period of time. In SDH networks, as in many other networks the basic time period is 125 μs. This is due to the conversion of analog voice signals into digital data signals, where the analog channel is scanned 8000 times per second and the signal is converted into a byte pattern.

SDH technology uses unified data frames, the so called *Synchronous Transport Modules* (STM). The basic transport module is called STM-1 module. It consists of a 2430 byte frame, which is transmitted with a bit rate of 155.52 Mbit/s. The transmission

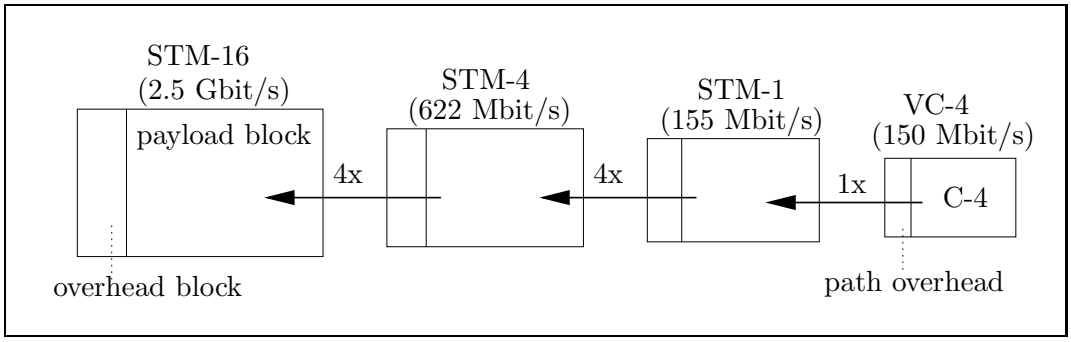

Figure 1.3: Embedding the largest virtual container VC-4 into the STM hierarchy.

Container	Capacity (Mbit/s)
C-4	149.760
C-32	48.384
C-31	36.864
C-22	9.088
C-21	6.784
C-12	2.176
C-11	1.600

Table 1.1: Container size specifications for SDH networks.

duration is 125 μs, as mentioned above. Combining STM-1 frames leads to a higher bandwidth. Four STM-1 frames can be combined to a single STM-4 module with a bit rate of approximately 622 Mbit/s and four STM-4 modules give a single STM-16 module with a bit rate of roughly 2.5 Gbit/s. The STM modules consist of an overhead block and a payload block. The payload block contains the services and systems to be transported through the network,whereas the overhead block contains meta information about the content and the structure of the payload block.

The STM-1 frame payload block has a size of 150.34 Mbit/s. The payload block of an STM frame is filled with *virtual containers* of different sizes. The size of a container depends on the multiplexing element which combines different data streams into a single container. By adding the so called *path overhead* to the container, the container becomes a *virtual container*. The path overhead stores meta information on the content of the given container and serves as control mechanism for transmission quality. Additional elements in the specifications allow for compensation of a physical displacement of phase and further mechanisms to ensure quality of service. The specified container sizes can be seen at Table 1.1.

The VC-4 is the largest virtual container. An STM-1 frame may contain exactly one VC-4 or three VC-32 and VC-31 respectively, and so on. Usually, the smallest virtual

container that is used in European SDH networks is the VC-12. Its capacity suffices to transmit 32 64-kbit/s-streams (voice telephony, for instance, is transmitted by 64 kbit/s-streams).

In this thesis, the VC-12 is the basic unit for demands in the SDH network. Whenever the superior layer induces a demand of value k between two nodes n_1 and n_2 in the SDH network, the task is to find a routing of k VC-12 between n_1 and n_2.

1.2.7 Protection mechanisms

Protection mechanisms are implemented to reduce the harm caused by failure of network components. Component failure may have various reasons. The malfunction of a single switching device, the physical destruction of a fiber, or the breakdown of a complete location could lead to a temporarily outage of the network. The routing requirements of certain demands could not be fulfilled and data may get lost.

To describe the failure of components, we use the concept of *operating states*. The term *normal operating state* (NOS) denotes the situation in which each single network component is operational. Each other operating state is characterized by a set of network components that are not operational. In Chapter 3, we give a formal definition of operating states to be able to include them into the mathematical model. Furthermore, we distinguish between *single failures* and *multi failures*. Although the malfunction of a single component seems to be similar to the outage of a set of components, the mathematical problems become much more complex and difficult to solve. However, we show in Section 1.2.8 that multi failures must not be neglected in the planning process, especially when including multi layer aspects into the planning procedure.

A network which is topologically designed and dimensioned to avoid the loss of data caused by component failures or to minimize the downtime of the network in failure cases is called a *survivable network*. If demands are affected by component failures, their originally allocated routing paths are temporarily not available. Replacement paths have to be found. A main distinctive feature between different protection strategies is the way of finding replacement paths for affected demands. *Restoration techniques* determine these paths in failure case at runtime. They can further be categorized w.r.t. the complexity of the routing reconfiguration (complete end-to-end re-routing of affected demands vs. local replacements of sub-paths to avoid failing components). The main disadvantage of restoration techniques is the large expenditure of time which is required to compute a reconfiguration. For a detailed description of restoration techniques, see [Orl03] for example. Other protection strategies make use of *dedicated backup paths*. Potential replacement paths are already assigned in the planning process and not just at the moment of a component breakdown.

One concept that is used in practice is *1:1 protection*. Routing paths can be protected by preassigned backup paths. Each protected routing path disposes of a private dedicated replacement path. Possibly, it may be used by low-priority data traffic in the normal operating state. The disadvantage of this protection mechanism is the huge

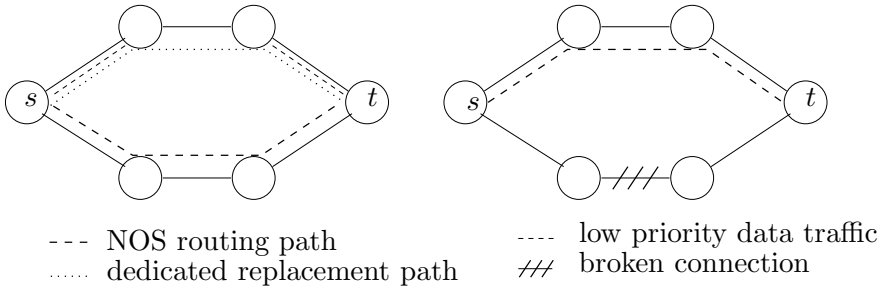

Figure 1.4: Sketch of the 1:1 protection mechanism. A dedicated backup path can be used in normal operating state to route low priority data traffic. In case of component malfunction, it is used to route the protected demand between s and t. However, in the time that is used for path switching, data may get lost.

amount of reserve capacity that has to be provided. A tradeoff between routing protection and the need for reserve capacity is the *1:N protection* which can be seen as a generalization of the 1:1 protection. Since the concurrent malfunction of different components becomes more unlikely the more components are affected, in most cases it should suffice to provide a single backup path for a set of routing paths. N routing paths share one replacement path and as before, low-priority data traffic may be transmitted over the backup path in the case of normal operating state.

The next step of generalization of this protection idea is the implementation of *M:N protection* where M backup paths are reserved for N routing paths. If the networks capacities are appropriately dimensioned, there will be no enduring loss of data, and the outage time is reduced to the time required by the switching devices which have to realize the change of routing paths.

A slightly different concept is used by *1+1 protection*. The demand signal is duplicated and routed along disjoint paths (w.r.t. potential failure components) to the target location which has to provide hardware installments to receive both signals and to choose the better one. The advantage of this protection method is that no additional network downtime occurs if one of the two signaling paths fails, since the other one remains operational all the time. The obvious disadvantage is, as in the case of 1:1 protection, the large amount of additional capacities. However, in contrast to 1:1 protection, where reserve capacities can be used by low-priority data traffic in the normal operating state, dedicated backup capacities are completely occupied by the routing of the protected demands. Figures 1.4 and 1.5 display the differences between the two protection strategies 1:1 protection and 1+1 protection. Additionally, it should be mentioned that a changing between different protection strategies is often difficult to accomplish because of different hardware requirements (e.g., switching device versus signal splitting device). A special form of 1+1 protection is the *Subnetwork Connection Protection* (SNCP) which does not necessarily protect a complete routing path of a demand but only a sub-path by 1+1 protection.

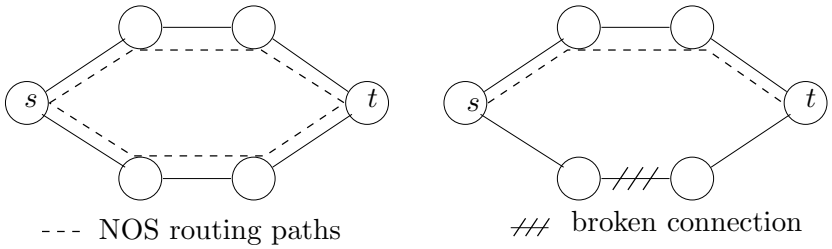

Figure 1.5: In contrast to 1:1 protection, in the case of 1+1 protection no low priority data traffic can be used along a dedicated backup path. In normal operating state, two paths are used for demand routing form s to t. At location t the better signal is accepted, the other one is dropped. In case of a single component failure that does not affect one of the terminal nodes s or t, one of the signal streams can still be received at t without loss of switching time.

The design of the network, the dimensioning of network capacities, as well as routing strategies and additional planning parameters may support the chosen protection strategies furthermore. The higher the *connectivity* of different network locations, i.e., the more disjoint paths may be used for demand routing between certain network locations, the more replacement paths can be found. The drawback of a high network connectivity is the large number of connections between locations that have to be installed and lead to an expensive network in terms of installments and maintenance. A simple method to establish a basic 2-connectivity of a certain subnetwork is to choose a ring structure to connect the subnetwork's nodes.

A routing planning decision which aims at minimizing the impact of component failures is to restrict the length of routing paths. The shorter a routing path, the less components it passes, the less the possibility to be affected in the case of a component breakdown. A protection concept that allows for an easy modeling of a survivability idea is the *diversification* mechanism. A diversification parameter $\delta_k \in [0, 1]$ defines a maximum fraction of an arbitrary demand k that may be routed through any component which may fail. If $\delta_k < 1$, the demand k is splitted and routed on different paths through the network. In case of single component failure, only a part of the demand k may be affected and not the whole demand. Diversification can be combined with other protection mechanisms. In our model, we use the diversification idea basically as a tool to realize other protection mechanisms, especially SNCP protection (see Chapter 3).

1.2.8 Multi-layer aspects in the planning process

As mentioned in Section 1.2.6, SDH can be seen as an interface layer between a superordinate layer, such as ATM for example, and the physical network structure. From the ATM layer's point of view, the SDH network provides connections and paths to route ATM demands. An integrated planning procedure which takes the various technical requirements and characteristics of the different network layers into account is

difficult to implement and in practice rarely done. The planning process is usually divided into multiple steps, each of them corresponding to a single network layer. The result of the planning process of a single layer are demands induced in the underlying layer and so on. A great problem arises with the realization of security mechanisms in the network, because realizing protection mechanisms in a superordinate layer does not automatically lead to an induced realization in an underlying layer.

SDH and physical layer

A logical link in the SDH layer, as an STM-1 link for example, is realized by a path in the physical layer. This path consists of one or more physical links. From the knowledge of the SDH network layout, there can be no assumptions deduced about the physical network layout and the logical link realization in the physical layer. Figure 1.6 shows a possible network configuration for an SDH layer and the underlying physical layer. In contrast to the SDH network, there is an additional node u in the physical layer,

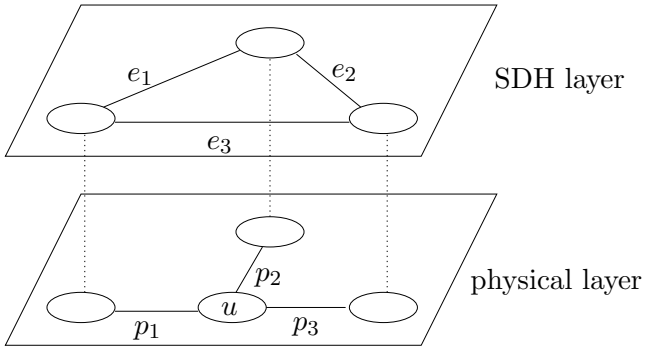

Figure 1.6: Example for the physical realization of logical links.

which is used to establish each of the logical links e_1, e_2, and e_3 respectively. The logical link e_1, for instance, is realized by the physical path (p_1p_2) [notation: a path is denoted by the concatenation of its links]. Consider two paths in the SDH layer: (e_1) and (e_2e_3). They are edge disjoint in the SDH layer, but not in the physical one, since the physical link p_2 is used both for the realization of e_1 and e_2. That situation is a severe problem for planning survivable networks, since the failure of nodes and links happens actually in the physical layer, but has to be mapped into the logical layer. However, if the physical realization of the logical layer is known, it is possible to define appropriate operating states which map the failure of single or multiple components. To take the possible failure of the link p_1 of the former example into account, there has to be the operating state $\{e_1, e_3\}$, whereas the failure of the physical location u can be modeled by the multi-failure state $\{e_1, e_2, e_3\}$. In Section 3.2.2 we describe how to use operating states in the mathematical problem modeling to avoid this kind of problems.

ATM and SDH layer

Symmetrically to the relationship between SDH and physical layer, it is possible to focus on the influences of ATM layer planning on the SDH layer planning process. An ATM link corresponds to a path in the SDH layer consisting of one ore more links. Therefore, a routing defined in the ATM layer has an impact on the routing in the SDH layer. With the model of Chapter 3.2, it is possible to ensure survivability restrictions for the SDH layer to reduce the impact of physical component failures. However, if there are already such restrictions in the implementation of the ATM routing, it should be possible to map these survivability concepts into the SDH layer as well.

The ATM layer induces demands in the SDH layer. Large demands can be splitted up into sets of demands with smaller demand values which are routed separately through the network. For the solution process of the reconfiguration tasks, we will split demands into the basic routing unit of VC-12 in the SDH layer (see Section 3.3.1). The disadvantage of the splitting process is the loss of information about properties of a large sized induced demand in the underlying layer. For example, if survivability mechanisms such as diversification for the routing of a large sized ATM demand were implemented in the ATM layer, this information could not be equivalently passed through to the number of SDH demands of size one w.r.t. the basic unit VC-12. In Section 3.2.2, we introduce the concept of commodity groups which help to transform properties that belong to a superordinate layer demand to the corresponding group of demands in the SDH layer.

Thus, with our model it is possible to plan only a single SDH layer, but account for the characteristics of the underlying physical layer as well as for the existing routing in an superordinate layer. In this way, it is possible to include aspects of a multi-layer planning in the process of a single-layer SDH planning.

Chapter 2

Reconfiguration Scenarios

After the introduction about telecommunication network layout in general and the corresponding planning tasks, we focus on the specific planning scenarios that are covered by this thesis. As mentioned before, the regarded planning process does not aim at a change of hardware configuration. Conversely, the main task is to make the best of a given hardware and routing situation. When performing an integrated network planning which respects both the network layout, the installed capacities, and the definition of a feasible routing, capacity can be seen as the binding link between routings and hardware (see [Krö03], for example). For the reconfiguration scenarios considered in this thesis, hardware installments and capacities are immutable parameters. An initial routing is the basis for the configuration of a new one. The initial routing is the binding link between the fixated hardware installments with given capacities on the one hand and a desired new routing on the other hand. In this chapter, we present several reconfiguration scenarios that represent the kind of planning tasks covered by the solution methods of this thesis. The list of problems is not complete, additional planning tasks are also conceivable. First of all, we comment on the term *efficient reconfiguration* and the idea of bounding the reconfiguration planning in certain ways. Afterwards, we present examples for the usage of such an efficient reconfiguration. Two of the reconfiguration instances will serve as examples for the development of the more precise mathematical model in Chapter 3.

2.1 A note on *efficient reconfiguration*

In Chapter 1, the typical multi-layered structure of modern telecommunication networks w.r.t. different data transmission technologies is described. Obviously, an integrated network planning respecting different technology layers promises better planning results compared to single layer planning procedures. However, because of the complexity of the planning process, it is usually divided into the planning of single technology layers as for example the planning of the ATM layer with the corresponding ATM demands and an SDH layer planning process that is almost independently accomplished of the ATM planning. The only connecting link between the different

layer planning processes are the lower-layer demand specifications induced by the superordinate layer.

The demand routing within the SDH layer is typically not done automatically. In the course of a reconfiguration planning process, network planners configure routing paths for single demands or aggregated demand sets. For a demand of size VC-12, it has to be decided into which larger container it should be embedded and how to route these larger containers. Although the actual nesting procedure is performed by the multiplexing devices, the configuration of the routing paths for large virtual containers and the decision which small containers are transported, is only made by the responsible SDH network planner. This is the most important reason to restrict the number of changes of a given routing which serves as basis for reconfiguration. Additionally, it might be possible that parts of the network are not full-operational during a reconfiguration procedure.

Within fixed hardware installments, there are usually different possibilities to choose feasible routings. Depending on the strategical objectives, some routings are better than others. Possible objectives are for instance the reduction of costs or a better distribution of link loads in the network. The reconfiguration of a given routing can lead to a better routing w.r.t. the declared objective. Often it is even possible to calculate optimal routings for the demands of a given network. The reconfiguration of an initial routing into an optimal routing might imply a large number of routing path changes. Because the realization of the reconfiguration is performed by network planners and not automatically, the number of implied changes is often too large and the optimal reconfiguration is practically not applicable. Therefore, from our point of view, an *efficient reconfiguration* has to keep large parts of a given initial routing. For a practical applicable routing reconfiguration, the changes of demand routing paths must not be too large. It must be possible to control the complexity of a reconfiguration process.

This thesis deals with the task to improve routings with this definition of *efficient*. An implementation of the algorithm developed in Chapter 4 can be seen as a support tool for an SDH network planner who has to realize such a reconfiguration of an initial routing. From the network planner's point of view, a reconfigured routing is optimal, if the best solution w.r.t. the planning objective is found, realized with a practically applicable number of routing path changes.

To our best knowledge, there are no publications that cover planning processes for limited routing reconfiguration in telecommunication networks in general or in SDH networks in particular.

2.2 Reconfiguration applications

Planning scenarios which only involve changing a given routing without expansion of the hardware installation are manifold. In this section, a number of possible reconfiguration tasks are presented. All these applications are covered by the algorithm

developed in Chapter 4. Since the model of Chapter 3 is very general, further optimization scenarios that are covered by the model and algorithm developed in this thesis are imaginable. The common initial situation to all these scenarios is the following: A network is given together with an initial routing, i.e., for each demand in the network there is a specification of end-to-end paths to fulfill the communication demand requirements. The planning task is to find alternative routing paths for a subset of the demands to achieve a certain planning objective. A modification of hardware installments is not part of the actual reconfiguration process.

Connection clearing

This scenario outlines the impact of the strategic decision to clear a connection permanently from the network. Telecommunication service providers are not necessarily proprietors of their operating networks. Networks might be partially or completely be leased from other telecommunication companies for own service providing purposes. For different reasons it can be useful to clear a connection. If a connection is leased from another company, its clearing will reduce costs as long as the clearing process itself is not too expensive. On the other hand, a company that is hiring out connection capacities to other companies and has enough reserve capacity could reconfigure its demand routing to gain free capacity on connections which then can be leased to other companies.

Capacity reduction

Capacity reduction can be seen as a generalization of the connection clearing planning task. The motives to reduce capacity in the network are similar to the ones above. Capacity of connections is leased in specified block sizes. Instead of abstaining from all off the leased capacity blocks, it can be desired to do without only a part of the leased capacity blocks.

Including additional demand specifications

In Chapter 1, we stressed the importance of a reasonable data traffic estimation in the planning process. Routing decisions are based on demand forecasts and the resulting routing paths for demands remain constant for a certain time. Usually, the corresponding time horizon spans from a couple of weeks up to several months.

New traffic estimations may lead to problems concerning the pre-configured demand routing paths. The decrease of demands does not have an impact on the feasibility of a routing, the corresponding routing paths can simply be removed from the routing. However, the increase of demands or the specification of new ones have to be accurately considered in the planning process. In general, it is not sufficient to find a separate routing for the new or increased demands and combine it with the initial routing. Due to capacity restrictions, it might be necessary to partially reconfigure the initial routing as well, because initial demand routings with admissible alternative

routing paths may occupy capacity resources that are necessary for a feasible routing of the additional or increased demands.

Strengthening feasibility restrictions

Whenever there is a decision to restrict the feasibility of a routing, it could happen that an originally defined routing is not admissible w.r.t. new conditions. Additional protection mechanisms to be applied to a given routing, e.g., including SNCP protection for all or a limited set of demands, are one example of tightening a *feasibility* definition.

Shortening of routing paths

Occasionally, the routing capacities in the network have to be increased. If a demand forecast results in routing paths that exceed the capacities of some network components, additional capacities have to be installed. New connections might be established between network nodes. Since long paths (w.r.t. the number of edges [*hops*]) consume capacities on many edges and usually pass more potentially failing components than short paths, it is often desired to find short routing paths. For some routings a hop limit is defined, i.e., a routing is only feasible if all routing paths contain at most a certain number of edges. Another feasibility limitation derived from a hop limit is the restriction to paths that contain at most the number of edges of a shortest path plus an additional hop limit.

After the installation of new connections, it might be necessary to reconfigure an initial routing, because new short paths are available and the initial routing is infeasible w.r.t. certain hop limits.

Link load distribution

The *link load distribution* is a network performance indicator, which describes the capacity consumption of network links. The more free capacity on network links, the more flexible the network, as the integration of new demands will usually be relatively easy. The implementation of the *restoration* mechanism, as briefly sketched in Section 1.2.7, will be easier if enough free capacity is available, to configure restoration paths.

Contrary, a large amount of free capacities can be seen as wastage of resources. Depending on the current operational situation and strategic decisions, there may be several definitions of an optimal link load distribution which may even contradict each other. It is often desirable to obtain evenly distributed high link loads throughout the network with a certain amount of reserved capacities for security reasons on each edge. A great problem for link load scenarios are the so-called *bottleneck links*. With bottleneck links or *bridges* one denotes connections which are unique connections between different sub-networks.

For example, in the loosely connected network in Figure 2.1, all demand between loca-

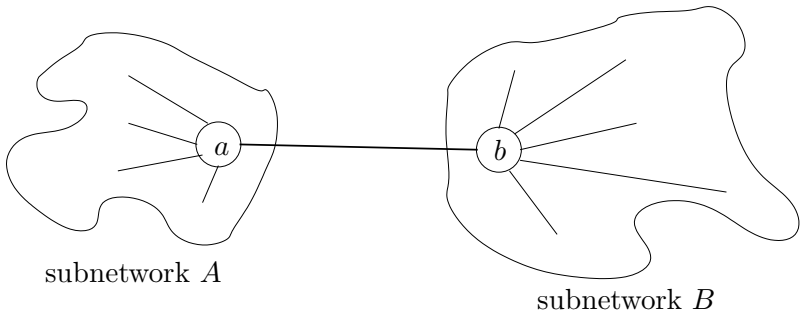

Figure 2.1: Potential bottleneck between the locations a and b.

tions in the subnetwork A and locations in the subnetwork B has to be routed over the bottleneck link between the locations a and b. In terms of graph theory a bottleneck link in the network corresponds to a *cut* in the supply graph which consists only of a single edge. If the routing capacity on this bottleneck link hardly exceeds the routing demand between A and B its link load is very high and no reconfiguration procedure is able to reduce this value. An optimization task as minimizing the maximal link load in the network will deliver no improvement if the maximal link load is achieved on a bottleneck link. To avoid these problems, we propose the definition of weights for edges to increase or decrease the importance of link load reduction on single connections (see Section 3.2.1).

There is another variant of the link load distribution scenario in which the maximum link load reduction is not the objective. A maximum link load for all edges can be defined as fixed planning parameter. In this case, the reconfiguration task is to reduce all link loads below the given limits with as few changes of routing paths as possible.

Cost reduction

If the operating costs of the telecommunication network may be reasonably attributed to the usage of connection capacity in the network, one intention of an optimization planning procedure could be cost reduction for this network. For instance, if a network consists completely of leased links, there are costs specified for the usage of each single edge. By reconfiguring the initial routing w.r.t. the cost structure, a more efficient routing could lead to a reduction of operating costs. Again, it is possible either to define cost reduction as objective for this scenario or to set a cost limit which should be achieved by a reconfiguration process with as few changes to the initial routing as possible.

2.2.1 Partial reconfiguration

All of the sketched reconfiguration scenarios can be applied to both the complete set of demands and only to a subset. In the latter case we will talk about *partial*

reconfiguration. Partial reconfiguration may be obtained by the same methods as the complete reconfiguration. The initial routing is divided into a partition of a fixated routing and a partition which contains paths that may be replaced by others to improve the routing.

Example 2.1. *A given routing does not implement any survivability constraints. For a set of particular critical demands a protection mechanism like 1+1 protection is to be introduced. The routing of the other demands should not be affected. In this case the routing paths of the other demands would be fixed, and only the routing of the protection candidates would be released for reconfiguration. Additional backup paths have to be found and some of the protection demands will probably have to change their original routing paths.*

Each of the former reconfiguration scenarios can also be applied only to subnetworks of telecommunication networks. The division of the telecommunication network into subnetworks can be done by geographical criteria, for example. Only demands routed through specified subnetworks in the initial routing will be considered for rerouting purposes. Partial reconfiguration contains both pros and cons. Advantages are the usually smaller reconfiguration problems and a smaller number of routing path changes. The drawback of restricting the reconfiguration process to subnetworks is that in general it is not clear whether optimal solutions to the restricted scenario are also globally optimal. Again, *optimality* depends on the specific reconfiguration scenario.

In this chapter, we presented a set of practical interesting reconfiguration scenarios for the operational planning process of telecommunication networks. The list of scenarios is not complete, we only intend to give the reader an impression of the wide range of different reconfiguration tasks. For the formulation of a mathematical model, the development of an algorithm to solve these tasks, and the implementation of the algorithm, we focus primarily on four of the presented reconfiguration scenarios: connection clearing, adding of new demands, shortening of routing paths, and link load reduction. Nevertheless, the solution approach developed in this thesis can be applied to a larger set of reconfiguration problems. At the end of Chapter 3, we show how further scenarios sketched in this chapter can be modeled with only slight changes to the formulation of the mathematical model.

Chapter 3

Mathematical Model

In this chapter, we develop a mathematical model for the reconfiguration tasks described in Chapter 2. Although there is a wide range of reconfiguration problems, starting from cost optimization to shortening routing path lengths, all these problems can be modeled by similar mathematical formulations. In the remainder of this thesis, we distinguish only between two problem specifications. All of the reconfiguration tasks aim at configuring an optimal routing with a small number of changes of the initial one, where the exact meaning of *optimal* depends on the specific scenario. The main reason for a distinction between two specifications is that a number of optimization tasks aims directly at minimizing the number of changes, while another set of scenarios only requires a restriction of the number of changes. We formulate a integer and a mixed-integer linear program to fulfill both tasks: minimizing and bounding the number of changes when reconfiguring a given network.

In terms of linear optimization, the models differ in their objective functions and in their constraint set. However, it will turn out that the solution methods to both of the model formulations are very similar.

The development of a mathematical model is useful in many respects. On the one hand, there is a more precise problem formulation as the informal verbal description from before. Otherwise, a (integer/mixed-integer) linear program formulation can often be solved using ideas and algorithms from *Combinatorial Optimization*. Therefore, this chapter can be seen both as a more precise description of the optimization problems presented in Chapter 2 and as a basis for the solution approach of Chapter 4.

For the task of optimizing the network with a given upper bound on the number of changes, we develop a *mixed-integer linear program* (MIP) **BoundNoC**. The reconfiguration tasks with the common intention of minimizing the number of changes in a reconfiguration process will be represented by the formulation of the *integer linear program* (IP) **MinNoC**. The remainder of this chapter is organized as follows: In Section 3.1, we introduce all the parameters and variables that are used for model development. Section 3.2 provides the mathematical problem formulation for both the **MinNoC** and the **BoundNoC** task. In Section 3.3, we discuss the advantages and

disadvantages of several modeling decisions with respect to runtime and model complexity.

The objective functions of the reconfiguration tasks represented by the **BoundNoC** model depend on the specific problem description. We choose the link load reduction scenario as an exemplary application for the development of the mathematical model formulation for **BoundNoC** reconfiguration problems. In the last part of this chapter, we categorize the reconfiguration problems which were described in Section 2.2 as **MinNoC** or **BoundNoC** problems and show how to choose parameters to apply the corresponding model formulation.

3.1 Parameters and variables

We distinguish between parameters and variables. The set of parameters include the structure of the network, e.g., locations and connections, and routing capacities on the network's connections. The initial routing is also part of the parameter set as well as planning decisions like the definition of diversification values. Depending on the reconfiguration scenario, there can be further parameters defined.

3.1.1 Parameters

In the following, we give a short description of the parameters used in the remainder of this chapter. The parameters are divided into network, demand, routing and survivability parameters. However, this separation is not strict and is only used to provide clarity. For a brief overview of all parameters and variables, see Table 3.1.2.

Network The telecommunication network is represented by an undirected graph $G = (V, E)$. The node set V corresponds to locations in the network. E represents the set of connections between network locations which can be chosen for the routing of telecommunication demands. For each edge $e \in E$, the capacity parameter C_e states the maximum number of basic routing units, i.e., VC-12 (see Section 1.2.6), that can be used for routing purposes.

For the **BoundNoC** model, we introduce additional edge weights w_e for all $e \in E$.

Demands/Commodities Demands are defined between pairs of locations. Each pair has a communication demand, i.e., a specific number of basic routing units that must be routed through the network. Communication demands in a telecommunication network can be interpreted as *commodities* in a *multi commodity flow problem* on the mathematical modeling level. Therefore, when talking about commodities, we usually refer to the mathematical model. The set of all commodities will be denoted by $\mathcal{K}$. Each commodity $k \in \mathcal{K}$ has a source $s_k \in V$ and a sink $t_k \in V$. The number of basic routing units that have to be routed between s_k and t_k for a specific commodity k is denoted by the *demand value* $d_k \in \mathbb{N}$.

Routing With $\mathcal{P}_k$ we denote the set of all feasible paths for commodity $k \in \mathcal{K}$. A feasible path is a sequence of edges connecting s_k to t_k. The feasibility of paths may be restricted by a maximum number of edges that can be used for a connection. Such a maximum edge number is also called a *hop limit* on the path. Hop limits are represented by the parameter $l_k \in \mathbb{N}$ for all $k \in \mathcal{K}$.

With $\mathcal{Q}_k \subseteq \mathcal{P}_k$ we denote the set of chosen paths for the initial routing of commodity k. The amount of consumed link capacity for transporting a commodity on a path through the network is called the *flow* of the corresponding path. We introduce a flow parameter $p_k \in \mathbb{N}$ for each commodity $k \in \mathcal{K}$. It corresponds to the number of basic routing units routed along each path used for the routing of commodity k. As mentioned before, in the case of planning SDH layers the basic routing unit is a VC-12. In other words, the parameter p_k denotes the number of VC-12 paths combined to route the communication demand of k. Typical values for p_k are either 1 or d_k. In the former case the communication demand for k is splitted and routed along d_k paths separately through the network, whereas in the latter case there is a single path routing for the complete communication demand for commodity k. Other choices of p_k lead to a routing in fixed block sizes for fractions of the communication demand of k (see Section 3.3). For the model, the choice of d_k and p_k is restricted to values, such that: $\frac{d_k}{p_k} \in \mathbb{N}$.

Example 3.1. *For an arbitrary commodity $k \in \mathcal{K}$ let $d_k = 2$ and $p_k = 1$. Then, a communication demand of two basic units has to be routed between s_k and t_k. On each path that is used for this task, exactly one basic unit has to be routed. A feasible solution consists of $\frac{d_k}{p_k} = 2$ different paths for this routing.*

Survivability As mentioned in Section 1.2.7, there is a distinction between *protection* and *restorations* mechanisms. In case of component failure the former one uses dedicated backup paths for the routing of affected demands, while the latter one tries to find a feasible routing after service breakdown. We consider only the protection mechanism *diversification*. It can be used to implement different protection strategies like *SNCP* for example.

To model the failure of links or nodes in the network, we use *operating states*. The set of all operating states is denoted by S. A single operating state will usually be denoted by $s \in S$. The situation in which all connections and all hardware components at each location are operational is called *normal operating state* (NOS). It is denoted by $s = 0$. Each operating state different from NOS is either a *single failure* or a *multi failure state*. It describes which components of the network are out of service. Therefore, each $s \in S$ is a set containing network elements (nodes and/or links) which may potentially breakdown simultaneously. Often, multi-failures in a certain layer are caused by a single failure in a subjacent layer (see Section 1.2.8).

In our model it is possible to aggregate commodities to *commodity groups*. Restrictions as diversification (see below) for example can be stated for a set of commodities. This is often useful to propagate protection mechanisms from a superordinate layer to the

current network layer. If a commodity induced by a superordinate layer is divided into a set of commodities in the currently regarded layer, diversification conditions can be formulated for the complete set of divided commodities. The set of commodities $\mathcal{K}$ is split up into disjoint commodity groups K_i w.r.t. an arbitrary index set $\mathbb{I}$:

$$\mathcal{K} = \bigcup_{i \in \mathbb{I}} K_i, \quad K_j \cap K_k = \emptyset, \quad j \neq k, \quad j, k \in \mathbb{I}.$$

Diversification is applied to the model w.r.t. commodity groups. For each commodity group K_i there is a *diversification parameter* δ_{K_i} which denotes the maximum fraction of common demand values for the commodities of this group that is allowed to pass through a potentially failing component.

Example 3.2 (operating states)**.** *If $S = \{0\} \cup V \cup E$ then all components and all links can cause a single failure. If $S = \{0\} \cup \{\{v_1\}, \{v_2\}\}, v_1, v_2 \in V$, only the two network nodes v_1 and v_2 can fail, but not simultaneously. The synchronous failure of v_1 and v_2 would be denoted by $S = \{0\} \cup \{\{v_1, v_2\}\}$. All other locations and all links are expected to be fail-safe.*

3.1.2 Variables

The most important variables in this chapter are the *path flow variables* $f_k(P) \in \{0,1\}$. Each $f_k(P)$ states whether a certain path P is chosen to route parts of the demand (exactly p_k, see above) of commodity $k \in \mathcal{K}$. The task of an optimization algorithm applied to the model is to find the optimal combination of path flow variables which allows a feasible routing and optimizes a given objective function. These path flow variables are used for all of the following models.

As proposed in Chapter 2, one of the exemplary reconfiguration problems we will investigate in more detail is the link load reduction scenario. With $\alpha_e \in [0, 1]$ we denote the fraction of occupied routing capacity for each edge $e \in E$.

3.2 Mathematical formulation

In this section, we develop objective functions and constraints to formulate the reconfiguration tasks presented in Chapter 2 as integer and mixed-integer linear programs respectively. First, we ignore all survivability constraints and focus primary on similarities and differences of different reconfiguration problems. Afterwards, we introduce survivability constraints. The outcome of this section will be the two models **BoundNoC** and **MinNoC**. which are the basis for the solution approach of Chapter 4.

3.2.1 Non-survivable networks

In a first modeling attempt all components of regarded networks are assumed to be full operational all the time. No protection mechanisms are implemented, no operating states are defined.

Name	Description
$G = (V, E)$	supply graph
$C_e \in \mathbb{N}$	edge capacity
$w_e \in \mathbb{R}$	edge weight
$\mathcal{K}$	set of commodities
$s_k \in V$	source of commodity k
$t_k \in V$	sink of commodity k
$l_k \in \mathbb{N}$	length restriction for a feasible $s_k t_k$-path
$\mathcal{P}_k$	set of feasible paths for k
$\mathcal{Q}_k$	current routing of k
$d_k \in \mathbb{N}$	demand value of k
$p_k \in \mathbb{N}$	single path flow of k
$\frac{d_k}{p_k} \in \mathbb{N}$.	number of paths for k
S	set of operating states
$K_i \subseteq \mathcal{K}$	commodity group
$\delta_{K_i} \in [0,1]$	diversification parameter
$f_k(P) \in \{0,1\}$	flow variable for path P
$\alpha_e \in [0,1]$	edge multiplier ($\approx$ link load on edge e)

Table 3.1: List of all used parameters and variables for Chapter 3. The unit of measurement for C_e, d_k, and p_k is the number of basic routing units

Bounded number of changes

The first considered problem is the one of optimizing the network with a bounded number of changes. In the remainder of this section, we want to cope with the task of reducing the link loads in the given routing as much as possible. In other words, it is necessary to find a feasible routing in which the free capacity on each link is as large as possible. The MIP for this task has to fulfill the following constraints:

- The amount of capacity to be reduced will be measured in percentage with the free variable α_e for each edge $e \in E$. The overall flow on a given link may not exceed the initial capacity on that edge times α_e:

$$\sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{P}_k : e \in P} p_k \cdot f_k(P) \leq C_e \cdot \alpha_e. \tag{3.1}$$

- The overall flow on all paths of a given demand $k \in \mathcal{K}$ has to match exactly the demand value of k:

$$\sum_{P \in \mathcal{P}_k} p_k \cdot f_k(P) = d_k. \tag{3.2}$$

- Let $b \in \mathbb{N}$ be the upper bound for the number of changes of the given routing. Then $B = \sum_{k \in \mathcal{K}} |\mathcal{Q}_k| - b$ denotes the number of paths of the current routing which have to be reused. The corresponding inequality reads:

$$\sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{Q}_k} f_k(P) \geq B. \tag{3.3}$$

- The last constraint is an upper bound on the values of α_e. If α_e was unbounded, this could lead to optimal solutions with values of α_e greater than 1 on some edges. This would mean: To optimize the overall link load in the network, the capacity on some edges has to be increased. However, since the aim of this thesis is optimal reconfiguration rather than expansion planning, α_e values greater than 1 are not permitted:

$$\alpha_e \leq 1, \qquad e \in E.$$

The optimization objective in this scenario is to minimize the sum of all α_e. If this sum is as small as possible, the link load distribution in the network is optimized. Additionally, in Section 3.1, we introduced edge parameter w_e which allow for a weighting of edge importance. The larger the value of w_e for a certain edge, the more important it is to gain free capacity on this edge. Negative values of w_e lead to the fixation of $\alpha_e = 1$ for the corresponding edge $e \in E$. However, this does not necessarily mean that the *occupied* capacity on this edge is actually at 100%. If $w_e = \frac{2}{|E|}$ for all $e \in E$, the *average link load* in the network will be minimized.
The complete MIP reads as:

(MIP 3.A) [**BoundNoC** without survivability]

$$\begin{aligned}
\min \sum_{e\in E} w_e \cdot \alpha_e & \\
\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:e\in P} p_k \cdot f_k(P) - C_e \cdot \alpha_e &\le 0, && e \in E,\\
\sum_{P\in\mathcal{P}_k} p_k \cdot f_k(P) &= d_k, && k \in \mathcal{K},\\
\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{Q}_k} f_k(P) &\ge B, &&\\
\alpha_e &\le 1, && e \in E,\\
\alpha_e &\ge 0, && e \in E,\\
f_k(P) &\in \{0,1\}, && k \in \mathcal{K}, P \in \mathcal{P}_k.
\end{aligned}$$

Minimizing the number of changes

In the former section, the given routing was feasible. The task was to improve the routing. However, for some of the presented reconfiguration tasks of Chapter 2, it is not clear whether the initial routing is furthermore feasible. In these cases, we focus primarily on finding a feasible routing which contains as many initial routing paths as possible. Therefore, the objective function for these problems reads as:

$$\max \sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{Q}_k} f_k(P). \tag{3.4}$$

As before, a feasible routing has to fulfill capacity restrictions for each edge of the supply graph and demand constraints. The new edge restriction differs from (3.1) since there is no need for a multiplier α_e:

$$\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:e\in P} p_k \cdot f_k(P) \le C_e. \tag{3.5}$$

By omitting the continuous variables α_e the mathematical model for this task is no mixed-integer linear program but an integer linear program:

(IP 3.B) [**MinNoC** without survivability]

$$\begin{aligned}
\max \sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{Q}_k} f_k(P) & \\
\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:e\in P} p_k \cdot f_k(P) &\le C_e, && e \in E,\\
\sum_{P\in\mathcal{P}_k} p_k \cdot f_k(P) &= d_k, && k \in \mathcal{K},\\
f_k(P) &\in \{0,1\}, && k \in \mathcal{K}, P \in \mathcal{P}_k.
\end{aligned}$$

Obviously, if the initial routing is feasible itself, then nothing will happen. The optimal solution for the linear program is to reuse all initial routing paths. There have to be no changes of the routing at all.

3.2.2 Survivable networks

Typically, telecommunication networks are not completely fail-safe. Components break down due to environmental influences, sabotage or simply because of abrasion. Usually, networks are designed and dimensioned to be survivable, i.e., even in case of a failure of network components, as much telecommunication demand as possible has to be fulfilled; furthermore, the loss of data must be minimized. Planning survivable networks requires decisions at different planning stages. The strategic planning decisions of network layout and dimensioning must provide a certain degree of connectivity and reserve capacities to define replacement routing paths.

This section deals with the more operational planning decisions that must be made to ensure survivability for telecommunication networks. As mentioned before, the concept of diversification is used to implement protection mechanisms. Two main decisions must be made:

- **operating states**
  The definition of appropriate operating states is the most important part of this protection planning. Similar to the demand forecast, the more precise this definition, the better the routing w.r.t. to survivability. Each single component and each combination of components that might fail concurrently has to be a single operating state. However, as will be shown in the remainder of this section, each operating state introduces not only a single constraint but a set of constraints into the integer and mixed integer linear program formulations, respectively. That means: the more precise the determination of operating states, the better the routing, but also the larger the MIP and IP formulations and the more computational problems arise. Therefore, there has to be a tradeoff between the precision of failure estimation and size of the mathematical models. In practice, usually only single failures are respected in the planning process. To be more precise, in most cases only single link failures are taken into account. However, to be able to to respect the embedding of the currently regarded SDH network layer into the physical layer as described in Section 1.2.8, it is possible to define multi-failures as well.

- **diversification parameter and commodity groups**
  After the definition of operating states, there has to be the decision of how to deal with potentially failing components for routing purposes. To limit the impact of a component breakdown, the amount of data traffic that is routed along such a component is restricted. In this model, the restriction of traffic amount is defined w.r.t. commodity groups, i.e., only a certain part of routing traffic of such a commodity group might be routed along a routing paths that contains

potentially failing components. Components assumed to be fail-safe may be passed by an arbitrary part of the demand of a commodity group. A typical value for the diversification parameter δ_{K_i} is $\frac{1}{2}$ to ensure SNCP, for instance.

Now, the following constraints are added to the mathematical models:

$$\sum_{k \in K_i} \sum_{P \in \mathcal{P}_k : s \in P} p_k \cdot f_k(P) \;\leq\; \delta_{K_i} \cdot \sum_{k \in K_i} d_k, \quad s \in S, i \in \mathbb{I}. \tag{3.6}$$

Remark 3.1. *$s \in P$ denotes the situation in which a path is affected by a component failure. If s is a multi failure state, $s \in P$ means that P passes at least one of the network components combined in s. However, the possibly failing network components must not be one or both of the terminal nodes of P. Operating states containing one or both terminal nodes of a path P are explictly excluded from $s \in P$, because there is no need for a routing on replacement paths for the corresponding commodity k.*

In Section 3.1, we introduced the parameters d_k and p_k which correspond to the demand value of a demand $k \in \mathcal{K}$ and the exact flow on each path for this demand. Thus, $\frac{d_k}{p_k}$ is the exact number of routing paths for commodity k. However, the chosen paths need not be disjoint. Only the diversification constraints (3.6) enforce disjointness w.r.t. to operating states.

3.2.3 BoundNoC and MinNoC

Combining the model formulations from section 3.2.1 with the survivability restriction (3.6), we get the two basic models **BoundNoC** and **MinNoC**. With these formulations, we are able to not only model the introduced problems but also a number of different tasks. In Section 3.3.3, we give an overview on some other interesting tasks and how to choose parameters in our basic models to transform them into related problems. The complete basic model **BoundNoC** for optimizing the network's link loads when only a constant number of changes of the initial routing is allowed, reads as:

(MIP 3.C) [**BoundNoC**]

$$
\begin{aligned}
\min \sum_{e\in E} w_e \cdot \alpha_e & & \\
\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:e\in P} p_k \cdot f_k(P) &\le C_e \cdot \alpha_e, & e \in E,\\
\sum_{P\in\mathcal{P}_k} p_k \cdot f_k(P) &= d_k, & k \in \mathcal{K},\\
\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{Q}_k} f_k(P) &\ge B, & \\
\sum_{k\in K_i} \sum_{P\in\mathcal{P}_k:s\in P} p_k \cdot f_k(P) &\le \delta_{K_i} \cdot \sum_{k\in K_i} d_k, & s \in S, i \in \mathbb{I},\\
\alpha_e &\le 1, & e \in E,\\
\alpha_e &\ge 0, & e \in E,\\
f_k(P) &\in \{0,1\}, & k \in \mathcal{K}, P \in \mathcal{P}_k.
\end{aligned}
$$

Similarly, the complete formulation of the **MinNoC** model for finding a feasible routing respecting demand, capacity and diversification constraints and reuse as many initial routing paths as possible reads as:

(IP 3.D) [**MinNoC**]

$$
\begin{aligned}
\max \sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{Q}_k} f_k(P) & & \\
\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:e\in P} p_k \cdot f_k(P) &\le C_e, & e \in E,\\
\sum_{P\in\mathcal{P}_k} p_k \cdot f_k(P) &= d_k, & k \in \mathcal{K},\\
\sum_{k\in K_i} \sum_{P\in\mathcal{P}_k:s\in P} p_k \cdot f_k(P) &\le \delta_{K_i} \cdot \sum_{k\in K_i} d_k, & s \in S, i \in \mathbb{I},\\
f_k(P) &\in \{0,1\}, & k \in \mathcal{K}, P \in \mathcal{P}_k.
\end{aligned}
$$

3.3 Discussion of the model

3.3.1 Integer versus binary flow variables

In the data definition of Section 3.1, the flow variable on path P to route commodity k is defined as $f_k(P) \in \{0,1\}$. Since the flow on path p is binary, we can interprete $f_k(P)$ as a decision variable of using path P for routing commodity k or not. Thus, counting changes between different routings is simple. The model is very flexible, because demands between the same end-nodes can be treated completely different. Each of it can have its own protection mechanism, for example. However, this flexibility leads to a huge number of path variables and restrictions in both of the models.

If the problem formulation for the initial routing contained demands with integer flow values, these demands would have to be split up into demands with a flow value of size one to be able to count changes between different routings. If an approximation to an optimal solution suffices or if the number of variables becomes too huge for further computation by splitting the original demands into demands of size one, it is possible to adapt the demand value d_k and the path multiplier p_k to route in fixed block sizes of p_k on each path for a demand k. However, since d_k and p_k are parameters that are fixed to their values before optimization, an optimal solution for such an adapted formulation may not be optimal for the original formulation. Furthermore, if the problem formulation for fixed demand blocks had no feasible solution at all, it would not be clear whether there is an solution for the original formulation without blocks of aggregated demands.

3.3.2 Parameter choices for SNCP protection

Since SNCP is often used as protection mechanism in SDH networks, it is necessary to integrate this concept into a mathematical model. With the models **BoundNoC** and **MinNoC**, it is possible to model SNCP protection for a single demand or a group of demands. In this section we show how to choose parameters and demand groups to realize the SNCP mechanism. Let $k \in \mathcal{K}$ be an arbitrary commodity which should be protected by SNCP. Define K_i as a demand group which consists only of demand k. Since the demand has to be duplicated, set $d_k := 2$. This duplicated demand has to be routed along two paths. Therefore, define $p_k := 1$. These two routing paths have to be disjoint with respect to operating states which means that each component which can possibly fail, is passed by only one unit of this demand. Therefore, set $\delta_{K_i} := \frac{1}{2}$. Now, the demand constraints (3.2) and diversification constraints (3.6) ensure SNCP protection for commodity k. Likewise, it is possible to protect a group of demands by SNCP. Let $K_i \subseteq \mathcal{K}$ be an arbitrary demand group. For each $k \in K_i$ define $d_k := 2$ and $p_k := 1$. The diversification parameter for the demand group δ_{K_i} has to be set to $\frac{1}{2}$ to use the SNCP concept for a group of demands.

3.3.3 Application of the mathematical model

This section deals with the appropriate parameter choice to transform the reconfiguration scenarios from Section 2.2 into one of the two basic model formulations **MinNoC** and **BoundNoC** respectively.

Connection Clearing

In the connection clearing scenario, a set of the supply edges has to be temporarily or continuously removed from the network. Therefore, the given initial routing might not be feasible anymore. We apply the **MinNoC** formulation to model this reconfiguration scenario. The capacity on the edges to be removed is fixed to 0. The task is to keep as many paths of the initial routing as possible, while a feasible routing for all demands

has to be found. Replacement paths have to be assigned for demands that were initially routed along edges to be removed.

Capacity reduction

A variation of the connection clearing scenario is the capacity reduction scenario. In this case the capacity on a subset of the edge set will not necessarily be set to zero but reduced in comparison to the edges' initial capacities. In the case of capacity reduction, a feasible initial routing may not be admissible anymore. Thus, this scenario can also be seen as an application of the **MinNoC** model. The task for this planning scenario is to find a feasible routing w.r.t. capacity, demand and diversification constraints with as few changes of the initial routing as possible.

Additional demand specifications

This optimization scenario is another application of the **MinNoC** problem class. The task is to find a feasible routing respecting all demand restrictions (initial and new ones), capacity, and diversification constraints. No further change in parameter specification is required.

Shortening routing paths

This scenario does not require a transformation of parameters but of the initial input data. In a first step, all routing paths exceeding a certain length parameter l_k defined for a commodity $k \in \mathcal{K}$ are removed from the input. The second step is to solve the problem similar to the previous scenario with additional demand specifications. After removing a set of routing paths, the situation is quite similar. There is a subset of demands for which no initial routing is defined (anymore). The task is to find a feasible routing that contains as many of the (feasible) initial routing paths as possible. Attention has to be payed to the way of counting changes in this scenario. It has to be decided whether the removal of an infeasible routing path counts as a change or not. The impact of the parameter l_k to the reconfiguration task is hidden in the set $\mathcal{P}_k$ for each $k \in \mathcal{K}$. As defined before, this set contains all feasible routing paths for commodity k. Paths exceeding the length restriction are not contained in this set.

Network costs

With the same methods as before, it is possible to optimize a network w.r.t. its cost structure and bound the number of changes of a current routing. Particularly, for the special case of a network with a cost structure that depends proportional on the used capacity on the network's edges, we simply have to redefine the edge weights w_e to be the edge costs and use the model **BoundNoC**. Likewise, the traffic load of the network's nodes can be computed and costs for node usage can be assigned. For instance, let the parameter y_v define the costs for routing one unit of demand through

node $v \in V$. The variable β_v will measure the demand traffic for node v:

$$\sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{P}_k : v \in P} p_k \cdot f_k(P) = \beta_v, \qquad v \in V.$$

Adapting the objective function delivers:

$$\min \sum_{e \in E} w_e \cdot \alpha_e + \sum_{v \in V} y_v \cdot \beta_v.$$

Partial reconfiguration

Since the growth of a network is usually a process of adding demands and increase capacities step by step, it is far from probable that the network at a late point in time of its evolutionary process can be characterized as optimal, regardless of the definition of *optimality*. Thus, a reconfiguration of the network would likely improve the situation. However, reconfiguring the complete network is often not practically applicable. Sometimes, not only the number of changes is bounded, but a specific set of demands is to be omitted from this procedure. Suppose that the task is again to optimize the network's link loads. Let $\mathcal{F} \subseteq \mathcal{K}$ the set of fixed demands whose routing paths should not be changed. We use the model **BoundNoC**. The objective function remains unchanged. It does not suffice to split the set $\mathcal{K}$ into disjoint sets of fixed and reroutable demands and just reconfigure the reroutable ones, since the free capacity on each link has to be aligned to the fixed demand values which have to be routed along this edge. Define

$$F_e := \sum_{k \in \mathcal{F}} \sum_{P \in \mathcal{P}_k : e \in P} p_k \cdot f_k(P)$$

as the fixed capacity for each edge e of the supply graph. Then, the new capacity restrictions read as:

$$\sum_{k \in (\mathcal{K} \setminus \mathcal{F})} \sum_{P \in \mathcal{P}_k : e \in P} f_k(P) \leq \alpha_e \cdot (C_e - F_e), \qquad e \in E.$$

The other constraints have to be restricted to the reroutable demands:

$$\sum_{P \in \mathcal{P}_k} p_k \cdot f_k(P) = d_k, \qquad k \in (\mathcal{K} \setminus \mathcal{F})$$

and

$$\sum_{k \in (\mathcal{K} \setminus \mathcal{F})} \sum_{P \in \mathcal{Q}_k} f_k(P) \geq B.$$

We only consider the flow variables of demands that are not fixed:

$$f_k(P) \in \{0, 1\}, \qquad k \in (\mathcal{K} \setminus \mathcal{F}), P \in \mathcal{P}_k.$$

Again, for each $e \in E$, α_e is bounded by 0 and 1.

Chapter 4

Algorithmic Approach

The focus of this chapter is an algorithmic procedure to solve the optimization problems which have been presented informally in Chapter 2 and more precisely in Chapter 3. After a short overview on the algorithm, we focus on details concerning column-generation and branch-and-bound techniques. Although the applied solution methods are well-known for other multi-commodity flow problems, it turns out that counting changes in the routings complicates the solution process both in the column-generation and in branch-and-bound subproblem. In particular, integrality conditions on variables generated during the column-generation procedure raise problems for the solution approach.

4.1 First survey on the algorithm

The IP **MinNoC** and the MIP **BoundNoC** are similarly structured. Both formulations contain an exponential number of binary variables.

A common approach for the solution of integer and mixed-integer linear programs is the application of a *branch-and-bound* procedure. Usually, the integrality restriction on the variables is relaxed and the resulting linear program is solved. Integrality of the variables is ensured by bounding and enumeration techniques.

Remark 4.1. *Whenever we use* fractional *or* integer *to characterize a solution, we refer to the path flow variable values of the specific solution and not to the solution value or the link load variable in the* ***BoundNoC*** *case. Thus, a solution is called integer if and only if all path flow variables are integer. Similarly, a solution is called fractional if at least one path flow variable value is fractional.*

A technique to solve large scale linear programs is the column-generation approach. Only a subset of the variables is actually generated. The solution process is applied to this subset of all modeled variables. Nevertheless, it is possible to guarantee optimality for the complete linear program.

The combination of both techniques is called *branch-and-price*. At each node in the

branch-and-bound tree, the column-generation technique is applied to construct missing variables if necessary. The developed algorithm for the solution of the reconfiguration tasks presented in Chapter 2 consists of such a branch-and-price framework.

4.2 Column-generation

An introduction to the column-generation approach can be found in the linear and integer programming literature (e.g., [Chv83]). Desrosiers and Lübbecke ([DL05] and [LD02]) give an introduction to column-generation in the context of network optimization. The reader is assumed to be familiar with the *duality theory* of linear optimization.

4.2.1 Introduction to column-generation

The main idea of the column generation approach is to solve a large scale linear program without stating each variable explicitly. Several optimization problems have a structure in which only a small set of variables is different from zero in a feasible solution. In these cases, it is possible to respect the majority of variables only implicitly in the solution process and to state only a small number of variables explicitly. Column-generation is mainly based on the following results and properties of *duality theory* of linear optimization:

- Variables of the primal linear program correspond to constraints of the dual linear program formulation and vice versa.
- If a primal linear program has an optimal solution, then there is also an optimal solution of the dual LP formulation and the objective values are equal.
- Adding variables to a linear program might only improve and will never impair the LP solution. Contrary, adding constraints to a linear program might only impair and will never improve a solution.

The large scale primal LP formulation is called *master program* (MP). Neglecting a subset of the variables leads to the so called *restricted master program* (RMP). Similarly, we denote the dual linear program that corresponds to the master program *dual master program* (DMP) and the dual linear program that corresponds to the RMP *dual restricted master program* (DRMP). Figure 4.1 sketches a tabloid scheme of the master program, the contained restricted master program and the corresponding dual LP formulations. The column-generation approach is divided into two parts:

1. Solve RMP to optimality.
2. Decide whether the optimal solution for the RMP is also an optimal solution for the MP. If not, find variables that are modeled for the MP but neglected in the RMP formulation, and could improve the current optimal RMP solution, if considered in the optimization process.

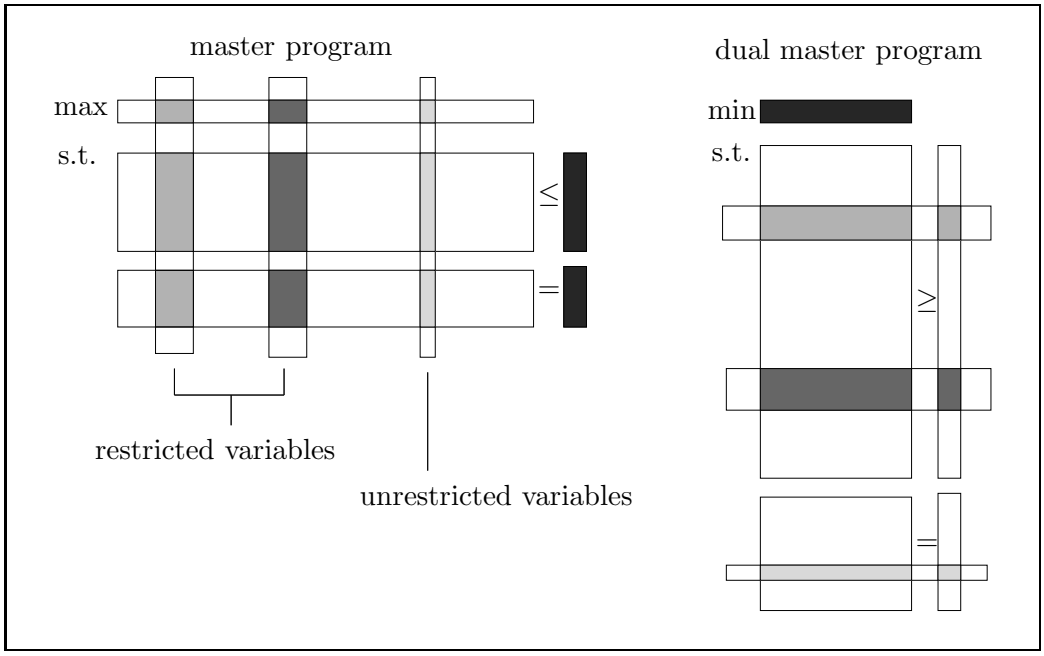

Figure 4.1: Relationship between master program, restricted master program, dual master program, and dual restricted master program. The RMP (shaded parts of the left figure) contains only a small subset of the modeled variables for the MP. Similarly, the DRMP (shaded parts of the right figure) contains only a subset of all modeled constraints. All RMP variables correspond to a DRMP constraint and vice versa. Unrestricted primal variables correspond to restrictions of the equality block of the DMP and restricted primal variables like $x_i \geq 0$ correspond to restrictions of the inequality block. The right hand side of the MP corresponds to the objective function of the DMP.

The second step is called *pricing problem*. Variables that are priced out are included into the RMP formulation, and the process is iterated until no more missing variables can be identified.

The remainder of this column-generation introduction is restricted to the solution of the pricing problem.

The RMP respects all constraints of the master program. An optimal solution of the RMP will satisfy these constraints. If there is an optimal solution for the RMP, there is also an optimal solution for the DRMP with an equal objective value. More important than the objective value of the optimal dual solution is the assignment of dual variable values. Suppose that a constraint of the DMP is violated by the current optimal solution of the DRMP. After adding this constraint to the DRMP, there are two possibilities:

1. The modified DRMP is not feasible anymore.

2. The objective value of an optimal solution of the modified DRMP might be impaired compared to the objective value of the original DRMP formulation.

In the first case there is no optimal solution for the primal MP. In the following, we assume the second case. After adding the primal variable corresponding to the dual violated constraint to the RMP, there must be an optimal solution with the same objective value as the one of the modified DRMP. In the primal case, it can only be an improvement compared to the original RMP solution. If an optimal solution of the DRMP respects all modeled constraints of the DMP, it is also an optimal solution for the DMP. Similarly, an optimal solution of the corresponding RMP must be an optimal solution for the MP. No further variables need to be added to the set of primal variables in the RMP.

If the pricing procedure is exact, i.e., violated dual constraints are reliably detected, this method delivers exact results. Either an optimal solution of the MP is found or there is no optimal solution because of infeasibility or unboundness. However, the efficiency of this procedure depends on two aspects. At first, there must be a reasonable initialization of the RMP and second, the detection of violated dual constraints must not be too difficult.

The remainder of this section describes the application of the column-generation approach to the mathematical model, developed in Chapter 3. First, we focus on the pricing problem for instances of the **MinNoC** problem class without survivability constraints, and second, on the feasibility of the RMP. Afterwards, we show that the results of the **MinNoC** column-generation method can easily be adopted to solve the **BoundNoC** model as well. Section 4.2.3 deals with the topic of using column-generation for the survivable model formulations. It turns out that the pricing problem can not always be resolved as easily as before if survivability constraints are included into the problem formulation.

4.2.2 Column-generation for non-survivable network reconfiguration

The number of feasible paths (which correspond to the variables in the models **MinNoC** and **BoundNoC**) usually grows exponentially with the size of the network. In fact, the only potential restriction is the length restriction l_k for a commodity $k \in \mathcal{K}$. However, in an optimal integer solution, only a small number of the feasible paths will be chosen to route the demands through the network (exactly: $\frac{d_k}{p_k}$). Using column-generation techniques, it is possible to start with a restricted set of variables and add additional paths to the problem only if needed to improve the current solution. The master program for all reconfiguration tasks is the linear program relaxation of the **MinNoC** and **BoundNoC** IP and MIP formulations, respectively. The binary path flow variables are relaxed to continuous variables (see LP formulation below). In this section, the survivability restrictions are neglected.

First, we develop a solution for the pricing problem for instances of the **MinNoC** problem classes for the case that an initial feasible solution exists and is to be improved by adding additional variables to the RMP. Afterwards, we discuss the feasibility problem and show how to modify the problem formulation to find a feasible solution or to decide that none exists. The last part of this section focuses on the adaption of the solution approach to **BoundNoC** problems.

Solving the pricing problem for the MinNoC LP relaxation

As mentioned before, the pricing problem will be solved by identification of violated constraints of the DRMP. The LP relaxation for **MinNoC** (without survivability) reads as:

(LP 4.A) [LP relaxation for **MinNoC** without survivability]

$$\max \sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{Q}_k} f_k(P)$$

$$\sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{P}_k : e \in P} p_k \cdot f_k(P) \leq C_e, \qquad e \in E, \tag{4.1}$$

$$\sum_{P \in \mathcal{P}_k} p_k \cdot f_k(P) = d_k, \qquad k \in \mathcal{K}, \tag{4.2}$$

$$f_k(P) \leq 1, \qquad k \in \mathcal{K}, P \in \mathcal{P}_k, \tag{4.3}$$

$$f_k(P) \geq 0, \qquad k \in \mathcal{K}, P \in \mathcal{P}_k.$$

The corresponding dual linear program contains the following dual variables: μ_e for the capacity constraints (4.1), π_k for the demand constraints (4.2), and λ_P for the flow restrictions (4.3). The complete dual linear program can be expressed as:

(LP 4.B) [Dual LP of (LP 4.A)]

$$\min \sum_{e\in E} C_e\mu_e + \sum_{k\in\mathcal{K}} d_k\pi_k + \sum_{k\in\mathcal{K}}\sum_{P\in\mathcal{P}_k} \lambda_P$$

$$\sum_{e\in P} p_k\mu_e + p_k\pi_k + \lambda_P \geq 1, \qquad k\in\mathcal{K}, P\in\mathcal{Q}_k, \tag{4.4}$$

$$\sum_{e\in P} p_k\mu_e + p_k\pi_k + \lambda_P \geq 0, \qquad k\in\mathcal{K}, P\in(\mathcal{P}_k\setminus\mathcal{Q}_k), \tag{4.5}$$

$$\mu_e \geq 0, \qquad e\in E,$$
$$\lambda_P \geq 0, \qquad k\in\mathcal{K}, P\in\mathcal{P}_k,$$
$$\pi_k \in \mathbb{R}, \qquad k\in\mathcal{K}.$$

In terms of column-generation, this linear program is the dual master program in which all constraints are modeled. Each optimal solution of the RMP corresponds to an optimal solution of this DMP with a restricted set of constraints. The task of the pricing procedure is to identify restrictions which are modeled in the general DMP formulation but violated in a specific DRMP solution.

Constraints of type (4.4) are satisfied from the beginning, since the paths from the initial routing are in the LP from the beginning. Therefore, all these variables are considered throughout the complete solution process. Thus, it suffices to examine constraints of type (4.5) when searching for violated dual constraints:

$$-\pi_k \leq \sum_{e\in P}\mu_e + \frac{\lambda_P}{p_k}, \qquad k\in\mathcal{K}, P\in(\mathcal{P}_k\setminus\mathcal{Q}_k) \tag{4.6}$$

$$\Leftrightarrow -\pi_k \leq \min_{P\in(\mathcal{P}_k\setminus\mathcal{Q}_k)}\{\sum_{e\in P}\mu_e + \frac{\lambda_P}{p_k}\}, \qquad k\in\mathcal{K}. \tag{4.7}$$

The transformation (4.5) $\Leftrightarrow$ (4.6) holds because parameter p_k must not be zero (see Chapter 3).
Criterion (4.6) is a necessary and sufficient condition to identify missing primal path variables. All feasible path variables for a commodity k that are not used for the initial routing have to fulfill this condition. If there is one path variable that violates this constraint, this variable is missing in the primal restricted LP formulation.

The reformulation (4.7) of this criterion shows how to test this condition for all possible feasible path variables of a commodity. It suffices to test whether it is fulfilled for a shortest path w.r.t. the dual edge weights μ_e and the dual variable λ_P. If a shortest path w.r.t. this length definition fulfills the condition, the condition is also fulfilled by each other feasible path for the corresponding commodity. If the shortest path violates this constraint, it has to be added to the primal path variable set. By increasing the set of primal variables and resolving the restricted LP, the values of the dual variables might change and in a subsequent condition check of (4.7) other feasible path variables

for commodity k may turn out to be shortest paths.

However, the search for violated constraints can not simply be solved by a shortest-path procedure for all commodities $k \in \mathcal{K}$. Formulation (4.7) provides two restrictions to a shortest-path search:

1. The index set $P \in (\mathcal{P}_k \setminus \mathcal{Q}_k)$ for the minimization function restricts the set of feasible paths. A nicer problem to solve would be:

$$-\pi_k \leq \min_{P \in \mathcal{P}_k} \{\sum_{e \in P} \mu_e + \frac{\lambda_P}{p_k}\}, \qquad k \in \mathcal{K}. \tag{4.8}$$

2. The addition of $\frac{\lambda_P}{p_k}$ to the sum of dual variables μ_e complicates the shortest path search. In contrast to the μ_e which define weights to the edges of the supply graph and are constant for all path variables, the dual variable λ_P depends on the currently regarded path P. A simple shortest-path search in the supply graph would not be sufficient to identify a shortest path w.r.t. $\sum_{e \in P} \mu_e + \frac{\lambda_P}{p_k}$.

The remainder of this section deals with the solution of these two difficulties. Proposition 4.1 shows that the restriction of the index set is easy to handle and Proposition 4.2 investigates the additional complexity of the problem due to the second restriction.

Proposition 4.1. *For an optimal solution of (LP 4.B) holds: (4.7) $\Leftrightarrow$ (4.8).*

Proof. Let $k \in \mathcal{K}$ be arbitrarily chosen. Each path from the set $\mathcal{Q}_k$ fulfills constraint (4.4) because the path variable is contained in the LP from the beginning. Obviously, such a path does also satisfy (4.5). Therefore, if there is a routing path P for commodity k that violates (4.5), it follows that $P \in \mathcal{P}_k \setminus \mathcal{Q}_k$. The formulations (4.7) and (4.8) are equivalent in this case. □

Proposition 4.2. *For an optimal solution of the restricted primal LP and the corresponding restricted dual LP holds: For each $k \in \mathcal{K}$,*

$$\min_{P \in \mathcal{P}_k} \{\sum_{e \in P} \mu_e + \frac{\lambda_P}{p_k}\}$$

can be found using a K-shortest-paths algorithm with $K \leq \frac{d_k}{p_k} + 1$.

Proof. Let $k \in \mathcal{K}$ be arbitrarily chosen. λ_P are the dual variables corresponding to the upper flow bound constraints of type (4.3). From the theory about complementary slackness (e.g., Theorem about weak complementary slackness in [Grö04]),

$$\lambda_P > 0 \Rightarrow f_k(P) = 1, \qquad P \in \mathcal{P}_k$$

holds.
Furthermore, the demand constraints (4.2) imply that the number of $P \in \mathcal{P}_k$ with

$f_k(P) = 1$ is at most $\frac{d_k}{p_k}$ and therefore, the number of path variables P for which the dual variables λ_P are greater than 0 is also bounded by $\frac{d_k}{p_k}$. More precisely:

$$|P \in \mathcal{P}_k : \lambda_P > 0| \leq \frac{d_k}{p_k} \tag{4.9}$$

Interpret μ_e as edge costs in the supply graph. Let $P^{(1)} \in \mathcal{P}_k$ be a shortest path in the supply graph w.r.t. these costs. We distinct the following two cases:

- **case 1** ($\lambda_{P^{(1)}} = 0$):
  In this case, it is clear that

  $$\sum_{e \in P^{(1)}} \mu_e = \min_{P \in \mathcal{P}_k} \{\sum_{e \in P} \mu_e + \frac{\lambda_P}{p_k}\}$$

  and we are done.

- **case 2** ($\lambda_{P^{(1)}} > 0$):
  Similar to the notation of **case 1**, $P^{(i)}$ denotes an i^{th} shortest path w.r.t. the dual edge weights μ_e for commodity k .
  The set of the $\frac{d_k}{p_k} + 1$ shortest paths for k contains at least one path $P^{(j)}$ with $\lambda_{P^{(j)}} = 0$, because of (4.9). This implies:

  $$\min_{P^{(i)},\, i \in \{1,..,\frac{d_k}{p_k}+1\}} \{ \sum_{e \in P^{(i)}} \mu_e + \frac{\lambda_{P^{(i)}}}{p_k}\} = \min_{P \in \mathcal{P}_k} \{\sum_{e \in P} \mu_e + \frac{\lambda_P}{p_k}\}. \tag{4.10}$$

  If (4.10) is satisfied, the proposition will be proven. To verify (4.10), suppose that there is a path $P^* \in \mathcal{P}_k$, such that:

  $$\sum_{e \in P^*} \mu_e + \frac{\lambda_{P^*}}{p_k} < \min_{P^{(i)},\, i \in \{1,..,\frac{d_k}{p_k}+1\}} \{ \sum_{e \in P^{(i)}} \mu_e + \frac{\lambda_{P^{(i)}}}{p_k}\}. \tag{4.11}$$

  Since P^* is not contained in the set of $\frac{d_k}{p_k} + 1$ shortest paths,

  $$\sum_{e \in P^*} \mu_e \geq \sum_{e \in P^{(i)}} \mu_e$$

  for all $i \in \{1, .., \frac{d_k}{p_k}+1\}$. Therefore, to fulfill (4.11), λ_{P^*} must be less than $\lambda_{P^{(i)}}$ for **all** $i \in \{1, .., \frac{d_k}{p_k}+1\}$. However, $\lambda_{P^*} \geq 0$ and there is at least one $i \in \{1, .., \frac{d_k}{p_k}+1\}$, such that $\lambda_{P^{(i)}} = 0$. Therefore, assumption (4.11) is wrong.

The shortest path P w.r.t. $\sum_{e \in P} \mu_e + \frac{\lambda_P}{p_k}$ for all $P \in \mathcal{P}_k$ can be found using a K-shortest-paths algorithm with $K \leq \frac{d_k}{p_k} + 1$. □

Algorithm 1 Pricing algorithm for **MinNoC**

1: **repeat**
2: compute optimal primal and dual solution of the restricted LP relaxation (e.g., use Simplex algorithm)
3: set edge weights in supply graph to μ_e
4: **for all** $k \in \mathcal{K}$ **do**
5: **if** $-\pi_k > 0$ **then**
6: **for** $i = 1$ **to** $\frac{d_k}{p_k} + 1$ **do**
7: $P^i = i$th-shortest path for k
8: **if** $-\pi_k > \sum_{e \in P^i} \mu_e + \frac{\lambda_{P^*}}{p_k}$ **then**
9: add P^i to the set of primal variables
10: **end if**
11: **end for**
12: **end if**
13: **end for**
14: **until** no primal variables added to variable set

The main result of this section is Algorithm 1 on page 40 which solves the pricing problem for the linear relaxations of **MinNoC** problem formulations without survivability constraints. If the restricted master program (RMP) has an optimal solution, it is possible to identify all path variables that are currently missing in the restricted problem version and could improve the optimal solution.

Lines 9 and 10 of Algorithm 1 differ slightly from the proposed procedure of Proposition 4.2. For simplification purposes, all of the found $\frac{d_k}{p_k} + 1$ paths are tested for compliance with restriction (4.8) and added to the primal variable set otherwise. For the correctness of the algorithm it makes no difference, since increasing the primal variable set might only improve the primal solution.

Remark 4.2. *The pricing problem for many multi commodity flow problems, especially for demand routing in networks, can often be solved by a shortest path procedure, instead of a K-shortest-paths algorithm (see [Wes00], for instance). By omitting (4.3) from the primal linear program relaxation (LP 4.A), the pricing problem for the* ***MinNoC*** *setting would be simplified to a shortest path problem. However, the flow variables $f_k(P)$ would not be bounded by 1, but by $\frac{d_k}{p_k}$. In this case, the simplification of the pricing problem will be accomplished by a more difficult branch-and-bound procedure.*

The following example illustrates the path generation process for a small routing reconfiguration, due to capacity reduction in the network.

Example 4.1. *Let $\widetilde{\mathcal{P}_k} \subseteq \mathcal{P}_k$ be the set of already generated paths from the set of all feasible paths for a commodity $k \in \mathcal{K}$. In the remainder of this example, a path is denoted by the concatenation of its nodes. Figure 4.2 displays a small network. Link e_4 should be removed from the network. The demand and routing information can be seen in Table 4.1.*

$e \in E$	C_e
e_1	1
e_2	1
e_3	1
e_4	1→0
e_5	2

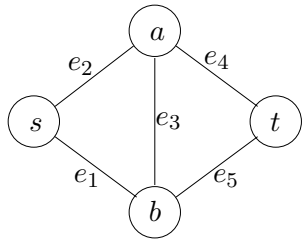

Figure 4.2: Example network for a link cutback scenario. Connection e_4 will be removed.

$\mathcal{K}$	s_k	t_k	d_k	p_k	$\mathcal{Q}_k$	$\widetilde{\mathcal{P}_k} \setminus \mathcal{Q}_k$
k_1	s	t	1	1	$\{(sat)\}$	$\{(sbt)\}$
k_2	s	t	1	1	$\{(sbt)\}$	$\{(sabt)\}$

Table 4.1: Demand and routing information table for Example 4.1

.

The primal linear program relaxation applied to this setting reads as:

(LP 4.C)

$$
\begin{aligned}
\max\ & f_{k_1}(sat) + f_{k_2}(sbt) && \\
f_{k_1}(sbt) + f_{k_2}(sbt) &\le 1 && \\
f_{k_1}(sat) + f_{k_2}(sabt) &\le 1 && (4.12)\\
f_{k_2}(sabt) &\le 1 && (4.13)\\
f_{k_1}(sat) &\le 0 && \\
f_{k_1}(sbt) + f_{k_2}(sabt) + f_{k_2}(sbt) &\le 2 && (4.14)\\
f_{k_1}(sat) + f_{k_1}(sbt) &= 1 && \\
f_{k_2}(sabt) + f_{k_2}(sbt) &= 1 && \\
0 \le f_{k_1}(sat), f_{k_1}(sbt), f_{k_2}(sabt), f_{k_2}(sbt) &\le 1 && (4.15)
\end{aligned}
$$

An optimal solution for this linear program is $f_{k_1}(sbt) = f_{k_2}(sabt) = 1$ and $f_{k_1}(sat) = f_{k_2}(sbt) = 0$ with an objective value $z = 0$. To decide whether this solution is optimal for the master program, we have to look at the dual linear program:

(LP 4.D)

$$\begin{aligned}
\min \mu_{e_1} + \mu_{e_2} + \mu_{e_3} + 2\mu_{e_5} + \pi_{k_1} + \pi_{k_2} + \lambda_{sat_1} + \lambda_{sbt_1} + \lambda_{sabt_2} + \lambda_{sbt_2} \\
\mu_{e_2} + \mu_{e_4} + \pi_{k_1} + \lambda_{sat_1} \geq 1 \\
\mu_{e_1} + \mu_{e_5} + \pi_{k_2} + \lambda_{sbt_1} \geq 1 \\
\mu_{e_1} + \mu_{e_5} + \pi_{k_1} + \lambda_{sbt_1} \geq 0 \\
\mu_{e_2} + \mu_{e_3} + \mu_{e_5} + \pi_{k_2} + \lambda_{sabt_2} \geq 0 \\
\mu_{e_1}, \mu_{e_2}, \mu_{e_3}, \mu_{e_4}, \mu_{e_5} \geq 0 \\
\lambda_{sat_1}, \lambda_{sbt_1}, \lambda_{sabt_2}, \lambda_{sbt_2} \geq 0
\end{aligned}$$

An optimal solution to this LP is:
$\mu_{e_1} = 1, \mu_{e_4} = 2, \pi_{k_1} = -1$ *and all other variables are set to zero with an objective value* $z = 0$. *Since* $\pi_{k_2} = 0$, *constraint (4.8) holds and no further paths have to be generated for commodity* k_2. *The path (sabt) is the shortest path w.r.t.* μ_e *as edge costs for demand* k_1. *Applying criterion (4.8) for commodity* k_1 *gives:*

$$-\pi_{k_1} = 1 \not\leq 0 = \mu_{e_2} + \mu_{e_3} + \mu_{e_5},$$

which shows, that the path is missing in the path set for commodity k_1. *After adding* $f_{k_1}(sabt)$ *to the left hand sides of constraints (4.12), (4.13), and (4.14) and include it into the bounds section (4.15) of (LP 4.C), there is a new optimal solution:* $f_{k_2}(sbt) = f_{k_1}(sabt) = 1$ *and all other variables are set to zero with an objective value* $z = 1$. *The corresponding dual linear program can be obtained by adding the variable* λ_{sabt_1}, *which has to be non-negative, to the objective function of (LP 4.D) and inserting the following constraint:*

$$\mu_{e_2} + \mu_{e_3} + \mu_{e_5} + \pi_{k_1} + \lambda_{sabt_1} \geq 0.$$

Then, an optimal solution for this linear program is: $\mu_{e_1} = \mu_{e_4} = 1$, *and all other variables are set to zero with an objective value of* $z = 1$. *Now, both* $\pi_{k_1} = 0$ *and* $\pi_{k_2} = 0$ *and therefore, no further paths are missing for an optimal solution of the master program and the primal solution* $f_{k_2}(sbt) = f_{k_1}(sabt) = 1$ *is optimal.*

Feasibility problem

The pricing algorithm of the former section can only be applied if RMP, i.e., the LP relaxation (LP 4.A) with a restricted set of initial path flow variables, has an optimal solution. In this case the pricing algorithm will step by step improve the solution by adding missing primal variables. If however, there is no optimal solution for the RMP, i.e., the RMP is infeasible (it cannot be unbounded since the objective is to keep as many initial path variables as possible and the initial routing contains only a finite set of routing variables), there is no dual solution for the problem and Algorithm 1 cannot be applied. Only from the fact that there is no primal solution for the RMP, it is not trivial to determinable, whether this is caused by missing primal variables or by an infeasible master program.

To test the master program for feasibility, we use a reformulation of (LP 4.A). If there exist enough routing paths to fulfill all demand constraints (4.2), the master program (MP) will only be infeasible if the edge capacities on at least one edge in the supply graph are not sufficiently dimensioned to allow for a feasible routing of all commodities. This missing routing capacity can also be interpreted as additional amount of capacity which has to be installed in the network to allow for a feasible routing. The following problem reformulation tries to minimize this additional capacity and provides, after a reasonable initialization, always optimal solutions to an RMP. Therefore, it is possible to apply the column-generation approach to this reformulation. The solution of this process consists either of a feasible initial variable set for (LP 4.A) or a certificate that there is no feasible routing for all demands because of missing capacities.

To assure feasibility of the reformulated RMP, it is necessary to find feasible routing paths for each commodity $k \in \mathcal{K}$, i.e., paths that fulfill the demand constraint (4.2). This can easily be done using standard shortest path algorithms. Whether the set of initial paths violates the capacity constraints (4.1) is neglected at this stage of the algorithm.

Afterwards, the following linear program must be solved:
(LP 4.E) [Feasibility LP]

$$\begin{aligned} & \max -\gamma & & \\ \sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k:e\in P} p_k \cdot f_k(P) - \gamma &\leq C_e, & e \in E, \qquad (4.16)\\ \sum_{P\in\mathcal{P}_k} p_k \cdot f_k(P) &= d_k, & k \in \mathcal{K}, \\ f_k(P) &\leq 1, & k \in \mathcal{K}, P \in \mathcal{P}_k, \\ f_k(P) &\geq 0, & k \in \mathcal{K}, P \in \mathcal{P}_k, \\ \gamma &\geq 0. & \end{aligned}$$

The variable γ represents the amount of additional capacity that has to be installed on at least one link to allow a feasible routing for all demands (see (4.16)). The objective function states that this amount should be minimized (the maximization formulation is chosen because of the strong similarity to (LP 4.A)). The algorithm uses the following property:

Remark 4.3. *A feasible solution of (LP 4.E) with $\gamma = 0$ is also a feasible solution for (LP 4.A) and vice versa.*

Remark 4.4. *To ensure the resolvability of the feasibility LP, there has to be a feasible routing for all demands, i.e., path variables for each commodity that meet the demand restriction and satisfy a potential length restriction l_k. Provided that the length restrictions are not too restrictive, this is always possible if no supply edges are removed from the network. Even in the case of a connection clearing scenario, the corresponding links will not be removed. Instead, their routing capacity is fixed to zero. The advantage is*

that these links can be used for routing in the feasibility LP. If the objective value of an optimal solution of the feasibility LP is zero, these links are not used for routing purposes anyway. Otherwise, these connections cannot be cleared.

The pricing problem for the Feasibility LP

Again, the decision whether the restricted set of initial primal variables is sufficient for solving the LP to optimality will be made by identifying violated dual constraints. The formulation of the dual linear program uses the same dual variables μ_e, π_k, and λ_P as before:

(LP 4.F) [Dual LP of (LP 4.E)]

$$\begin{aligned}
\min \sum_{e\in E} C_e\mu_e + \sum_{k\in\mathcal{K}} d_k\pi_k + \sum_{k\in\mathcal{K}}\sum_{P\in\mathcal{P}_k} \lambda_P & \\
\sum_{e\in P} p_k\mu_e + p_k\pi_k + \lambda_P \;&\geq\; 0, && k\in\mathcal{K}, P\in\mathcal{P}_k, && (4.17)\\
\sum_{e\in E} \mu_e \;&=\; 1, && && (4.18)\\
\mu_e \;&\geq\; 0, && e\in E,\\
\lambda_P \;&\geq\; 0, && k\in\mathcal{K}, P\in\mathcal{P}_k,\\
\pi_k \;&\in\; \mathbb{R}, && k\in\mathcal{K}.
\end{aligned}$$

Constraint (4.18) will not be violated, since it does not depend on path flow variables. Constraints of type (4.17) are the only ones that could be violated. Thus, the pricing problem is similar to the one of the **MinNoC** LP relaxation. Criterion (4.8) and Proposition 4.2 can be adapted. The pricing problem of the feasibility problem can be solved with the same pricing algorithm as before. The result of the application of Algorithm 1 to the feasibility LP is either a certificate that the master program is infeasible because of missing routing capacities or a feasible initial variable set for the original problem formulation.

Column-generation for BoundNoC

The **BoundNoC** model is used as a reconfiguration model to improve a feasible initial routing. Therefore, in the course of the column-generation part of the algorithm, no distinction has to be made between a feasibility subproblem and an optimization subproblem. The task of identifying missing primal path variables for **BoundNoC** is essentially the same as for the **MinNoC** problem. Since the two basic models are similar, the pricing problems hardly differ at all.

(LP 4.G) [LP relaxation for **BoundNoC** without survivability]

$$\min \sum_{e\in E} w_e \cdot \alpha_e$$

$$\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k : e\in P} p_k \cdot f_k(P) - C_e \cdot \alpha_e \leq 0, \qquad e \in E, \tag{4.19}$$

$$\sum_{P\in\mathcal{P}_k} p_k \cdot f_k(P) = d_k, \qquad k \in \mathcal{K}, \tag{4.20}$$

$$\sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{Q}_k} f_k(P) \geq B, \tag{4.21}$$

$$\alpha_e \leq 1, \qquad e \in E, \tag{4.22}$$

$$f_k(P) \leq 1, \qquad k \in \mathcal{K}, P \in \mathcal{P}_k, \tag{4.23}$$

$$\alpha_e \geq 0, \qquad e \in E,$$

$$f_k(P) \geq 0, \qquad k \in \mathcal{K}, P \in \mathcal{P}_k.$$

The following variables are used for dualization: As before, there are μ_e variables for the capacity constraints (4.19), π_k for the demand constraints (4.20), and λ_P for the flow bound constraints (4.23). Additionally, we introduce ϑ for the single constraint (4.21) and ν_e for Constraints (4.22):

(LP 4.H) [Dual LP of (LP 4.G)]

$$\max \sum_{k\in\mathcal{K}} d_k\pi_k + B\vartheta - \sum_{k\in\mathcal{K}} \sum_{P\in\mathcal{P}_k} \lambda_P - \sum_{e\in E} \nu_e$$

$$p_k\pi_k + \vartheta \leq \sum_{e\in P} p_k\mu_e + \lambda_P, \qquad k \in \mathcal{K}, P \in \mathcal{Q}_k, \tag{4.24}$$

$$p_k\pi_k \leq \sum_{e\in P} p_k\mu_e + \lambda_P, \qquad k \in \mathcal{K}, P \in (\mathcal{P}_k \setminus \mathcal{Q}_k), \tag{4.25}$$

$$C_e\mu_e - \nu_e \leq w_e, \qquad e \in E,$$

$$\mu_e \geq 0, \qquad e \in E,$$

$$\vartheta \geq 0,$$

$$\lambda_P \geq 0, \qquad k \in \mathcal{K}, P \in \mathcal{P}_k,$$

$$\pi_k \in \mathbb{R}, \qquad k \in \mathcal{K}.$$

When searching for violated constraints in an optimal solution of (LP 4.H), it suffices to focus on constraints of type (4.25). These read exactly as (4.6) except for the algebraic sign of π_k. Again, since a path of the initial routing has to fulfill (4.24), it cannot violate (4.25). Thus, the pricing problem for **BoundNoC** can be solved by the application of a K-shortest-paths algorithm with K bounded by $\frac{d_k}{p_k} + 1$ as well.

K-shortest-paths algorithm for the pricing problem

In the former sections, we proved that the pricing problem for both of the models **MinNoC** and **BoundNoC** can be solved with a K-shortest-paths algorithm with

$K \leq \frac{d_k}{p_k} + 1$. In this section, we show that this bound for K is tight, i.e., there are problem instances that require the usage of a K-shortest-paths procedure with $K = \frac{d_k}{p_k}+1$. Therefore, this bound is not theoretically important, but must be respected in the solution process if optimality should be guaranteed. The following example sketches a reconfiguration problem which would not be solved to optimality if K was chosen too small.

Example 4.2. *This example is an application of the **MinNoC** model. The given routing has to be reconfigured because of capacity reduction on some links. The network layout can be seen at Figure 4.3. Demand and routing information can be found at Table 4.2.*
[Notation: As before, $\widetilde{\mathcal{P}_k}$ denotes the set of already generated paths for commodity $k \in \mathcal{K}$, and a path is denoted by the concatenation of its network nodes.]
When solving the dual LP (LP 4.B) to identify missing primal paths, the following solution is optimal: $\mu_{e_3} = 1$, $\mu_{e_4} = 0.5$, $\mu_{e_6} = 0.5$, $\pi_{k_1} = -0.5$, $\lambda_{ade_1} = 0.5$. Since π_{k_1} is less than 0, it is necessary to search for violated constraints for k_1. A shortest path algorithm may find the path (ade) with costs of: $\mu_{e_8} + \mu_{e_5} = 0$. This path has already been generated before, and does not violate (4.8):

$$-\pi_{k_1} = 0.5 \leq 0.5 = \mu_{e_8} + \mu_{e_5} + \lambda_{ade_1}.$$

In a previous iteration, this path has been generated as a shortest path violating criterion (4.8). However, the path (abe) has also costs of $\mu_{e_1} + \mu_{e_9} = 0$ and violates (4.8). If several shortest paths are of the same length, a deterministic shortest path procedure will always find only the same path again and again.

Hence, it is necessary to use a K-shortest-paths procedure with $K = \frac{d_k}{p_k} + 1 = 2$ to identify all missing primal paths.

$e \in E$	C_e
e_1	1
e_3	2→1
e_4	2→1
e_5	1
e_6	1
e_7	1
e_8	1
e_9	1

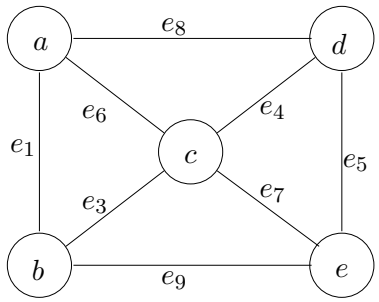

Figure 4.3: Network and edge capacity information for the capacity reduction scenario of Example 4.2. The routing capacity of both supply edges e_3 and e_4 is decreased from 2 to 1.

$\mathcal{K}$	s_k	t_k	d_k	p_k	$\mathcal{Q}_k$	$\widetilde{\mathcal{P}_k} \setminus \mathcal{Q}_k$
k_1	a	e	1	1	$\{(abcde)\}$	$\{(ace), (ade)\}$
k_2	b	e	1	1	$\{(bce)\}$	$\{(be)\}$
k_3	d	a	1	1	$\{(dca)\}$	$\{(da)\}$

Table 4.2: Demand and routing information table for Example 4.2

.

Remark 4.5. *Reducing the pricing problem to a K-shortest-paths problem has an impact on the computational complexity of the pricing problem. The complexity of a K-shortest-paths algorithm depends polynomially only on the size of the graph and on K. Since K is bounded by $\frac{d_k}{p_k} + 1$ for each commodity $k \in \mathcal{K}$, the running time for the solution of the pricing problem depends on the input parameter definition. The computational complexity of the pricing problem is therefore pseudo-polynomial. For practical applications, $\frac{d_k}{p_k}$ will be small in most cases.*

4.2.3 Column-generation for survivable networks

The mathematical model for instances of both problem types (**MinNoC** and **BoundNoC**) contains diversification constraints which allow the application of protection mechanisms like SNCP, for example. This section deals with the impact of these constraints to the column-generation approach. Considering the **MinNoC** LP relaxation (LP 4.A) with the additional diversification constraints (3.6), the corresponding dual linear program provides constraints that indicate which path variables have to be generated to find optimal solutions of the primal LP relaxation. With the same dual variables μ_e, π_k, and λ_P as before and additional dual variables $\omega^s_{K_i}$ for the diversification constraints (3.6), the complete dual linear program reads as:

(LP 4.I) [Dual LP of **MinNoC** with survivability]

$$\min \sum_{e\in E} C_e\mu_e + \sum_{k\in\mathcal{K}} d_k\pi_k + \sum_{s\in S}\sum_{i\in\mathbb{I}}(\delta_{K_i}\sum_{k\in K_i} d_k)\,\omega^s_{K_i} + \sum_{k\in\mathcal{K}}\sum_{P\in\mathcal{P}_k}\lambda_P$$

$$\sum_{e\in P} p_k\mu_e + p_k\pi_k + \sum_{i\in\mathbb{I}:k\in K_i}\sum_{s\in P} p_k\omega^s_{K_i} + \lambda_P \;\geq\; 1, \qquad k\in\mathcal{K}, P\in\mathcal{Q}_k, \tag{4.26}$$

$$\sum_{e\in P} p_k\mu_e + p_k\pi_k + \sum_{i\in\mathbb{I}:k\in K_i}\sum_{s\in P} p_k\omega^s_{K_i} + \lambda_P \;\geq\; 0, \qquad k\in\mathcal{K}, P\in(\mathcal{P}_k\setminus\mathcal{Q}_k), \tag{4.27}$$

$$\begin{aligned} \mu_e &\geq 0, & e\in E,\\ \omega^s_{K_i} &\geq 0, & i\in\mathbb{I},\ s\in S,\\ \lambda_P &\geq 0, & k\in\mathcal{K}, P\in\mathcal{P}_k,\\ \pi_k &\in \mathbb{R}, & k\in\mathcal{K}. \end{aligned}$$

For the meaning of $s \in P$, see Remark 3.1 on page 27. As before, the pricing problem consists of identifying violated constraints of type (4.27). Again, constraints of type (4.26) will not be violated and paths of the initial routing will not violate the Constraints (4.27). The pricing criterion can be transformed into:

$$-\pi_k \leq \sum_{e \in P} \mu_e + \sum_{i \in \mathbb{I}: k \in K_i} \sum_{s \in P} \omega^s_{K_i} + \frac{\lambda_P}{p_k}, \qquad k \in \mathcal{K}, P \in \mathcal{P}_k. \tag{4.28}$$

To solve the pricing problem exactly with a K-shortest-path approach as before, it is necessary to find a mapping between the dual costs of operating states and edges of the supply graph. It is necessary to define edge weights q_e for all $e \in E$ such that (4.28) can be equivalently transformed into:

$$-\pi_k \leq \sum_{e \in P} q_e + \frac{\lambda_P}{p_k}, \qquad k \in \mathcal{K}, P \in \mathcal{P}_k. \tag{4.29}$$

We call this transformation *equivalent* if the length calculation of an arbitrary path in the network leads to the same value for both formulations.

Remark 4.6. *Note the implementational consequences for the solution of the pricing problems if survivability is respected, compared to the pricing problems for tasks without survivability restrictions. In the latter case, the dual edge weights in the supply graph depend only on the dual variables of the primal edge capacity constraints. Thus, they are equal for all commodities. For a problem respecting survivability, these dual weights depend additionally on the commodity group K_i. The consequences are:*

- *If no survivability is respected, it suffices to initialize the dual edge weights once for the application of the K-shortest-path procedure of all commodities $k \in \mathcal{K}$.*
- *The dual edge weights in the supply graph have to be initialized at least once for each commodity group K_i separately, if survivability restrictions are included. However, this is only possible if the commodities $k \in \mathcal{K}$ are ordered w.r.t. commodity groups. Otherwise, the dual edge weights have to be initialized for each commodity k separately.*

Single component failures

An equivalent transformation can easily be defined if all operating states are only single failures, i.e., only one single component (edge or node) might fail at once:

- If an operating state consists of a single edge e, the dual costs of this edge μ_e and the dual costs $\omega^s_{K_i}$ for the operating state $s = \{e\}$ and the commodity group K_i can be added for an equivalent transformation: $q_e = \mu_e + \omega_s$. For any commodity $k \in \mathcal{K}$ belonging to this commodity group and any path $P \in \mathcal{P}_k$ containing e, the application of (4.28) or (4.29) leads to the same result.

- Suppose that operating state s consists of a single node. Each path containing this node as an inner node uses exactly two of the adjacent edges (terminal nodes for a commodity are supposed to be fail-safe, at least in the routing planning for this commodity, since there is no possibility to find a feasible routing otherwise). This property is due to the fact that only simple, i.e., loop-free paths are admissible. Therefore, it is possible to define: $q_e = \mu_e + \frac{\omega^s_{K_i}}{2}$ for any given $k \in \mathcal{K}$.

Note that many different operating states can influence the weight of a single edge concurrently. If only single component failures are regarded, both end nodes and and the possible failure of this edge itself could increase its dual weight q_e for a the commodities k of a commodity group K_i. Operating states that include the failure of a set of components are more difficult to handle.

Multi-failures

When dealing with multi-failures, it is generally not possible to find an equivalent mapping of dual component failure costs to the edges of the supply graph. Even the dual costs $\omega^s_{K_i}$ for the failure of only two components cannot be equivalently distributed to the dual costs of the network edges. Let operating state $s = \{v_1, v_2\}$ denote the concurrent failure of the two nodes v_1 and v_2. The dual costs $\omega^s_{K_i}$ have to be distributed to the network edges such that a path that includes only v_1, a path that includes only v_2, and a path that includes both v_1 and v_2 would have to be elongated by the same value. In general, this is not possible. If the dual costs are evenly distributed as before, i.e., the dual weight of each adjacent edge is increased by $\frac{\omega^s_{K_i}}{2}$, a path that passes both nodes is disadvantaged compared to a path that passes only one of the failing network nodes. This problem can be reduced although not completely solved if the dual weights of edges between nodes that might fail concurrently are not increased by fractions of the corresponding operating state's dual value. Multi-failures can be used to model the impact of a single failure in the underlying network to the currently regarded network layer (see Section 1.2.8). Therefore, if a single component failure in the underlying physical network enforces a concurrent failure of nodes in the SDH layer, theses nodes will often be connected. If the dual weight of the connecting links is not additionally increased by fractions of $\omega^s_{K_i}$, at least the paths that pass the two potentially failing nodes in sequence are not disadvantaged compared to paths passing only through one of the potentially failing nodes. However, paths that pass the nodes not directly in sequence are furthermore artificially elongated.

The problem becomes more complicated as the number of concurrently failing components rises. There are additional possibilities to reduce the potential error that is made by the pricing algorithm. If the algorithm is not exact, then there are paths that violate (4.29), but are not found by the K-shortest-paths algorithm, because of an imprecise transformation of dual operating state costs to dual edge weights. Therefore, it could improve the pricing result to increase the value of K. More paths are considered in the pricing step, and the potential error might be reduced.

The conclusion of this section is:

- Only in case of single component failures the pricing procedure is guaranteed to work exactly.
- In case of multi-failures, the task is to reduce the potential error that is made by the pricing procedure when ignoring path variables. Modifying dual edge weights appropriately and increasing the number of regarded paths in the pricing routine will potentially improve the quality of a solution.

Survivability and the feasibility problem

The solution of the feasibility problem for instances of the **MinNoC** problem class yields either the proof that the master program itself is infeasible or an initial variable set such that the RMP of the original problem formulation always has an optimal solution. Additional survivability constraints can easily be applied to the non-survivable solution process. The only differences to the feasibility problem of Section 4.2.2 are:

1. The primal restricted LP relaxation contains additional diversification constraints.
2. The edge costs in the supply graph for the application of a K-shortest-path algorithm depend on both the dual variables μ_e of the capacity constraints and the dual variables q_e of the diversification constraints.

With these slight changes the feasibility solution process can be applied as before. The interpretation of the objective value of the feasibility LP is also the same. It corresponds to the amount of additional capacity that has to be installed onto at least one edge in the supply graph to allow a feasible routing that respects all demand and all diversification constraints.

4.3 Branch-and-bound

The branch-and-bound procedure is a decision tree based enumeration technique which is often applied to optimization problems. At each node in the branch-and-bound tree, the solution space of the current subproblem is divided into different parts (branching). Each child node of a given node corresponds to one of these parts and is solved independently from other sibling nodes. By identifying feasible upper and lower bounds for both the current subtree and the original optimization problem, it is often possible to prune some of the branches (bounding) to avoid a complete enumeration. For a more detailed introduction, the reader is referred to the appropriate literature (e.g., [Wol98]). Desrosiers and Lübbecke ([DL05]) show, among other things, the impact of different branching rules to a constrained shortest path problem. In the course of the algorithm, we use the branch-and-bound approach to find optimal integer solutions if the optimal solution of the LP relaxation for either **MinNoC** or **BoundNoC** is fractional.

4.3.1 Branch-and-bound decisions

Generally, in the branch-and-bound process, two different kinds of decisions have to be made:

- **Branching strategy:**
  How to divide the solution space? Branching on single variables or a set of variables? The choice of a branching strategy determines the shape of the branch-and-bound tree. The aim is to build a well-balanced tree to allow for a good performance when searching for bounds in the tree nodes. Thus, at each decision node, the solution space should be divided as evenly as possible. Branching on a set of variables instead of branching on single variables usually tends to divide the problem more evenly.

- **Node selection:**
  Independently of the choice of an appropriate branching strategy, there has to be a decision about the order of processed branch-and-bound nodes. The quality of the applied node selection strategy has an impact on the number of explicitly processed branch-and-bound nodes. Generally, the exploration of the branch-and-bound tree has two goals: finding feasible solutions and prove the optimality of a feasible solution. In the case of a maximization LP, finding feasible solutions corresponds to increasing the lower bound on the objective value. Optimality can be proven by decreasing the upper bound. If a feasible solution is found with an objective value equal to the upper bound, the solution is optimal. Different node selection strategies focus differently on the two goals. A *depth-first* exploration of the branch-and-bound tree aims for example more at the feasibility aspect, whereas a *best-first* node exploration focuses more on the optimality goal.

In the remainder of this section, we focus only on theoretical aspects of the branching strategy for the branch-and-price algorithm.

Branching strategy for reconfiguration scenarios

Whenever there are paths in the optimal solution of the LP relaxation of one of the reconfiguration models with fractional values, these values are between 0 and 1. In an optimal integer solution, a path $P \in \mathcal{P}_k$ for a commodity $k \in \mathcal{K}$ has exactly one status: either it is used ($f_k(P) = 1$) or not ($f_k(P) = 0$) [if $f_k(P)$ is fixated to 0, we call the corresponding path *forbidden*]. A standard procedure for branching on binary variables is to fix the variable value to zero in one branch and to one in the other one. However, there are different problems associated with this so-called dichotomy branching:

- **Unbalanced branching tree:**
  The solution space is not evenly divided. Fixing a path flow variable $f_k(P)$ for an arbitrary commodity $k \in \mathcal{K}$ to 1 is a very strict limitation of the solution space, because there are only $\frac{d_k}{p_k}$ used paths for commodity k in a feasible integer

solution and therefore, almost all other path flow variables for commodity k are implicitly fixed to 0. In contrast to this strict limitation, fixating $f_k(P)$ to 0 hardly affects the solution space at all, since the number of admissible paths for k usually grows exponentially with the size of the network.

- **Difficult pricing problem:**
  Whenever a new branch-and-bound node is created, it is necessary to resolve the LP relaxation, because the optimal solution of the parent node may be infeasible in the current branch. Resolving the LP relaxation implies the usage of the column-generation procedure, since paths may not have been generated although necessary in the current branch for an optimal LP relaxation solution. If a path flow variable is fixed to 0 in the current branch, such a path will usually be very attractive in the pricing procedure: Fixing $f_k(P)$ to 0 leads to the following changes in the dual LP relaxation:

  1. $\lambda_P = 0$ (complementary slackness).
  2. Constraint (4.4) or (4.5), is ignored for the specific path P when solving the current dual restricted master program.

  An optimal solution of the dual LP relaxation might results in values for the dual variables μ_e, such that the forbidden primal path P turns out to be a shortest path with respect to the edge costs μ_e. Thus, it might be necessary to look for the *next shortest path*. Although it is possible to solve this problem quickly for a single forbidden path P, it might turn out to be very attractive for the pricing procedure in each subtree below the current branch-and-bound node. For each forbidden path, it can be necessary to look for a *next shortest path*, since the next shortest path after a forbidden one may be another forbidden path. In general, the pricing procedure leads to another K-shortest-paths problem. However, in contrast to the K-shortest-paths problem which is used to solve the pricing problem if the λ_p variable of a shortest path is greater than 0, the value K can not be bounded by a constant this time. K depends only on the number of concurrently forbidden paths for a commodity k. Thus, in the worst case: $K = |\mathcal{P}_k| - \frac{d_k}{p_k}$ for an arbitrary $k \in \mathcal{K}$.

Therefore, dichotomy branching on single variables does not guarantee an optimal solution when combined with the proposed column-generation approach.

Barnhart et al. ([BHV00]) propose branching on arc flow variables instead of path flow variables for integer multi-commodity flow problems. Their solution approach uses two equivalent problem formulations. While the path flow variable formulation is used for a column-generation solution approach for a primal LP relaxation, there is another arc flow variable formulation which is used for the branch-and-bound process to find feasible integer solutions. For each commodity k with fractional path flow variable values, the proposed branching strategy forbids a set of arcs for routing in one branch and another set (roughly equally sized) in a second branch. This strategy results in a fairly balanced branch-and-bound tree and furthermore, it does not affect the pricing

problem (except for the fact that a set of edges is temporarily excluded, when searching for shortest paths).

For the mathematical model of Chapter 3, it is not possible to find an equivalent arc flow formulation. This is due to the applied mechanism of counting changes in the routing, which is encoded in the usage of path flow variables and can not be equivalently transformed into arc flow variables. Nevertheless, it is possible to use the same branching concept when using path flow variables.

Branching on arc flow variables

This branching strategy has originally been designed for single path routings in directed graphs. It makes use of the fact that if the flow for a commodity is routed fractionally on path flow variables, there is a last node which all of the path flow variables have in common. At this node, the adjacent edge set is divided into two parts according to the following rules:

- The edge partitions should be evenly dimensioned.
- Each edge set contains at least one edge that is used by a fractional path flow variable.

Two new branches are generated. For each of them, exactly one of the two edge partitions is forbidden for routing of commodity k. Forbidding edges does not destroy the pricing problem. A K-shortest-paths algorithm is still applicable and can be used to solve the pricing problem exactly (see Figure 4.4).

The drawback of this branching approach are the requirements of a single path routing and a directed graph. Only with these requirements, it is guaranteed that forbidding the edge partitions as described before does not cut off parts of the solution space that might turn out to be optimal. Since only simple (loop-less) paths are feasible for the routing of demands, each path passes through at most one outgoing arc of each node. Thus, partitioning the outgoing arcs and forbidding one partition in each subtree of the current branch-and-bound node for the currently regarded commodity k preserves all feasible solutions in one of the subtrees.

If this branching approach is applied to an undirected graph, optimal solutions might be cut off. An optimal integer solution could use the two arcs that are currently used by the fractional routing paths and would be contained in different partitions if the former approach is used (see Figure 4.5).

Neglecting the restriction of using only single paths routings could also lead to an incomplete branch-and-bound tree, in which optimal solutions are not considered. Partitioning the edge set and forbidding each partition in on subtree of the current branch-and-bound node excludes routings which use edges of both partitions.

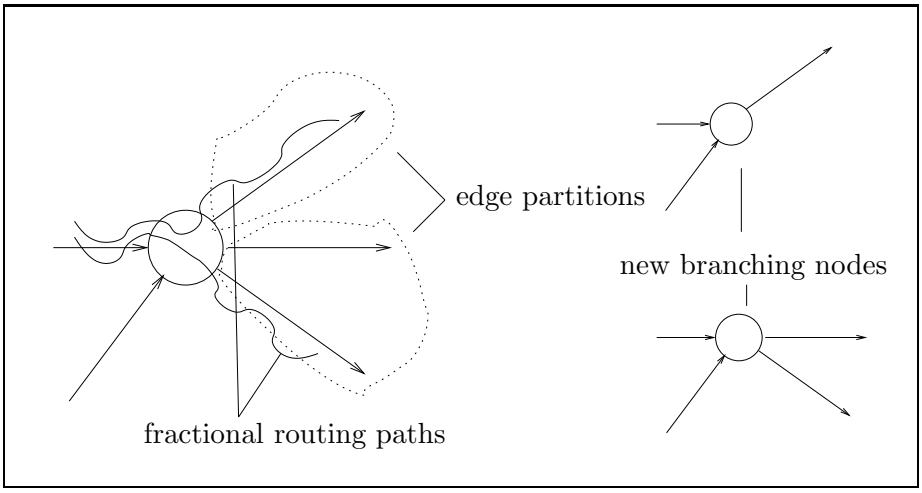

Figure 4.4: Branching procedure proposed by Barnhart et al. ([BHV00]). If the flow for a commodity is routed fractionally on paths variables, there will be a last node which have all paths in common. Such a node is illustrated at the left hand side of the picture. The set of outgoing arcs is divided into two partitions, each of them containing one arc that is used by one of the fractional path variables. In the branching process, two child nodes are created, each of them forbidding exactly one of the two edge partitions for the regarded commodity (right hand side).

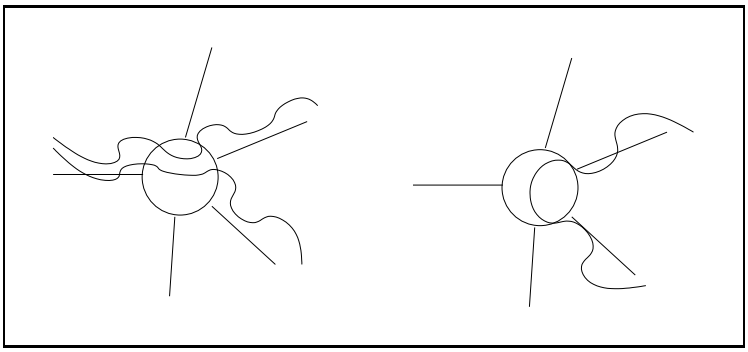

Figure 4.5: Imprecise branching for undirected networks. Partitioning the edge set in a way that the edges that are used by the fractional routing paths on the left hand side of the picture are contained in different partitions cuts off routing paths as the one on the right hand side.

Both problems affect the mathematical model of the reconfiguration tasks of the former chapter. For the problem with directed/undirected networks it suffices to slightly modify the problem formulation. To prevent the impreciseness caused by the usage of multiple routing paths, the branching rule has to be changed:

- **directed/undirected network**
  It is possible to modify the mathematical model formulation as follows: Each undirected edge $e \in E$ is replaced by two directed and antipodal arcs e^1 and e^2 to allow routings in both directions. The edge capacity constraint in all of the problem formulations is changed:

  $$\sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{P}_k : e^1 \in P} p_k \cdot f_k(P) + \sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{P}_k : e^2 \in P} p_k \cdot f_k(P) \leq C_e, \qquad e \in E.$$

  Note that there is still a single edge capacity parameter C_e which is an upper bound for the routing paths that use either e^1 or e^2 for commodity routing. Dividing the common edge capacity explicitly would not be equivalent to the former model.

  The value of the dual variable μ_e could be applied to both e^1 and e^2 when solving the pricing problem. Since no feasible routing path passes both e^1 and e^2 (feasible paths are loop-free), the pricing problem can still be solved exactly.

- **multiple routing paths**
  In the mathematical model presented in Chapter 3, exactly $\frac{d_k}{p_k}$ routing paths are required for a feasible routing of commodity k. The only possibility to guarantee a single path routing is to split such a commodity furthermore into $\frac{d_k}{p_k}$ commodities with a feasible single path routing.

The branching on the arc flow variables approach as presented by Barnhart et al. can be used for the branching problem of the reconfiguration scenarios under certain restrictions. The main advantage of this approach is a more balanced branching tree compared to the one of simple dichotomy branching. Without further problem modification, both presented branching procedures could be imprecise and cannot guarantee optimal solutions.

4.4 Summary

We are now in a position to merge the different subproblems into a complete algorithm. Figure 4.6 shows the algorithmic steps performed at each branch-and-bound node in the solution process of the integer and mixed-integer linear programs presented in Chapter 3. The main difference between problems from the **MinNoC** problem class, like the connection clearing scenario for example, and the **BoundNoC** class, like the link load reduction problem, is the need for the solution of a feasibility problem. Since the **BoundNoC** problems are feasible right from the beginning, no feasibility problem must be solved. The upper part of Figure 4.6 is skipped in this case (indicated by the

dotted arrow on the left hand side). Depending on the specific problem, there are also slight differences in the column-generation approach. These differences were described in detail in the former sections.

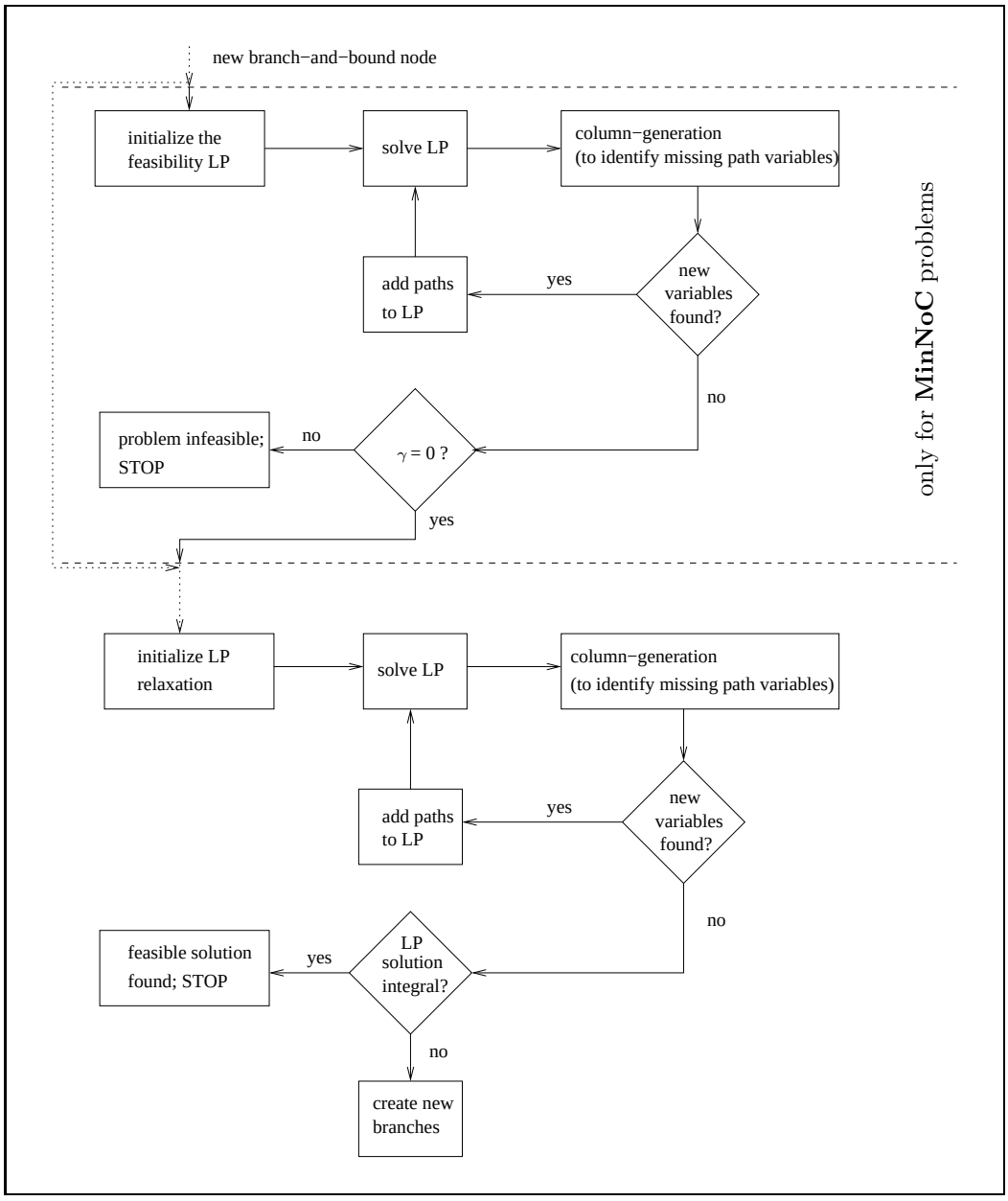

Figure 4.6: Flow chart of the branch-and-price framework.

Chapter 5

Implementational Issues

After the model formulation and the theoretical development of an algorithm to solve the reconfiguration task in the former chapters, this chapter deals with implementational aspects of the algorithm. We focus on design decisions and how they affect the algorithm. We start with a discussion of the applied K-shortest-paths (KSP) algorithm. Afterwards, we focus on implementational variations of the branch-and-bound approach.

5.1 K-shortest-paths algorithm

The pricing procedure of the algorithm of Chapter 4 requires the implementation of a K-shortest-paths procedure. The developed mathematical model and the practical networks to be reconfigured impose a set of requirements on the K-shortest-paths algorithm.

- **Network topology**
  The supply graph presented in the mathematical model consists of nodes and undirected edges. Parallel edges are also admissible. In Section 4.3.1 we discussed the possible need for a problem reformulation with a directed supply graph.

  The KSP algorithm must be able to deal both with directed and with undirected edges of the supply graph. The edge weights which define the length of a path are derived from the dual variables μ_e and q_e respectively (see Section 4.2.2 and Section 4.2.3). Since these are nonnegative variables, all edge weights are nonnegative, i.e., no negative cycles are possible.

- **Routing paths**
  For the routing purposes in telecommunication networks it is useful to restrict the set of feasible paths to loop-less ones. A path containing a loop provides no routing advantages compared to the same path without loop. Instead, it consumes unnecessarily edge routing capacities. Furthermore, the mathematical model provides the possibility to restrict the length of feasible paths w.r.t. the number of used edges.

A KSP algorithm for the presented pricing problems must be able to deal with both of these restrictions. In fact, this restricts the choice of appropriate algorithms.

Usually, a K-shortest-paths algorithm computes a shortest path or a shortest path tree (the shortest path from the source node to all other nodes) in the graph. Afterwards, nodes and edges are temporarily removed from the graph, such that the next call of the shortest path procedure results in another path.

One of the most sophisticated algorithms was presented by Eppstein ([Epp94]) who developed an algorithm which runs in $O(m+n\log n+K)$ computational runtime, where m denotes the number of arcs and n the number of nodes in the graph. The algorithm is designed for large networks and typically large values of K. The good performance is due to the usage of complex data structures and an implicit representation of paths. Instead of storing each path explicitly, the paths are stored in a tree structure in which the child nodes only store the differences to the parent node. For the relatively small dimensioned telecommunication networks we want to reconfigure and the usually small values of K, the algorithmic and administrative overhead is too large compared to the gained performance. Furthermore, the algorithm allows a cycling of paths which should be prevented in our case.

Many KSP algorithms are based on the *Optimality Principle* which states that a K^{th} shortest path is formed by J^{th} shortest sub-paths with $J \leq K$. Thus, these algorithms often calculate J^{th} shortest sub-paths and concatenate them to an I^{th} shortest path with $J \leq I \leq K$. However, if a constrained shortest path problem is considered, as in our case the additional condition of paths being loop-less, this Optimality Principle can not be applied anymore. Although a K^{th} shortest loop-less path consists of J^{th} shortest loop-less sub-paths, J might be larger than K ([MPS98]). Thus, we implemented a version of Yen's K-shortest-loop-less-path algorithm as described by Brander and Sinclair ([BS95]). Yen's algorithm is known to have a computational complexity of $O(Kn^3)$, where $O(n^2)$ is due to the single shortest path calculation. If hop limits are considered, the running time increases as will be discussed below.

Implementation of Yen's algorithm

The algorithm stores the found K shortest paths in a *result list*. From the last path entry of the result list, a couple of candidate paths are computed and stored in a *candidate heap*. For each node in the last found shortest path, the *root path* is defined as the sub-path from the source node to the currently considered node, the so-called *root node*. Edges and nodes in the supply graph are labeled to forbid routings through these components, and a so-called *spur path* is computed from the current root node to the target node. With labeling of former root nodes a cycling of routing paths is prevented. Labeling of edges prevent the shortest path calculation to follow a former shortest path, i.e., copy an already found path. The root path and the spur path are concatenated and inserted into the candidate heap. After the spur path computation for each root path of the last entry of the result list, the shortest path from the

candidate heap becomes the next shortest path of the result list. This procedure is iterated until either K shortest paths are found or the candidate heap is empty. In the latter case, it is not possible to find K shortest paths. A more detailed description is presented by Algorithm 2.

Notes on the algorithm

- The algorithm is designed to fill the *result list* with shortest paths. The return value is the number of actually found shortest paths which might be less than the desired K.
- All shortest path computations are performed by an implementation of Dijkstra's shortest path algorithm (e.g., see [Kor02]).
- The **repeat-until**-loop (lines 17-22) is necessary, because it is possible that the same candidate path is inserted several times into the *candidate heap*.
- The worst case computational complexity is given by

  1. the K-loop (line 6) [$O(K)$],
  2. the **for all**-loop (line 7) [$O(n)$, since a loop-less path might pass each network node at most once], and
  3. the shortest path calculation (line 9) [$O(n^2)$].

  Thus, the overall worst case computational complexity is $O(Kn^3)$.
- We pass on a proof of the algorithm's correctness.

If path length restrictions should be additionally considered, the computational complexity will be increased. This is due to the fact that even paths that exceed the given hop limit might provide the basic *root path* to form a new next shortest path that respects the hop limit. Hence, it is necessary to calculate the shortest paths as before and insert paths into the *result list*, even if they exceed the hop limit. Additionally, the number of paths that respect the given hop limit is counted. If K such paths are found, all other paths have to be removed from the *result list*. The worst case computational complexity occurs, if there are less than K paths in the network respecting the length restriction. In this case, all simple paths between source and target node are enumerated until detection that no more paths exist. Thus, the running time of the algorithm depends on the number of simple paths between source and target in the network which is exponentially dependent on the number of nodes in the network. To bound the algorithmic running time we introduce limits on the maximum number of iterations of the former K-loop of Algorithm 2. The drawback of this decision is that the algorithm might not be exact anymore. Although it can be unlikely that further paths respect a given hop limit after a certain number of iterations producing only paths which violated this bound, it cannot be completely excluded.

Algorithm 2 Adaption of Yen's KSP algorithm without path length restrictions

INPUT: graph $G = (V, E)$
source and target nodes $(s, t) \in V \times V$
empty *result list*
K – number of paths to be found
OUTPUT: J – number of actually found paths
result list containing J shortest s-t-paths

compute shortest s-t-path
if no path found **then**
  return 0
end if
store the found path in the *result list*
for $i = 1$ to K **do**
  for all *root path* of the i^{th} shortest path **do**
    label edges that are used as next edge by a previously found j^{th} shortest path $(1 \leq j \leq i)$ with an equal root path
    calculate *spur path* as shortest path from the *root node* to t (the *spur path* must not contain any labeled nodes or edges)
    if shortest path found **then**
      concatenate *root path* and *spur path*
      insert new path into the *candidate heap*
    end if
    remove all edge labels
    label the current *root node*
  end for
  repeat
    if *candidate heap* is empty **then**
      return i
    end if
    new path = shortest path from *candidate heap*
  until *new path* not already contained in *result list*
  insert *new path* into *result list*
  remove all node labels
end for
return K

5.2 Branch-and-price approximation

The CPLEX software used to solve the linear and (mixed-) integer linear programs does currently not support branch-and-price algorithms ([SA05]). Although branch-and-bound for integer linear programs and column-generation for linear programs is supported, the integration of these solution methods into a branch-and-price framework is not possible. In this section, we describe variations of the algorithm presented in Chapter 4 to solve the reconfiguration problems without an integrated branch-and-price framework. We discuss several techniques and the quality of possible solutions.

Remark 5.1. *The column-generation process to solve optimization problems of the* ***MinNoC*** *problem class is divided in two parts. There is a feasibility check to generate an initial set of path variables for further optimization or to determine infeasibility (see Chapter 4). We will refer to the different subproblems as* feasibility problem *and* original problem*. In other words, the* original problem *formulation is the LP relaxation of a* ***MinNoC*** *problem instance.*

5.2.1 Column-generation in the root node only

In the first solution approach, the branch-and-bound and the column-generation part of the algorithm were separated and used independently one after the other.

Description

Instead of an integrated branch-and-price procedure, the algorithm uses column-generation to determine potentially missing path variables only at the beginning. Therefore, we call this approach *price-and-branch*. Algorithm 3 sketches the applied solution procedure.

First the problem is initialized with network data, such as network topology and link capacities, and with the initial routing. The original routing paths are used to generate the initial set of path flow variables. Lines 2-5 are only executed if reconfiguration

Algorithm 3 1^{st} price-and-branch approach

```
1: initialize path variables with given routing
2: solve feasibility problem of LP relaxation with column-generation
3: if problem is infeasible then
4:     return
5: end if
6: solve LP relaxation of original problem with column-generation
7: if solution is fractional then
8:     branch-and-bound
9: end if
```

problems of the **MinNoC** problem class are considered. In this case, the LP relaxation is transformed into the feasibility problem formulation. Missing path variables are

identified by column-generation to obtain a feasible (fractional) routing reconfiguration. In case of infeasibility, there is also a certificate that the original integer linear program formulation is infeasible, since the LP relaxation is a generalization of the IP. In case of feasibility, the LP relaxation of the original problem is solved.

If the reconfiguration problem is one of the **BoundNoC** problem class, the algorithm starts with this LP relaxation. Again, column-generation is used to identify missing path variables which could improve the solution of the original problem. If the optimal solution is fractional (see Remark 4.1), a branch-and-bound procedure tries to find a feasible integer solution within the given set of variables. No further variables are generated at this stage of the algorithm.

Discussion

Algorithm 3 can be interpreted as a simplification of the solution process proposed in Chapter 4. Column-generation is only applied in the root node of the branch-and-bound tree. Therefore, in some cases it is not determinable whether a certain reconfiguration result is optimal, or whether a problem instance is feasible or not. Nevertheless, in some cases even the simplified version of the algorithm allows for assertions about optimality and feasibility.

- Suppose that the algorithm detects infeasibility when solving the feasibility problem (line 3). In this case it is proven that the optimization problem has no feasible solution.

- In many cases, there exists a trivial lower bound for the optimization problems. Single connection clearing, for example, requires rerouting at least all commodities which are currently configured to use the connection that should be dismantled. If there is a lower bound, it can be used to estimate the quality of a solution. If a solution is found whose objective value meets the lower bound, it is obviously an optimal solution and the application of the simplified algorithm was sufficient.

- The optimal solution of the LP relaxation provides another lower bound for the branch-and-bound algorithm. Again, an integer solution with an objective value equal to this lower bound is a certificate for an optimal solution for the integer linear program.

Other results are more difficult to handle. In particular, there are two main problems during the algorithm's branch-and-bound part.

- Suppose that the algorithm finds a feasible solution which is not as good as a computed lower bound. Without additional information it is not determinable whether the solution is optimal or not. There may be missing path variables which would have been generated in the column-generation process at a node in the branch-and-bound tree of an exact branch-and-price algorithm, but not

in the root node. The found solution might be optimal or an arbitrary integer solution.

- The worst case is the situation in which the branch-and-bound procedure does not find a feasible integer solution. The branch-and-bound process is only started if an optimal fractional solution is found. As before, performing column-generation only at the root node of the branch-and-bound tree makes it impossible to detect missing paths when restricting the solution space in the course of the branch-and-bound procedure. Again, there is no proof whether an integral solution exists or not.

5.2.2 Iterated branch-and-bound plus delayed column-generation

In fact, there is only one part of the branch-and-price algorithm that is not supported by the CPLEX optimization software, namely adding variables to the LP relaxation of a currently regarded branch-and-bound node. The pricing within a branch-and-bound node can still be performed. Thus, it is possible to circumvent the CPLEX branch-and-price problems and come to the same result as a branch-and-price algorithm.

Description

In each node of the branch-and-bound tree the pricing problem of the column-generation approach is solved. Instead of adding the primal variables directly to the variable set of RMP, the variables are stored in a list and added after the execution of the complete branch-and-bound procedure. Afterwards the process is iterated with the new variables. A detailed description is given by Algorithm 4.

As before, the execution of the feasibility problem solution depends on the kind of reconfiguration problem. For **BoundNoC** reconfiguration tasks, these parts are ignored, since the problems are always feasible but not necessarily optimally solved.

Discussion

The quality of a solution found by Algorithm 4 depends on the number of iterations of the **repeat-until**-loop (lines 8-20). If the branch-and-bound procedure is iteratively executed until no missing primal path variables could be identified in the course of the current iteration, the result of the algorithm is exact. If no integer solution is found, there is none at all. Otherwise, the best found integer solution is also an optimal solution for the considered problem.

However, although exact solutions are desired, it is usually not possible to use Algorithm 4 to solve a MIP or IP to optimality. The solution process for a single branch-and-bound process to solve an integer or mixed-integer linear program with several hundred variables to optimality can take a couple of hours, even if state of the art optimization software (ILOG CPLEX 9.1) is applied. The running time of an iterated branch-and-bound procedure where the number of iterations depends on the number

Algorithm 4 Iterated branch-and-price approach

```
1: initialize RMP path variables with initial routing
2: solve feasibility problem of LP relaxation with column-generation
3: if problem is infeasible then
4:    return
5: end if
6: solve LP relaxation of original problem with column-generation
7: if solution is fractional then
8:    repeat
9:       start branch-and-bound procedure
10:      for all nodes in the branch-and-bound tree do
11:         initialize node LP relaxation
12:         solve feasibility problem for current node LP relaxation
13:         solve pricing problem of the feasibility LP
14:         store missing primal path variables in path list
15:         solve original problem formulation for current node LP relaxation
16:         solve pricing problem of original problem
17:         store missing primal path variables in path list
18:      end for
19:      add paths from path list to the RMP
20:   until no new path variables generated or iteration bound reached
21: end if
```

of primal variables generated by a column-generation process cannot be reasonably estimated. Thus, this algorithm is not useful to solve integer or mixed-integer linear programs exactly. However, it can be used as heuristic algorithm to add further path variables to the set of initial paths. The corresponding path variables are not accidently generated, but they turn out to be useful in the course of the branch-and-bound process. Thus, this algorithm is an improvement of Algorithm 3. Solutions of Algorithm 4 are at least as good as solutions of the former algorithm.

For practical purposes, we introduce limits on both the number of iterations of the branch-and-bound procedure and the solution time spent to each of the iterations. Although this algorithm does not always guarantee an exact solution, it performs well if applied to real world reconfiguration problems, as discussed in Chapter 6.

The properties of the discussion of the Algorithm 3 w.r.t. optimality and feasibility of a reconfiguration problem (see page 63) are the same for Algorithm 4. There are only some additional remarks:

- Suppose that the algorithm delivers a feasible solution that is not as good as a formerly computed lower bound. If the algorithm was not stopped because of a limit on the number of iterations or on computational time, the obtained solution is as good as the best solution found by an integrated branch-and-price framework. If the pricing of the branch-and-price is exact, the solution is optimal.

- If the iteration of the branch-and-bound procedure does not provide a feasible integer or mixed-integer solution in the end, it is in most cases not determinable whether the optimization problem has a feasible solution at all. Only if the whole branch-and-bound tree has been completely processed, it is proven that there is no feasible solution (again, the pricing has to be exact). Otherwise, although the existence of a feasible solution might be unlikely, it cannot be excluded.

5.2.3 Heuristic solution approach

We also compared the previous approaches to a simple one. Since the formerly described branch-and-price algorithm is very complex, we implemented a heuristic solution approach to test whether it is possible to achieve reasonable results without the branch-and-price approach.

Description

The applied algorithm initializes path variables of the original routing and heuristically generates further path variables. Afterwards, a MIP solver is started to find the best integer (mixed-integer) solution for the set of initialized variables. There is no further column-generation process to generate path variables throughout the solution process of the MIP solver. The complete algorithm is given by Algorithm 5.

Algorithm 5 heuristic solution approach

1: initialize RMP path variables with initial routing
2: generate further path variables heuristically
3: start MIP solver

Discussion

Results of this algorithm are difficult to evaluate. If feasible solutions are found, they can only be compared to bounds obtained from the problem description. For instance, optimality for the connection clearing scenario can only be guaranteed if the trivial lower bound is met (see discussion of Algorithm 3 on page 63).

If no solution is found, it is not clear whether a feasible solution exists. The only possibility to guarantee exact results for this heuristic algorithm is to enumerate all feasible path flow variables which is practically not possible.

Chapter 6

Computational Results

For testing purposes we implemented four variants of the formerly described algorithm to evaluate the practical applicability of the solution procedure to different reconfiguration scenarios. In particular, these tested reconfiguration tasks were connection clearing, adding of new demands, shortening initial routing paths, and link load reduction. As test instances we used different real world telecommunication networks with different initial routings. All four reconfiguration scenarios were tested for all these networks. This Chapter serves as documentation of the performed tests and as a summary of the obtained results.

First, there is a general description of the test networks and the generation of initial routings. Afterwards, we focus on the implementation and results for the reconfiguration tasks.

Remark 6.1. *The calculations described in this chapter were performed on work stations of the* Optimization Department *of ZIB*[1]*. All computations were made single threaded on Dell Precision 650 PCs with 2 GB of main memory and Intel Pentium 4 CPUs with 3.0 - 3.2 GHz running SuSE Linux 9.3.*

6.1 Testing data

The choice of appropriate testing data for the reconfiguration tasks consists of two steps. First, the basic network topologies and demand specifications must be selected. Furthermore, initial routings for the network instances must be found. The following sections describe the choices we made for our testing procedures.

6.1.1 Network selection

From a collection of more than 70 real world telecommunication networks provided by different network operators we have chosen a set of eight instances for the proposed tests. The networks were originally specified for network dimensioning planing (see

[1] `http://www.zib.de`

Chapter 1). The specifications consist of potential network nodes and potential connections with different possibilities to install hardware at locations and capacities for connections. Additionally, demand specifications are given.

The eight test instances are chosen such that a reasonable spectrum of different topologies can be evaluated. Further selection criteria are the possibility for survivable routing definitions. All of the eight instances allow feasible routings if diversification with a diversification factor of $\frac{1}{2}$ for the majority of demands is specified. The original demand definitions consist of source and target node as well as demand value and diversification parameter. The demand value is of arbitrary integer size. As proposed in Section 3.3.1, for the applied mechanism of counting changes between different routings, it is indispensable to route small single demands through the network. Therefore, we splitted the demands into sets of single demands with demands value 1 or 2 (1 if the diversification parameter is 1, 2 for a diversification factor of $\frac{1}{2}$).

The operating states for all networks were defined to be single failures only. However, we did not declare any network component to be fail-safe, i.e., each component of each network might possibly fail. The direct consequence for feasible routings is that if there is a demand with demand value 2 and diversification factor $\frac{1}{2}$, a feasible routing for this demand consists of two routing paths completely disjoint except for the terminal nodes.

6.1.2 Initial routings

The chosen network instances provide only a basic topology and demand specification. For the reconfiguration tasks, there has to be a final node and connection selection and an initial routing to be reconfigured afterwards. For this purpose we used the DISCNET network planning tool from Roland Wessäly ([Wes00]). It calculated a cost optimized network topology for each network and an initial routing for this topology satisfying the demand and diversification restrictions. Additionally, the routing was restricted to use at most 90% of the available link capacity on each connection. The idea behind this decision was on the one hand to obtain enough free routing capacity for reconfiguration. On the other hand, an idealized real world operating scenario should be simulated.

Furthermore, we calculated a second initial topology and routing for each of the eight networks allowing only 80% maximum initial link load. Originally, this second initialization was supposed to be a backup configuration for networks with few reconfiguration possibilities when initialized with a maximum link load of 90%. The thought was: the more free capacity is provided, the more reconfiguration is possible. However, the cost optimization function for the initial topology and routing computation lead to completely different topologies, hardware installments and initial routing paths. Hence, we are equipped with two independent initial routings for each of the eight basic network instances.

INITIAL TEST NETWORK CONFIGURATIONS								
			80% LINK LOAD			90% LINK LOAD		
NETW.	NR. OF DEMANDS	DEMAND VALUES	NR. OF NODES	NR. OF LINKS	OPT. GAP [%]	NR. OF NODES	NR. OF LINKS	OPT. GAP [%]
p01	428	846	17	38	37.3	17	47	37.1
p07	774	1548	15	31	35.0	15	77	61.7
p09	928	1856	19	148	82.1	19	148	80.9
p20	862	1724	24	94	43.9	24	94	44.2
p30	2692	4153	46	61	60.6	44	48	21.8
p44	8635	9473	20	23	00.0	19	23	01.4
p45	2643	3669	19	22	00.0	19	22	01.0
p69	3024	4032	56	69	29.6	56	69	27.6

Table 6.1: Overview on the used test instances. In the terms of the mathematical model, the number of demands corresponds to $|\mathcal{K}|$. Similarly, the demand values are defined as $\sum_{k\in\mathcal{K}} d_k$. The OPT. GAP column states the gap between the best found solution and the computed lower bound corresponding to an optimal fractional solution w.r.t. the objective function minimizing network costs.

6.1.3 Test instances

After the explanation of the generation process of the test networks, we focus on the resulting instances. Table 6.1 gives a first overview on the test networks.

As a consequence of the different maximum initial link load bounds, the structures of the computed initial configurations belonging to the same networks often differ in the number of nodes, the number of links, the hardware installed at nodes, and the capacity installed on links. Figure 6.1 shows the differences between the initial network topologies for the p07 network. The 90% configuration contains more than twice the number of connections than the 80% configuration. This is mainly due to a large number of parallel edges.

The computation time for the initial network configurations was restricted to one hour for each instance. Depending on the size of the initial networks and the number of demands, this restriction led to different qualities of the resulting network configurations. In Table 6.1, the gap between the computed lower bound and the best found solution is listed for each instance. The gap for both initial configurations of p09 was greater than 80%. This resulted in highly intermeshed network structures. In contrast, for p44 and p45 optimal or near-optimal solutions w.r.t. the cost function could be found. The results are more ring-like structured networks as can be seen in Figure 6.2 for p45.

For p30, the initial computation limit of one hour was not sufficient to find initial network topologies. Hence, we increased the time limit to two hours. With this change, it was possible to obtain initial network configurations and routings. Although the network p69 is even larger w.r.t. the number of nodes and the number of links than p30, The time limit of one hour was sufficient for both initial configurations of p69. This is due to the distribution of the demand which is more evenly divided throughout the network than in the case of p30. Figure 6.3 shows the different demand distributions for these networks.

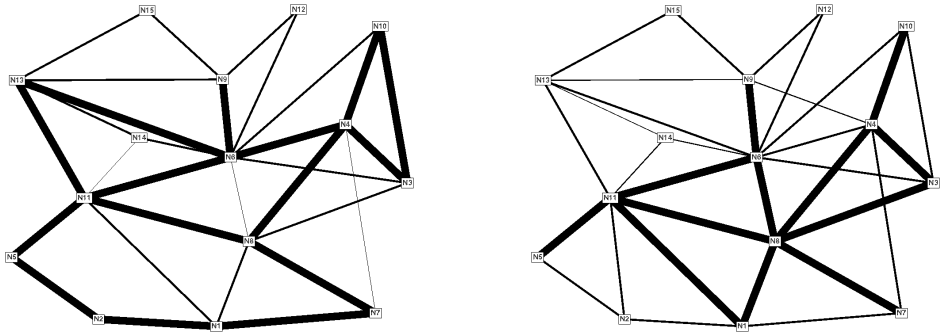

Figure 6.1: Network p07 with initially 80% maximum link load in the left part of the figure and 90% maximum link load on the right hand side. The thickness of connections between network nodes corresponds to routing capacity of the connections between these nodes. Note that a single line might symbolize a set of parallel supply edges. The thicker a line is drawn, the more capacity is installed on the set of parallel edges, and the more demand can be routed over this set of edges.

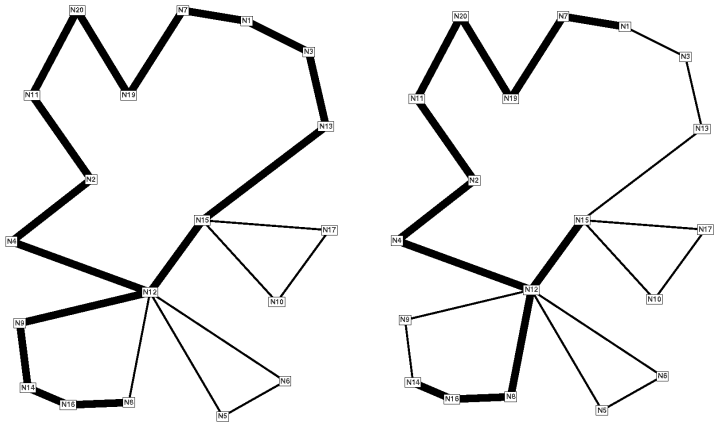

Figure 6.2: The optimal initial network design for p45 (left: 80% maximum initial link load, right 90%). Although the topologies are similar, there are differences in the installed routing capacities on the networks' edges. The ring-like structures will provide difficulties for reconfiguration, particularly, if diversification is respected.

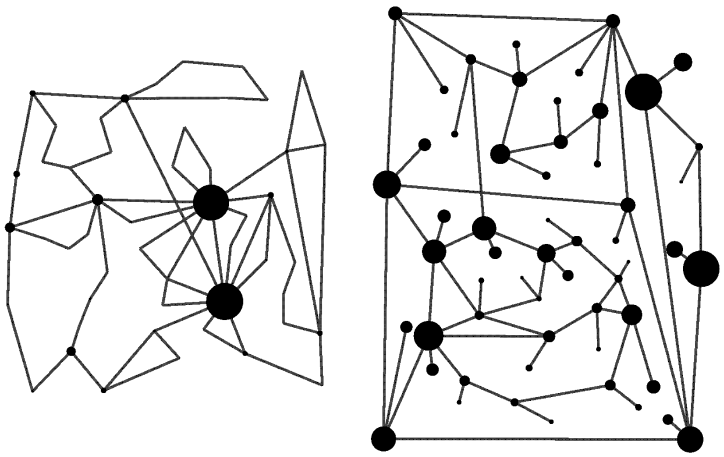

Figure 6.3: Differences in the demand emanating from nodes of different networks. On the left hand side: p30, on the right hand side: p69. The size of the network node increases with the number of demands starting or terminating at this node. The main demand is concentrated on few nodes in the p30 network, whereas the demand for the p69 network is more evenly distributed over the complete network.

6.2 Reconfiguration results

Each of the implemented reconfiguration scenarios was tested on all of the 16 different problem instances.

6.2.1 Connection clearing

As described in Chapter 2, this scenario deals with the task of removing a link from the telecommunication network. The demands currently routed over this link must be rerouted, such that all demand and diversification restrictions remain satisfied. To be statistically independent of results obtained with good or bad random choices of network connections to be removed, we implemented a single connection clearing test and tried to remove each link of each network in sequence.

Because of the differences in topology, routing capacities and total demand for the different test networks, the reconfiguration success ranged from "no connection can be cleared" to "almost each link can be removed". Mainly because of the complete impossibility to remove a single connection in certain networks, we modified the test conditions slightly and re-ran the complete test procedure. For this purposes we redefined the diversification parameter to 1 for all demands, independently of their demand value. As consequence, routings become feasible although they might contain several routing paths for the same demand which pass through the same network components. Routing paths for the same commodity need not to be disjoint w.r.t. operating

p01							
		DIVERS.	NO DIVERS.			DIVERS.	NO DIVERS.
LINK	INITIAL FLOW	CHANGES	CHANGES	LINK	INITIAL FLOW	CHANGES	CHANGES
L1-N17-N15	14	14	14	L27-N6-N8	51	51	51
L2-N4-N5	51	[$\gamma = 2$]	[$\gamma = 2$]	L28-N7-N8	51	51	51
L3-N5-N6	51	[$\gamma = 2$]	[$\gamma = 2$]	L29-N3-N8	51	52	51
L4-N6-N7	51	51	51	L30-N5-N7	51	[$\gamma = 2$]	[$\gamma = 2$]
L5-N7-N12	51	51	51	L31-N4-N7	51	51	51
L6-N12-N16	50	57	50	L33-N8-N9	51	[$\gamma = 1$]	51
L7-N16-N15	51	[$\gamma = 5$]	52	L36-N11-N9	51	51	51
L9-N16-N13	50	50	50	L37-N8-N10	51	[$\gamma = 42$]	[$\gamma = 11$]
L10-N13-N14	51	[$\gamma = 12$]	[$\gamma = 4$]	L39-N7-N10	51	[$\gamma = 42$]	[$\gamma = 11$]
L12-N17-N1	20	20	20	L40-N7-N9	51	60	60
L14-N12-N11	7	7	7	L49-N15-N14	50	[$\gamma = 14$]	50
L15-N11-N13	49	[$\gamma = 4$]	49	L54-N2-N15	51	51	51
L16-N7-N11	51	51	51	L55-N15-N12	51	51	51
L18-N12-N6	51	51	51	L56-N2-N12	51	51	51
L19-N5-N12	51	[$\gamma = 2$]	[$\gamma = 2$]	L57-N1-N3	50	50	50
L21-N3-N6	51	51	51	L58-N1-N4	50	50	50
L23-N17-N2	7	7	7	L59-N1-N15	51	51	51
L25-N2-N7	51	51	51	L60-N2-N3	50	50	50
L26-N4-N6	51	51	51	L61-N3-N4	51	51	51

Table 6.2: Reconfiguration results for the connection clearing scenario applied to p01 with the 80% maximum initial link load configuration. In addition to the link identifier and the initial number of routing paths that pass through this link, there is information on the minimum number of routing changes for both the original problem and the problem formulation with a relaxed diversification constraint. The number of changes contains both the initial flow over the cleared connection and additional reassignments for routing paths that do not pass through this link originally. In the case of infeasibility, the objective value γ of the optimal solution of the feasibility LP is given. It states the amount of additional capacity to be installed on at least one connection to obtain a feasible routing.

states anymore. Thus, this problem formulation is a relaxation of the former problem specification, and the results must be at least as good as the results of the former computations.

As exemplary instance of a network with many reconfiguration possibilities for this scenario, we present the results for p01 for the 80% maximum initial link load configuration (Table 6.2). The tables for the other test instances can be found in Appendix A.1.

Results for p01 with 80% maximum link load

The network contains 38 connections. If diversification is respected, there are solutions for 27 links. Without diversification, the number of connection that can be cleared is 31. The results are reasonable in so far as passing on the diversification constraint tends to improve the solution. However, it can be seen that in most cases in which a solution is found, there is no improvement. In these cases, it is sufficient to reassign only the initial flow that is routed over a connection to be cleared. If no solutions were found, infeasibility was detected while solving the feasibility LP. Hence, there is no link with a feasible fractional routing but no integer solution during the branch-and-bound procedure. In fact, there are only two links (L7-N16-N15, L40-N7-N9),

for which the branch-and-bound procedure was necessary at all (in both cases only for the relaxed formulation without diversification). All other solutions found by the column-generation procedure in the root node are integral from the beginning. For L7-N16-N15, the optimal fractional solution has an objective value of 51.5. As can be seen in Table 6.2, the best found integer solution has an objective value of 52. Hence, the solution found by the branch-and-bound procedure is an optimal solution. There is a similar situation for L40-N7-N9 for which the optimal fractional solution has an objective value of 59.5, such that the integer solution with objective value of 60 is optimal.

General results

As can be seen in the tables for this reconfiguration scenario, the results for p01 cannot be generalized. In general, we tried to clear 2030 connections in 16 networks altogether. Solutions were found in 1341 cases. However, depending on the network structure, the total demand and distribution of demands in the networks, results are different w.r.t. the reconfiguration possibilities. There are instances, like p01, with large reconfiguration amounts both in the case of 80% and 90% maximum initial link load, whereas other instances like p45 do not provide any connection clearing possibilities at all, independently of initial configuration and diversification relaxation. Network p07 allowed for little reconfiguration in the 80% initialization and for large reconfiguration in the 90% initialization case. As mentioned in Section 6.1.3, this can be traced back to the different qualities of the initial solutions. The large gap between best found solution and lower bound led to a large number of parallel edges in the 90% initialization case. Since the resulting network contains many edges with comparatively little initial flow, it is easier to remove a single link than in the case of the 80% initial link load configuration with fewer links and more initial flow on the links.

Altogether, there are five links for which the branch-and-bound procedure had to be started to obtain an optimal integer solution from optimal fractional ones. Besides the two links from p01, there are two other links in p69 with this property. In the 90% maximum link load configuration, the clearing of the connections L25-N21-N25 and L27-N21-N47 provide optimal fractional solutions of 212.5 and 208.5, respectively. The achieved integer solutions with values 213 and 209 are therefore optimal solutions. The only exception to this huge number of provable exact solutions is L25-N21-N25 in the 90% configuration of p69. The optimal fractional solution has an objective value of 249.5. The best found integer solution has a value of 256. Since the branching rule of dichotomy branching combined with the K-shortest-path pricing procedure might lead to imprecise results (see Section 4.3.1), optimality can not be guaranteed in this special case.

We logged the solution time for each instance. To keep the results tables small and concise, we passed on the solution time listing for this reconfiguration scenario. However, to give the reader a feeling of the approximate expenditure of time, we add the following observations. The time spent on the solution of the **MinNoC** LP relaxation

(including the feasibility LP) were few seconds and in more than 95% less than one second. The most time was spent to initialize the network and demand data with the original routing for all demands. This process lasted between several seconds and few minutes. Therefore a single connection clearing scenario for the small and medium sized real world telecommunication network we consider, can mostly be solved exactly within seconds or few minutes.

Conclusion

Since all considered operating states for all networks are single-failures, the column-generation process is exact (see Section 4.2.3). In general, optimal solutions of the **MinNoC** LP relaxation tend to be integral for this reconfiguration task. In 99.75% of all tested connection clearings, there was no need to start the branch-and-bound procedure of the branch-and-price algorithm. With the implementation of Algorithm 4, we obtained exact results in 99.95% of tested reconfiguration tasks for the connection clearing scenario. If no solutions could be found, there is a certificate that there are no feasible solutions anyway. In the cases where feasible solutions were found, the best found solution is almost always an optimal solution. For only one case, the precision of the obtained result is not clear. Neither optimality nor impreciseness can be concluded by the obtained solution and logged data.

The relaxation of the diversification constraints does not entail the strong improvement of results for which we hoped. Although several solutions could be improved, no solutions could be obtained for the networks without any reconfiguration possibilities (p44, p45). Therefore, we passed on this additional testing procedure for the other **MinNoC** reconfiguration tasks.

6.2.2 Connection clearing (heuristically)

In this section, we describe the results to the connection clearing scenario obtained with the heuristic Algorithm 5 described in Section 5.2.3.

The generation of additionally path variables was performed as follows:

Definition 6.1. *Let* $F_e = \sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{Q}_k : e \in P} p_k \cdot f_k(P) \in \mathbb{N}$ *be the flow of edge* e *and* $C_e \in \mathbb{N}$ *be the capacity of edge* e *with* $F_e \leq C_e$ *for each* $e \in E$*. Define* $L_e := \frac{F_e}{C_e} \in [0,1]$ *as link load of edge* e*. Let* $K_e \in [0,1]$ *denote the weight for each edge* $e \in E$*.*

We used a K-shortest-path-algorithm with two different cost functions for connections in the networks to create alternative paths:

1. $K_e = 1 \ \ \forall e \in E$.
2. $K_e = L_e^2 \ \ \forall e \in E$.

The first criterion is a simple shortest path w.r.t. the number of links in the network, whereas the second one privileges edges that have a small link load in the initial

routing. We have tested different combinations of these two criteria (one of them or both concurrently) with different values of $K \in \{3, 5, 10, 15, 20\}$ (in the case of $K = 20$ with both criteria applied, up to 40 path variables are generated for each single commodity).

Results

We could not identify any behavioral rule for solution results on which a general optimal parameter configuration could be based to yield always best results for the heuristic algorithm. There are connections in networks for which it was sufficient to generate shortest path w.r.t. the number of supply edges. Other instances provided best solutions only if shortest paths w.r.t. link load were generated. Finally there were instances which required the application of both criteria to obtain best results. Additionally, the number of paths to be generated to achieve best results covered the complete spectrum between 3 and 20.

The quality of obtained solutions could only be measured in comparison with the trivial lower bound of the number of demands that are initially routed over the connection to be cleared. If no solution is heuristically found, it is not clear whether a feasible solution exists at all.

In comparison to the almost always exact solutions of the former section, the heuristic algorithm provided reasonable results in most cases:

- For less than 1% of all tested connection clearing scenarios, the heuristic did not find a feasible solution, although existing.
- For about 95% of all tested connection clearing scenarios, the heuristic found the optimal solution.

Conclusion

With the applied heuristic approach it was not possible to identify a parameter setting — beyond the initialization of as many variables as possible — to reliably generate good additional path variables. However, initializing huge amounts of variables for each single commodity leads inevitably to enlarged integer and mixed-integer linear programs and to elongated running times for the solution process of the MIP solver. Compared to the implementation of Algorithm 4, worse results were obtained in longer running times. Therefore, we passed on pursuing this heuristic approach furthermore.

6.2.3 Adding new demands

This scenario deals with embedding new demand specifications into an operating network. For all network configurations, we increased the already given demand specifications step by step to observe the impact to the initial routing if new demands are additionally routed through the network. We increased the demands in certain steps and afterwards, tried to find feasible routings for all demands, old and new ones. As

in the former reconfiguration scenario, we applied an implementation of the algorithm for the **MinNoC** models to keep as many paths of the original routing as possible. The scenario was tested on all eight networks with both initial configurations.

The remainder of this scenario's documentation is organized as follows. First, we describe the initialization of new demands, then we focus on the results of p01 with 80% maximum initial link load in more detail. Afterwards, we give a more general overview on the results for the remaining networks, and finally, we point out the conclusions drawn from the results of this reconfiguration scenario.

Demand generation

Instead of generating randomly new demands for the networks, we decided to increase the original demands. In Section 6.1.1, we described the procedure of splitting demands of the original demand specification with arbitrary integer values into sets of single demands with demand value 1 or 2. For the reconfiguration task of including new demands into the network, this procedure was slightly modified. Before splitting the demands, the values were increased by $10\%, 20\%, .., 50\%$. All obtained fractional values were rounded to the next lower integer value. Afterwards, the splitting of demands into demands with value 1 or 2 was preceded as before. Due to the rounding procedure, the percentage levels of demand enhancement are only upper bounds. The computed initial routings could further be used. The combination of old and new demand specifications leads to the situation in which only a subset of all demands has an initial solution.

Before starting the branch-and-price routine, feasible routing paths are computed for the new demands. These routing paths are generated such that for each new demand/commodity k there are exactly $\frac{d_k}{p_k}$ routing paths which are disjoint w.r.t. operating states. This initialization is necessary, because the feasibility part of Algorithm 4 assumes locally feasible routings, i.e., enough feasible routing paths for each demand. These locally feasible routing paths might altogether violate global capacity restrictions. In this case, the column-generation procedure tries to identify commodities for which the generation of alternative routing paths could lead to globally feasible routings. If not all commodities had locally feasible routing paths, the algorithm would not act correctly. This algorithmic design decision leads to shorter running times in the branch-and-price part of the algorithm compensated by an elongated initialization phase. The results for this scenario can be found in Table 6.3 for the networks with 80% maximum initial link load. The results for the 90% configurations can be found in Appendix A.2.

Results for p01 with 80% maximum initial link load

Embedding new demands into this network does not have any impact on the original demand routing. With the implemented procedure of adding new demands, it was possible to integrate 176 new routing paths into the operating network, which corre-

ADDING NEW DEMANDS – 80% MAX. INITIAL LINK LOAD								
NET	MAX NEW [%]	DEMANDS	VALUES	ACTUAL NEW [%]	ACTUAL NEW [ABS.]	OPT. FRAC. SOLUTION	CHANGES	TIME
p01	10	436	862	1.89	16	16	16	00:00:00
	20	458	906	7.09	60	60	60	00:00:00
	30	516	1022	20.80	176	176	176	00:00:00
	40	538	1066	26.00	220	[$\gamma = 2$]	—	00:00:00
	50	637	1264	49.41	418	[$\gamma = 11$]	—	00:00:00
p07	10	829	1658	7.11	110	110	110	00:00:00
	20	909	1818	17.44	270	270	270	00:00:01
	30	988	1976	27.65	428	[$\gamma = 1$]	—	00:00:01
	40	1068	2136	37.98	588	[$\gamma = 9.2$]	—	00:00:01
	50	1160	2320	49.87	772	[$\gamma = 19$]	—	00:00:01
p09	10	990	1980	6.68	124	124	124	00:00:00
	20	1081	2162	16.49	306	306	306	00:00:02
	30	1175	2350	26.62	494	494	494	00:00:04
	40	1266	2532	36.42	676	676	676	00:00:07
	50	1391	2782	49.89	926	926	926	00:00:13
p20	10	897	1794	4.06	70	70	70	00:00:00
	20	972	1944	12.76	220	220	220	00:00:00
	30	1067	2134	23.78	410	410	410	00:00:00
	40	1142	2284	32.48	560	560	560	00:00:02
	50	1292	2584	49.88	860	860	860	00:00:08
p30	10	2823	4354	4.84	201	201	201	00:00:00
	20	3059	4704	13.27	551	551	551	00:00:02
	30	3273	5028	21.07	875	875	875	00:00:05
	40	3592	5533	33.23	1380	1380	1380	00:00:07
	50	3894	5993	44.31	1840	1840	1840	00:00:10
p44	10	8782	9672	2.10	199	199	199	00:00:00
	20	9002	9972	5.27	499	499	499	00:00:01
	30	9221	10276	8.48	803	803	803	00:00:02
	40	9443	10578	11.66	1105	1120	1120	00:00:05
	50	9765	11022	16.35	1549	1661	1661	00:00:12
	60	9916	11225	18.49	1752	1752	1752	00:00:00
p45	10	2843	3950	7.66	281	281	281	00:00:00
	20	3086	4295	17.06	626	626	626	00:00:00
	30	3331	4638	26.41	969	[$\gamma = 3$]	—	00:00:00
	40	3574	4983	35.81	1314	[$\gamma = 43$]	—	00:00:00
	50	3874	5413	47.53	1744	[$\gamma = 87$]	—	00:00:00
p69	10	3211	4221	4.69	189	189	189	00:00:00
	20	3475	4544	12.70	512	512	512	00:00:00
	30	3724	4842	20.09	810	810	810	00:00:00
	40	4059	5306	31.60	1274	1274	1274	00:00:04
	50	4403	5790	43.60	1758	[$\gamma = 8.66667$]	—	00:00:04

Table 6.3: Results of the new demands scenario for networks with 80% maximum initial link load. The column beneath the network identifier states the percentage of maximum enhancement. The next two columns describe the number of aggregated demands and the corresponding demand values. The fourth column delivers the actual demand and value enhancement which might diverge from the first column. The OPT. FRAC. SOLUTION column displays either the optimal fractional solution found by the column-generation procedure in the root node of the branch-and-bound tree or the γ-value of the feasibility LP in the case of infeasibility. The CHANGES column states the differences between old and new routing. It has to be mentioned that the new demands are already included into this number. The last column displays the solution time spent on the branch-and-price algorithm only.

sponds to an enhancement of 20.8%. Further demand increasing between 20.8% and 26% might be possible but was not tested.

In most cases, the actual increase in demand values is reasonably smaller than the maximum admissible enhancement. This is due to the original demand distribution which consists of many demands with small demand values. By rounding down the number of demands after little enhancement, not many new demands are left over. However, in the row with a maximum enhancement of 50%, this rate is almost reached. This is due to the fact that almost all original demands had an even demand value, such that rounding down after a demand increasing of 50% did not skip many new demands.

The listing of computation times shows that this reconfiguration task can be resolved quickly. In the case of success, the solution time has been below one second. Since infeasibility for the 40% and 50% demand increasing has been detected while solving the feasibility LP in the root node of the branch-and-bound tree, the spent solution time is comparably small in these cases. Note that the measured time corresponds only to the optimization process, i.e., the computations caused by the branch-and-price algorithm. Like in the connection clearing scenario, the branch-and-bound trees consisted only of the root node for all five demand enhancement levels. Since the solutions of the **MinNoC** LP relaxations have already been integral, there was no need for further branching. Actually, in the first two cases of demand increasing for this network, the initially generated paths for the new demands are feasible and can be integrated into the operating routing without violating any capacity constraint. Only in the case of a demand enhancement by 20.8% alternative routing paths have to be found to solve the feasibility LP in the root node of the branch-and-bound tree. The solution of the feasibility LP for the other two cases leads to $\gamma = 2$ and $\gamma = 11$ respectively. Thus, the capacity on at least one connection had to be increased by 2 to allow a feasible (fractional) routing of all demands.

Interestingly, the other initialization (90% maximum link load) of the p01 network has almost identical results for this scenario, in contrast to the connection clearing task, for which the 90% configuration performed much better than the 80% configuration. The only difference between the configurations in this scenario is the need to change additionally 11 paths of the original routing if the demands of the 90% configuration are increased by 20.8%. Even the feasibility LPs for the last two tested enhancement levels lead to the same values $\gamma = 2$ and $\gamma = 11$.

General Results

All tested networks have a certain reconfiguration contingency. For most networks, it is possible to increase the demand values by 17% to 20%. There are also networks for which a demand enhancement of 50% does not affect the original routing (p20, 80% configuration). Altogether, there are two constellations which delivered fractional optimal solutions of the **MinNoC** LP relaxation. Both occur for network p30 in the 90% configuration. The optimal number of changes is 1385.5 in the case of 33.23% demand enhancement and 1897.5 in the case of 44.31% increasing. Thus, the provided

solutions with values 1386 and 1898 are optimal integer solutions.

The computation time spent on the solution of the branch-and-price algorithm was small for all tested reconfigurations. After the network initialization phase for which the running time depends on the number of network nodes, the number of connections, demands, demand values and demand distribution, all problems were solved within seconds.

Conclusion

As in the former reconfiguration scenario, the column-generation procedure is exact, because all operating states are only single failures. Again, the optimal solution of the **MinNoC** LP relaxation tends to be integral. Only 2 out of 82 (2.44%) reconfiguration tests led to fractional optimal solutions. Even in these two cases, the solutions produced by the branch-and-price algorithm were optimal. Infeasibility was detected for all remaining test constellations within the solution process of the feasibility LP in the root node of the branch-and-bound tree. Therefore, the reconfiguration scenario of adding new demands to an operating network can be solved by the implementation of Algorithm 4 both quickly and exactly.

6.2.4 Shortening initial routing paths

The task for this scenario is to reduce the hop length, i.e., the number of edges of the used routing paths as far as possible. Especially after the enlargement of edge capacities or the installation of new connections between network nodes, a shortening of routing paths can help to save routing capacities and to distribute the link load more equally in the network.

For each commodity, the routing path lengths must not exceed the length of a shortest path for the corresponding commodity by a given number of supply edges.

Initialization procedure

In a preprocessing step, all initial routing paths were excluded if violating the individual hop length restriction of the corresponding commodity. Afterwards, replacement paths were initialized respecting both the individual length restriction and the diversification conditions for the corresponding commodity. Only the global edge capacity restrictions were neglected at this stage of the reconfiguration procedure. The replacement path generation has to be already performed within the initialization phase, because the column-generation based branch-and-price solution approach could not succeed otherwise. This is due to the fact that the column-generation procedure described in Chapter 4 requires an optimal solution of the primal and dual restricted master program to initialize edge weights in the supply graph for the solution of the pricing problem. If routing paths were only excluded without a concurrent generation of replacement paths, RMP could be infeasible.

If it is not possible to generate enough replacement paths within the initialization step, the reconfiguration problem is infeasible and the branch-and-price algorithm will not be started. Otherwise, Algorithm 4 is applied to find best solutions. The only difference to the previous reconfiguration scenarios is the length restriction defined for each commodity during the K-shortest-paths procedure for the column-generation process. As described in Section 5.1, we limited the maximum number of iterations for the application of Yen's KSP algorithm to reduce the running time of the optimization.

Results

The results for the network instances with maximum 90% initial link load can be found in Table 6.4. In Appendix A.3, the results for the 80% configurations are listed. For most network configurations the reconfiguration range is very small. The cost optimized initial solutions tend to use short routing paths. An exception from this rule is network p09 which offers most reconfiguration capability. For the 90% initial configuration, there are initial routing paths which exceed the length of a shortest paths for the corresponding commodity by eight supply edges. The algorithm was able to reroute the demands, such that no routing path exceeded the hop length of a shortest path by more than three links. For the 80% configuration of p09, the result is similar. In contrast, p44 and p45 have no reconfiguration potential at all. For p45 in the 90% configuration there are routing paths exceeding a shortest path by nine supply edges. However, shortening all commodities to eight additional edges to the length of a shortest path is not possible. In three reconfiguration computations, the iteration limit of the KSP algorithm was reached (p09, 90% configuration, $\text{SP}+= 3, 4$ and p20, 80% configuration, $\text{SP}+= 2$). In such a case, it is possible that the pricing problem is not solved exactly, i.e., too few path variables are generated. However, in all three cases, the best solution achieved in the root node of the branch-and-bound tree by the column-generation procedure, meets the trivial lower bound given by the number of paths to be skipped because of hop limit violations. The solution time of these three instances is significantly greater than the solution time of the other instances. In fact, it seems to be directly correlated with the number of early breaks of the KSP procedure. In the case of p09 for example, the number of breaks is roughly quadrupled from $\text{SP}+= 4$ to $\text{SP}+= 3$. The solution time is similarly expanded.

For all 16 initial network configurations it suffices to replace only routing paths exceeding the hop limit. No further changes of the routing are required.

Conclusion

Like the majority of **MinNoC** problems of the former reconfiguration scenarios, the task to shorten routing paths was solved by the branch-and-price algorithm already in the root node of the branch-and-bound tree. Therefore, obtained results are exact if the pricing procedure of the column-generation approach is. It turned out that even in the cases in which the KSP algorithm was abandoned early, the applied implementation of Algorithm 4 provided optimal solutions.

SHORTENING PATH – 90% MAX. INITIAL LINK LOAD							
NETWORK	SP+	SKIPPED INIT. PATHS	PATH LIMIT	LIMIT REACHED	OPT.FRAC. SOLUTION	CHANGES	SOLUTION TIME
p01	5	0	100	0	0	0	00:00:00
	4	1	100	0	1	1	00:00:00
	3	5	100	0	5	5	00:00:00
	2	25	100	0	25	25	00:00:00
	1	INITIALIZER DETECTED INFEASIBILITY					
p07	3	0	100	0	0	0	00:00:00
	2	INITIALIZER DETECTED INFEASIBILITY					
p09	8	0	100	0	0	0	00:00:00
	7	2	100	0	2	2	00:00:00
	6	43	100	0	43	43	00:00:01
	5	60	100	0	60	60	00:00:02
	4	119	100	72	119	119	00:08:53
	3	161	100	313	161	161	00:35:27
	2	INITIALIZER DETECTED INFEASIBILITY					
p20	6	0	100	0	0	0	00:00:00
	5	1	100	0	1	1	00:00:00
	4	3	100	0	3	3	00:00:00
	3	7	100	0	7	7	00:00:00
	2	23	100	0	23	23	00:00:00
	1	INITIALIZER DETECTED INFEASIBILITY					
p30	10	92	100	0	92	92	00:00:00
	9	INITIALIZER DETECTED INFEASIBILITY					
p44	7	0	100	0	0	0	00:00:00
	6	INITIALIZER DETECTED INFEASIBILITY					
p45	9	0	100	0	0	0	00:00:00
	8	INITIALIZER DETECTED INFEASIBILITY					
p69	10	1	100	0	1	1	00:00:00
	9	1	100	0	1	1	00:00:00
	8	INITIALIZER DETECTED INFEASIBILITY					

Table 6.4: Shortening routing paths reconfiguration results for networks with 90% maximum initial link load. The second column states the number of connections which is added to the hop length of a shortest path for each commodity and leads to a hop limit on feasible routing paths for each commodity individually. The next column displays the number of affected initial routing paths. The PATH LIMIT column represents the iteration bound for the K-shortest-paths procedure. The next column states how often this bound was reached and therefore how often the KSP procedure was abandoned before finding K shortest paths. For SP+, we tested all values between 1 and 10 for all instances. To keep the table small, we cut off rows without additional information. If the initializer detected infeasibility for KSP> 1, it detected infeasibility for all smaller values. Conversely, if 0 initial paths violated the hop limit for a certain value of SP+, no path violated the hop limit for greater values.

The certificate of infeasibility can be obtained very fast. If an optimal solution exists, the running time of the algorithm depends on the performance of the KSP algorithm used to solve the pricing problem.

6.2.5 Link load reduction

In Section 6.1.2 we gave an overview on the applied initialization procedure of the chosen exemplary networks for the reconfiguration tasks. The maximum link load on each connection in each network was restricted to 80% and 90% respectively of its routing capacity. The objective of the initialization was to minimize the costs of the network retrieved from location and connection selection, from dimensioning and hardware installments. The limits on the connections' link loads were only auxiliary conditions.

In this application of the link load reduction scenario, we try to further reduce the link load reduction. However, to reconfigure *efficiently* in terms of Section 2.1, the number of changes of the initial routings was limited.

Procedure

In Section 3.2, we suggested different variations of link load reduction tasks. For instance, depending on the exact formulation, it is possible to reduce the average link load or a weighted sum of link loads in the network. With the weighted sum link load reduction, it is possible to reduce the influence of bottleneck connections to the link load reduction task. If the maximum link load should be reduced, the formulation of MIP 3.C on page 28 has to be slightly modified. The α_e variables measuring the reduction amount for each edge e individually are replaced by a single variable α. The capacity constraints are modified to

$$\sum_{k \in \mathcal{K}} \sum_{P \in \mathcal{P}_k : e \in P} p_k \cdot f_k(P) - C_e \cdot \alpha \;\leq\; 0, \qquad e \in E,$$

and the new objective function simply reads as:

$$\min\ \alpha.$$

In contrast to the **MinNoC** reconfiguration tasks described in the former sections, the optimal solution obtained in the root node of the branch-and-bound tree is usually not integral for this **BoundNoC** scenario. To limit the running time of the solution process for each instance, we restricted the number of branch-and-bound iterations for the applied Algorithm 4. The first five iterations are supposed to detect missing primal variables. The computation time for these iterations was restricted to 10 minutes. For the last MIP solution process, the running time was limited to one hour. The task for this iteration was to retrieve the best solution out of the given variable set without

generating new path variables. Together with the initialization phase, the overall running time for each network configuration was therefore restricted to two hours.

For each initial network configuration, we tested the link load reduction capability if 20, 30, and 50 changes of the original routing are allowed. Additionally, we calculated the optimal link load reduced network configuration if there were no limits on the number of changes.

Like in the **MinNoC** case of connection clearing, we ran an additional testing procedure. We relaxed the diversification constraint and tested the link load reduction capabilities if all network components were supposed to be fail-safe.

Results

Table 6.5 displays the results for the network configurations with 90% maximum initial link loads when diversification is respected. The other results can be found in Appendix A.4. All network configurations have a certain link load reduction potential. Depending on the network structure, this potential is only marginal (p45) or really huge (p09). If more changes are admissible, solutions tend to use more changes, regardless of the necessity. In the case where the number of changes is unbounded for the 90% configuration of p69, the link load can be reduced to 85.625%. The actual number of changes to the initial routing is 184. However, the same link load reduction can be achieved with only 48 changes. This phenomenon can be observed in many cases. For p07 in the 80% configuration for example, the best achieved objective is equal in all reconfiguration tests. The number of actual routing changes depends on the maximal allowed changes.

The gap between the optimal solution retried in the root node of the branch-and-bound tree and the best found integral solution is usually between 0.0 and 2.6%. This gap might have two causes. One is the integrality gap between the optimal fractional and the optimal integral solution and the other one is a possible impreciseness of the branch-and-price algorithm as described in Section 4.3.1 and Section 5.2.2. Only p09 led to greater optimality gaps. Therefore, we repeated the reconfiguration for the 80% initialization and doubled the computation time bounds for all iterations to obtain smaller gaps. However, only the result for the unbounded reconfiguration could be improved. The results of these computations can be found in Table 6.6. Therefore, we assume that the large gap between the best solution and an optimal fractional one is mostly due to the integrality gap.

For this reconfiguration scenario, there is a connection between the initialization quality of the network and the reconfiguration amount. As mentioned in Section 6.1.3, the gap between calculated lower bound and best found solution was by more than 80% for both initial configurations of network p09. In this reconfiguration scenario a huge link load reduction by more than 50% is possible for both configurations. The two network p44 and p45 with optimal initial solutions have only small reconfiguration capabilities, as long as diversification is respected. For some networks, the relaxation

LINK LOAD REDUCTION – 90% MAX. INITIAL LINK LOAD												
						BnB NODES						
NET	MAX CHNG	ACT. CHNG	LINK LOAD	GAP [%]	SOLUTION TIME	IT 1	IT 2	IT 3	IT 4	IT 5	IT 6
p01	0		0.890625								
	20	20	0.84375	0.26	01:50:50	3609	3123	2979	2649	2700	17098
	30	29	0.828125	0.31	01:50:51	2818	2160	2044	1820	2090	11732
	50	50	0.8125	1.30	01:51:11	2946	1698	1854	1638	1291	10341
	10000	132	0.78125	0.00	00:04:01	1113	50	0	0	0	50
p07	0		0.875								
	20	19	0.78125	1.19	01:40:14	15518	8771	9363	9034	0	60736
	30	30	0.75	1.93	01:50:18	15686	12819	12742	12640	12638	77777
	50	50	0.6875	0.00	00:10:10	14670	10	0	0	0	10
	10000	104	0.6875	1.85	01:50:44	7505	12607	13533	11803	15434	69652
p09	0		0.875								
	20	20	0.833333	6.82	01:50:41	10887	8264	9138	9567	8894	62606
	30	30	0.833333	11.61	01:41:03	23035	26938	25300	25330	0	161219
	50	43	0.75	6.87	01:51:05	15341	15800	17178	16441	16010	104856
	10000	580	0.4375	8.91	01:51:01	451	651	479	474	413	3495
p20	0		0.890625								
	20	17	0.8125	1.42	01:40:15	13309	10945	12822	12572	0	56415
	30	29	0.78125	0.50	01:50:43	17851	15004	10750	5511	6075	54510
	50	49	0.75	0.84	01:50:36	4253	4387	4636	3469	4244	24105
	10000	306	0.6875	2.26	01:50:48	835	655	673	566	583	3179
p30	0		0.880952								
	20	20	0.84127	0.00	00:00:04	1	0	0	0	0	1
	30	30	0.827381	0.00	00:00:07	1	0	0	0	0	1
	50	50	0.80754	0.00	00:00:07	1	0	0	0	0	1
	10000	640	0.644841	0.09	01:42:24	60	57	56	57	0	332
p44	0		0.899802								
	20	20	0.889881	0.00	00:00:15	4	0	0	0	0	4
	30	30	0.884921	0.00	00:00:15	1	0	0	0	0	1
	50	50	0.875	0.00	00:00:14	1	0	0	0	0	1
	10000	523	0.83631	0.03	00:02:06	6	0	0	0	0	6
p45	0		0.89881								
	20	20	0.880952	0.00	00:00:05	1	0	0	0	0	1
	30	21	0.880952	0.00	00:00:05	1	0	0	0	0	1
	50	21	0.880952	0.00	00:00:05	1	0	0	0	0	1
	10000	24	0.880952	0.00	00:00:05	1	0	0	0	0	1
p69	0		0.89								
	20	20	0.86	0.19	01:51:42	177	192	171	159	169	1108
	30	30	0.8575	0.28	01:52:30	89	82	76	81	81	744
	50	48	0.85625	0.26	01:52:41	75	72	61	86	55	379
	10000	184	0.85625	0.26	01:51:45	98	103	101	96	97	605

Table 6.5: Results of the link load reduction scenario for network configurations with 90% maximum initial link loads. Beneath the column stating limit to the changes for the reconfiguration, there is the actual number of changes for the best found solution. The gapcolumn displays the gap between best found solution and the optimal solution of the root node of the branch-and-bound tree. The last six columns state the number of branch-and-bound nodes that are processed within the corresponding iteration. The first row for each instance displays the initial maximum link load. Since the demand sum of no network configuration exceeds 10000, the last row for each instance corresponds to the unbounded link load reduction task.

LINK LOAD REDUCTION – 80% MAX. INITIAL LINK LOAD											
						BnB NODES					
NET	MAX CHNG	ACT. CHNG	LINK LOAD	GAP [%]	SOLUTION TIME	IT 1	IT 2	IT 3	IT 4	IT 5	IT 6
p09	0		0.75								
	20	20	0.75	11.63	02:41:04	97472	99818	0	0	0	676700
	30	28	0.75	17.98	03:42:47	72455	75397	75230	75030	71371	427605
	50	40	0.625	6.01	03:41:02	29252	26650	17205	24168	23951	153548
	10000	693	0.302734	5.21	03:41:55	1330	1168	1297	1058	1171	5258

Table 6.6: Results for the 80% maximum initial link load configuration of p09 with doubled calculation time. Only the result for the unbounded reconfiguration could be improved.

of the diversification condition leads to an improvement of the solution. For the 80% of p45 the maximum link load can be reduced to roughly 45% compared to 78% if diversification is respected. For most network configurations the difference is rather small.

Three networks (p30, p44, p45) had integral solutions already in the root node of the branch-and-bound tree. This property is independent of the limit on the number of changes. On the contrary, in roughly 45% of the configurations, the time limit on the branch-and-price algorithm was reached in each of the six iteration steps.

For network p01 the situation occurred that a bottleneck link prevents better optimization results. The optimal maximum link load is at 78.125%, independently of the initial configuration and of diversification constraint relaxation. Such a connection hampers the link load reduction on other network connections, because the value of the α variable measuring the maximum link load cannot be further reduced.

Conclusion

In contrast to the **MinNoC** reconfiguration tasks, the link load reduction scenario does not tend to produce integer feasible solutions in the root node of the branch-and-bound tree. This leads to two disadvantages compared to the former reconfiguration scenarios. First of all, the time for the solution process rises. In contrast to the **MinNoC** reconfigurations with often exact results within few seconds, the solution process can take several hours. Secondly, the quality of the achieved solution can only be compared with the optimal fractional solution. It is not clear to what extent an optimality gap it is due to the integrality gap or to impreciseness because of an incomplete branch-and-bound tree. However, this gap is small in most cases. In more than 68% of the computations, this gap was less than 1%.

Several networks have a fair link load reduction contingency. However, the number of changes to the initial routing is often too large for an efficient reconfiguration process. In many cases it is possible to restrict the number of changes to the initial routing and obtain results only slightly worse than the unbounded optimal solution.

6.2.6 Summary of the results

The objective function formulation of the **MinNoC** model seems to favor the optimality of integer solutions. At least for the tested real world telecommunication networks, almost all test instances provided integer optimal solutions already in root node of the branch-and-bound tree. Therefore, for nearly all reconfiguration problems that can be formulated with the **MinNoC** model, it is possible to guarantee exact solutions. In all cases where the tested reconfiguration tasks could not be solved, we could find a certificate that the corresponding instance is infeasible. For almost all solutions obtained with one of the **MinNoC** problem formulations, it is possible to prove optimality of the solution. The problems could be solved very fast because the usage of a branch-and-bound procedure was unnecessary in most cases.

The test procedure of the **BoundNoC** problem formulation on real world networks provided feasible solutions for all instances. This is not surprising because the initial routings are valid from the beginning. In contrast to the objective function in the **MinNoC** case, the limitation of routing leads only rarely to an optimal integral solution detected already in the root node of the branch-and-bound tree. Because of the need for a branch-and-bound procedure, the solution time for **BoundNoC** problems is in most cases larger than the time spent to solve **MinNoC** problems. Due to the applied time limit for the solution process and to the possible incompleteness of the branch-and-bound tree, it is not clear whether the gap between the best found integer solution and an optimal fractional one is caused only by the integrality gap. However, this gap is fairly small for most instances.

Although there are possibilities for an impreciseness of the algorithm, as described in detail in Chapter 4 and Chapter 5, it was possible to obtain good results in practice. For the tested real world networks, the algorithm provided optimal or near-optimal solutions within reasonable computation times. With only a small number of changes, it was possible to improve the initial routing or to find a feasible routing in the case of former infeasibility.

Chapter 7

Conclusion

In this thesis, mixed-integer models have been developed covering a large spectrum of different reconfiguration scenarios. A branch-and-price framework has been developed and implemented. Four practically interesting reconfiguration scenarios have been chosen to test the implementation of the branch-and-price framework on different real world telecommunication networks with different sizes, structures, number of demands, and demand distribution.

We showed that the pricing problem for all different reconfiguration tasks can be reduced to a K-shortest-paths search, for which K depends on the choice of the input parameters. In terms of computational complexity, this leads to a pseudo-polynomial solution approach of the pricing problem.

Because of the practical interest of a small dimensioned reconfiguration w.r.t. the number of changes to an initial routing, great importance was attached to the results of the reconfiguration tests. It turned out that limitation or minimization of routing changes is possible and useful for most test instances:

- For the investigated **BoundNoC** reconfiguration problems, the following characteristics could be observed. If no limits on the number of routing changes are given, the resulting reconfiguration contains usually a large number of routing changes. However, with a limitation on the number of changes, it is often possible to achieve results of similar quality w.r.t. the corresponding reconfiguration objective.

- For **MinNoC** problem formulations, stating the minimization of routing changes as objective function has the characteristic to ensure optimal integer solutions of the linear program relaxation. Thus, for most reconfiguration tasks which can be modeled with the **MinNoC** formulation, exact solutions can be computed within short time.

With the inclusion of both multi-failure states and commodity groups in the mathematical model, we provided possibilities to include several aspects of an integrated multi-layer planning in the single-layer SDH reconfiguration process.

We propose the further examination of the following subjects:

- A generalization of the branch-and-bound approach of branching on arc flow variables to multi-path routings would allow for a more precise computation of optimality gaps.
- The implementation of arc flow variable branching in a branch-and-price framework could provide shorter running times for the solution process.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der ausfallsicheren und effizienten Rekonfigurationsplanung für SDH-Netze. Rekonfiguration bezieht sich dabei nicht auf auf Netztopologie oder Routingkapazitäten, sondern nur auf ein initiales Routing von Telekommunikationsbedarfen. Damit eine Rekonfiguration praktisch durchführbar ist, darf der Umfang der Änderungen nicht zu groß sein. In dieser Arbeit wird eine Rekonfiguration *effizient* genannt, wenn es möglich ist, den Umfang der Änderungen an einem initialen Routing zu beschränken oder direkt zu minimieren.

Es wird ein mathematisches Modell in Form eines gemischt-ganzzahligen Linearen Programms entwickelt, das auf verschiedene Rekonfigurationsszenarien anwendbar ist. Zur Lösung des Modells wird ein branch-and-price-Verfahren vorgeschlagen, das detailliert entwickelt wird. Besonders ausführlich wird dabei auf die Lösung des Pricing-Problems eingegangen, das sich für alle unterschiedlichen Rekonfigurationsaufgaben auf ein K-kürzeste-Wege-Problem zurückführen lässt. Dies hat insbesondere komplexitätstheoretische Bedeutung, da sich das Pricing-Problem dadurch pseudo-polynomial lösen lässt.

Zudem wird das vorgeschlagene Verfahren implementiert, und vier verschiedene Rekonfigurationsszenarien werden an mehreren realen Telekommunikationsnetzen getestet. Die Ergebnisse zeigen, dass in den meisten Fällen, in denen Rekonfiguration möglich ist, diese auch mit einer kleinen Anzahl von Änderungen des initialen Routings durchgeführt und damit auch effizient gestaltet werden kann.

Ausfallsicherheit der Netze wird berücksichtigt und verallgemeinert, so dass auch Aspekte einer integrierten Multi-Layer-Netzplanung in den Rekonfigurationsprozess einbezogen werden können.

Appendix A

Tables

A.1 Connection clearing

A.1.1 Instances with 80% maximum link load

p07

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L112-N9-N15	8	[$\gamma = 1$]	8	L114-N10-N6	50	[$\gamma = 3$]	50
L85-N7-N1	204	[$\gamma = 39$]	[$\gamma = 3$]	L111-N9-N13	51	[$\gamma = 11$]	51
L36-N1-N11	51	51	51	L35-N1-N2	202	[$\gamma = 38$]	[$\gamma = 13$]
L51-N3-N8	51	[$\gamma = 4$]	51	L32-N15-N9	50	[$\gamma = 20.5$]	[$\gamma = 9.33333$]
L58-N4-N6	204	[$\gamma = 10.8333$]	204	L40-N2-N5	202	[$\gamma = 38$]	[$\gamma = 13$]
L96-N8-N1	51	51	51	L70-N5-N11	1	1	1
L74-N6-N9	116	[$\gamma = 25$]	[$\gamma = 15$]	L77-N6-N11	204	205	204
L100-N8-N6	4	4	4	L92-N7-N8	204	[$\gamma = 38.5$]	[$\gamma = 16.3333$]
L129-N15-N13	50	[$\gamma = 46$]	[$\gamma = 9.33333$]	L99-N8-N4	204	[$\gamma = 32.25$]	216
L108-N8-N11	204	[$\gamma = 7$]	204	L69-N5-N11	203	[$\gamma = 32$]	[$\gamma = 16.3333$]
L80-N6-N13	183	184	183	L87-N7-N4	12	[$\gamma = 4$]	12
L9-N6-N12	51	[$\gamma = 46$]	[$\gamma = 10$]	L124-N14-N11	3	3	3
L49-N3-N6	51	55	51	L122-N13-N14	50	[$\gamma = 18$]	[$\gamma = 8$]
L118-N11-N13	168	[$\gamma = 30$]	168	L26-N12-N9	51	[$\gamma = 46$]	[$\gamma = 18$]
L67-N4-N10	142	[$\gamma = 26$]	142	L55-N4-N3	104	119	106
L82-N6-N14	51	[$\gamma = 18.5$]	[$\gamma = 8$]	L113-N10-N3	110	[$\gamma = 24$]	126

p44

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N3	1399	[$\gamma = 623$]	[$\gamma = 483$]	L15-N15-N2	1481	[$\gamma = 426.333$]	[$\gamma = 374.333$]
L2-N2-N11	1972	[$\gamma = 885.5$]	[$\gamma = 842.5$]	L16-N15-N10	870	[$\gamma = 384.5$]	[$\gamma = 384.5$]
L4-N4-N2	896	[$\gamma = 386.5$]	[$\gamma = 382.5$]	L17-N15-N12	403	[$\gamma = 280$]	[$\gamma = 52$]
L5-N4-N12	945	[$\gamma = 411$]	[$\gamma = 407$]	L18-N15-N13	1399	[$\gamma = 623$]	[$\gamma = 483$]
L7-N5-N12	535	[$\gamma = 217$]	[$\gamma = 217$]	L19-N15-N17	403	[$\gamma = 28$]	[$\gamma = 28$]
L8-N6-N5	535	[$\gamma = 217$]	[$\gamma = 217$]	L21-N16-N8	819	[$\gamma = 327$]	[$\gamma = 228.5$]
L9-N6-N12	403	[$\gamma = 128$]	[$\gamma = 124$]	L22-N19-N7	403	[$\gamma = 242$]	[$\gamma = 44$]
L10-N8-N12	1503	[$\gamma = 648.5$]	[$\gamma = 575.5$]	L24-N19-N20	1274	[$\gamma = 521.5$]	[$\gamma = 425.5$]
L11-N9-N12	1598	[$\gamma = 748$]	[$\gamma = 712$]	L25-N20-N11	1490	[$\gamma = 637.5$]	[$\gamma = 555.5$]
L12-N13-N3	1399	[$\gamma = 623$]	[$\gamma = 483$]	L26-N1-N7	826	[$\gamma = 292.5$]	[$\gamma = 133.5$]
L13-N14-N9	1008	[$\gamma = 447$]	[$\gamma = 411$]	L28-N10-N17	126	[$\gamma = 35$]	[$\gamma = 24$]
L14-N14-N16	402	[$\gamma = 325$]	[$\gamma = 214$]				

p09

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N14	6	6	6	L75-N5-N13	73	73	73
L2-N2-N6	24	24	24	L76-N5-N13	5	5	5
L3-N2-N14	6	6	6	L77-N5-N13	3	3	3
L4-N3-N10	3	3	3	L78-N5-N13	85	85	85
L5-N4-N5	6	6	6	L79-N6-N14	4	4	4
L6-N5-N8	6	6	6	L80-N6-N14	5	5	5
L7-N5-N12	6	6	6	L81-N6-N14	168	168	168
L8-N5-N13	6	6	6	L82-N7-N4	3	3	3
L9-N6-N14	12	12	12	L83-N7-N5	4	4	4
L10-N7-N4	3	3	3	L84-N8-N12	8	8	8
L11-N7-N5	2	2	2	L85-N8-N12	6	6	6
L12-N8-N4	5	5	5	L86-N8-N12	6	6	6
L13-N8-N12	6	6	6	L87-N8-N12	6	6	6
L14-N8-N13	135	135	135	L88-N8-N13	5	5	5
L15-N8-N16	155	155	155	L89-N8-N14	141	141	141
L16-N8-N17	6	6	6	L90-N8-N14	6	6	6
L17-N9-N1	147	[γ = 1.77778]	147	L91-N8-N16	38	38	38
L18-N9-N3	100	100	100	L92-N8-N17	7	7	7
L19-N9-N5	3	3	3	L93-N8-N17	5	5	5
L20-N9-N10	6	6	6	L94-N8-N17	6	6	6
L21-N10-N4	4	4	4	L95-N8-N18	5	5	5
L22-N10-N8	92	92	92	L96-N8-N18	5	5	5
L23-N10-N14	6	6	6	L97-N10-N5	6	6	6
L24-N10-N15	3	3	3	L98-N9-N1	5	5	5
L25-N11-N10	234	[γ = 22]	234	L99-N9-N1	3	3	3
L26-N11-N10	29	29	29	L100-N9-N3	98	98	98
L27-N11-N14	263	[γ = 108]	263	L101-N9-N5	5	5	5
L28-N12-N16	156	156	156	L102-N9-N5	6	6	6
L29-N13-N4	5	5	5	L103-N9-N10	6	6	6
L30-N13-N8	4	4	4	L104-N9-N10	3	3	3
L31-N14-N8	6	6	6	L105-N9-N10	5	5	5
L32-N14-N17	6	6	6	L106-N9-N10	6	6	6
L33-N14-N18	2	2	2	L107-N9-N10	5	5	5
L34-N15-N1	2	2	2	L108-N10-N1	6	6	6
L35-N15-N14	3	3	3	L109-N10-N1	6	6	6
L36-N17-N12	6	6	6	L110-N10-N1	12	12	12
L37-N17-N19	5	5	5	L111-N10-N1	6	6	6
L38-N18-N8	34	34	34	L112-N10-N4	120	120	120
L39-N18-N17	144	144	144	L113-N10-N5	5	5	5
L40-N19-N12	3	3	3	L114-N10-N5	5	5	5
L41-N1-N2	3	3	3	L115-N10-N8	5	5	5
L42-N1-N2	5	5	5	L116-N10-N8	6	6	6
L43-N1-N2	139	[γ = 13.5]	139	L117-N10-N8	4	4	4
L44-N1-N14	6	6	6	L118-N10-N8	6	6	6
L45-N2-N1	2	2	2	L119-N10-N14	12	12	12
L46-N2-N6	5	5	5	L120-N10-N14	10	10	10
L47-N2-N6	156	156	156	L121-N10-N14	6	6	6
L48-N2-N6	5	5	5	L122-N10-N14	7	7	7
L49-N2-N14	9	9	9	L123-N10-N14	5	5	5
L50-N2-N14	66	66	66	L124-N10-N15	3	3	3
L51-N2-N14	5	5	5	L125-N10-N15	3	3	3
L52-N2-N14	5	5	5	L126-N12-N16	17	17	17
L53-N1-N14	5	5	5	L127-N12-N17	5	5	5
L54-N1-N14	5	5	5	L128-N12-N17	5	5	5
L55-N1-N14	3	3	3	L129-N12-N17	4	4	4
L56-N1-N15	3	3	3	L130-N12-N19	146	[γ = 6.75]	146
L57-N3-N10	98	98	98	L131-N13-N4	5	5	5
L58-N3-N10	97	97	97	L132-N13-N8	22	22	22
L59-N4-N8	5	5	5	L133-N14-N6	5	5	5
L60-N4-N8	5	5	5	L134-N14-N8	4	4	4
L61-N4-N10	3	3	3	L135-N14-N17	7	7	7
L62-N4-N10	5	5	5	L136-N14-N17	6	6	6
L63-N4-N13	60	60	60	L137-N14-N18	133	133	133
L64-N5-N4	4	4	4	L138-N14-N18	3	3	3
L65-N5-N4	5	5	5	L139-N15-N14	3	3	3
L66-N5-N4	84	84	84	L140-N17-N14	6	6	6
L67-N5-N8	10	10	10	L141-N17-N18	16	16	16
L68-N5-N8	6	6	6	L142-N17-N19	4	4	4
L69-N5-N8	6	6	6	L143-N18-N14	29	29	29
L70-N5-N9	38	38	38	L144-N18-N17	3	3	3
L71-N5-N10	8	8	8	L145-N19-N12	5	5	5
L72-N5-N12	4	4	4	L146-N19-N17	148	[γ = 6.5]	148
L73-N5-N12	5	5	5	L147-N19-N12	4	4	4
L74-N5-N12	6	6	6	L148-N19-N17	5	5	5

p20

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N12-N2	44	44	44	L48-N18-N3	10	10	10
L2-N13-N4	43	49	43	L49-N18-N10	46	46	46
L3-N13-N1	23	23	23	L50-N15-N24	147	[$\gamma = 20$]	147
L4-N11-N6	50	50	50	L51-N15-N23	29	29	29
L5-N11-N15	77	77	77	L52-N2-N3	5	5	5
L6-N11-N7	66	66	66	L53-N2-N9	51	51	51
L7-N11-N21	7	7	7	L54-N9-N5	37	37	37
L8-N11-N17	6	6	6	L55-N5-N21	45	47	45
L9-N11-N14	7	7	7	L56-N13-N2	6	6	6
L10-N21-N20	58	58	58	L57-N10-N21	29	29	29
L11-N21-N18	11	11	11	L58-N11-N10	24	24	24
L12-N12-N13	7	7	7	L59-N16-N22	14	14	14
L13-N20-N24	38	38	38	L60-N1-N14	47	47	47
L14-N7-N15	109	[$\gamma = 3.2$]	109	L61-N8-N6	82	[$\gamma = 13.5$]	82
L15-N7-N6	154	154	154	L62-N17-N11	30	30	30
L16-N7-N17	34	34	34	L63-N12-N22	7	7	7
L17-N18-N2	9	9	9	L64-N12-N16	6	6	6
L18-N18-N5	46	46	46	L65-N13-N18	7	7	7
L19-N18-N10	6	6	6	L66-N13-N6	6	6	6
L20-N15-N24	27	27	27	L67-N13-N18	56	56	56
L21-N15-N23	6	6	6	L68-N13-N4	6	6	6
L22-N15-N19	6	6	6	L69-N11-N15	7	7	7
L23-N12-N7	47	47	47	L70-N11-N7	7	7	7
L24-N2-N3	5	5	5	L71-N11-N14	6	6	6
L25-N2-N9	19	19	19	L72-N21-N18	7	7	7
L26-N11-N18	106	106	106	L73-N20-N24	6	6	6
L27-N12-N13	90	90	90	L74-N7-N6	5	5	5
L28-N12-N1	49	59	49	L75-N7-N17	6	6	6
L29-N12-N16	28	28	28	L76-N18-N2	7	7	7
L30-N12-N4	45	45	45	L77-N18-N5	33	33	33
L31-N13-N2	44	44	44	L78-N13-N11	88	88	88
L32-N13-N1	73	73	73	L79-N18-N10	6	6	6
L33-N13-N6	119	119	119	L80-N15-N24	12	12	12
L34-N12-N22	27	27	27	L81-N15-N23	6	6	6
L35-N4-N8	20	20	20	L82-N15-N19	6	6	6
L36-N17-N6	26	26	26	L83-N2-N9	6	6	6
L37-N19-N15	154	[$\gamma = 6$]	154	L84-N12-N22	6	6	6
L38-N11-N21	153	[$\gamma = 1.16667$]	153	L85-N22-N2	6	6	6
L39-N11-N23	17	[$\gamma = 12$]	17	L86-N13-N14	6	6	6
L40-N11-N14	32	32	32	L87-N14-N11	5	5	5
L41-N11-N19	142	[$\gamma = 36$]	142	L88-N5-N9	5	5	5
L42-N21-N20	106	[$\gamma = 2$]	106	L89-N13-N6	6	6	6
L43-N20-N24	130	[$\gamma = 18$]	130	L90-N16-N22	4	4	4
L44-N7-N8	78	[$\gamma = 13$]	80	L91-N22-N2	16	[$\gamma = 1$]	16
L45-N12-N16	6	6	6	L92-N5-N9	6	6	6
L46-N7-N17	58	58	58	L93-N1-N14	4	4	4
L47-N18-N2	80	80	80	L94-N14-N10	33	33	33

p45

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N3	931	[$\gamma = 441$]	[$\gamma = 270$]	L14-N14-N16	573	[$\gamma = 395$]	[$\gamma = 260$]
L2-N2-N11	559	[$\gamma = 441$]	[$\gamma = 264$]	L16-N15-N10	235	[$\gamma = 36$]	[$\gamma = 24$]
L4-N4-N2	1088	[$\gamma = 441$]	[$\gamma = 270$]	L17-N15-N12	1256	[$\gamma = 444$]	[$\gamma = 238$]
L5-N4-N12	1100	[$\gamma = 445$]	[$\gamma = 270$]	L18-N15-N13	931	[$\gamma = 441$]	[$\gamma = 270$]
L7-N5-N12	156	[$\gamma = 156$]	[$\gamma = 156$]	L19-N15-N17	96	[$\gamma = 37$]	[$\gamma = 32$]
L8-N6-N5	156	[$\gamma = 156$]	[$\gamma = 156$]	L21-N16-N8	523	[$\gamma = 396$]	[$\gamma = 266$]
L9-N6-N12	196	[$\gamma = 156$]	[$\gamma = 82$]	L22-N19-N7	828	[$\gamma = 441$]	[$\gamma = 266$]
L10-N8-N12	403	[$\gamma = 394$]	[$\gamma = 230$]	L24-N19-N20	486	[$\gamma = 440$]	[$\gamma = 264$]
L11-N9-N12	723	[$\gamma = 394$]	[$\gamma = 311$]	L25-N20-N11	444	[$\gamma = 440$]	[$\gamma = 264$]
L12-N13-N3	931	[$\gamma = 441$]	[$\gamma = 270$]	L26-N1-N7	849	[$\gamma = 440$]	[$\gamma = 270$]
L13-N14-N9	651	[$\gamma = 395$]	[$\gamma = 275$]	L28-N10-N17	37	[$\gamma = 36$]	[$\gamma = 34$]

p30

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N23	38	[$\gamma = 2$]	[$\gamma = 2$]	L37-N29-N26	13	13	13
L2-N3-N21	318	[$\gamma = 24$]	[$\gamma = 12$]	L38-N30-N26	582	[$\gamma = 28$]	[$\gamma = 10$]
L3-N3-N45	320	[$\gamma = 24$]	[$\gamma = 12$]	L44-N35-N23	380	[$\gamma = 12$]	380
L4-N6-N32	619	[$\gamma = 82$]	[$\gamma = 34$]	L45-N35-N41	374	[$\gamma = 12$]	[$\gamma = 4$]
L5-N6-N40	615	[$\gamma = 77$]	615	L46-N36-N5	6	6	6
L7-N8-N23	674	[$\gamma = 164.5$]	[$\gamma = 34$]	L47-N36-N10	41	41	41
L8-N8-N38	633	[$\gamma = 22$]	633	L48-N36-N13	58	[$\gamma = 1$]	58
L9-N8-N42	37	37	37	L49-N36-N26	114	[$\gamma = 56$]	114
L10-N10-N19	244	[$\gamma = 34$]	[$\gamma = 11$]	L50-N37-N19	238	[$\gamma = 34$]	[$\gamma = 11$]
L11-N11-N27	250	[$\gamma = 34$]	[$\gamma = 6$]	L52-N41-N23	40	40	40
L12-N11-N37	244	[$\gamma = 34$]	[$\gamma = 10$]	L53-N42-N38	645	[$\gamma = 22$]	[$\gamma = 4$]
L13-N12-N50	615	[$\gamma = 140$]	[$\gamma = 6$]	L54-N42-N40	686	[$\gamma = 163$]	[$\gamma = 7$]
L14-N13-N5	6	6	6	L55-N43-N51	458	[$\gamma = 12$]	[$\gamma = 2$]
L15-N13-N10	38	[$\gamma = 8$]	38	L56-N44-N2	210	[$\gamma = 36$]	[$\gamma = 18$]
L16-N14-N4	201	[$\gamma = 44$]	[$\gamma = 19$]	L57-N46-N1	38	[$\gamma = 2$]	[$\gamma = 2$]
L17-N14-N16	205	[$\gamma = 44$]	[$\gamma = 18$]	L58-N46-N26	38	[$\gamma = 2$]	38
L20-N16-N40	215	[$\gamma = 44$]	[$\gamma = 18$]	L59-N47-N20	294	[$\gamma = 2$]	294
L23-N20-N12	588	[$\gamma = 142.5$]	[$\gamma = 42$]	L60-N47-N23	340	[$\gamma = 6$]	340
L24-N20-N43	460	[$\gamma = 12$]	[$\gamma = 2$]	L61-N47-N45	334	[$\gamma = 23$]	[$\gamma = 2$]
L25-N21-N51	452	[$\gamma = 10$]	[$\gamma = 1$]	L65-N50-N2	216	[$\gamma = 36$]	[$\gamma = 12$]
L26-N22-N44	206	[$\gamma = 36$]	[$\gamma = 21$]	L66-N50-N32	590	[$\gamma = 78$]	[$\gamma = 41$]
L27-N23-N25	715	715	715	L67-N52-N4	215	[$\gamma = 44$]	[$\gamma = 15$]
L28-N23-N26	361	361	361	L68-N52-N21	524	524	524
L29-N23-N27	40	40	40	L69-N52-N21	38	38	38
L30-N23-N28	350	[$\gamma = 49$]	350	L70-N52-N22	216	[$\gamma = 36$]	[$\gamma = 9$]
L31-N24-N23	115	115	115	L71-N52-N26	48	48	48
L32-N24-N25	115	115	115	L72-N52-N30	588	[$\gamma = 27$]	[$\gamma = 9$]
L33-N25-N26	817	817	817	L73-N52-N50	42	42	42
L34-N27-N26	287	[$\gamma = 6$]	287	L74-N54-N10	338	[$\gamma = 97$]	338
L35-N27-N28	330	[$\gamma = 49$]	[$\gamma = 33$]	L75-N54-N41	344	[$\gamma = 101$]	344
L36-N29-N25	13	13	13				

p69

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N13	340	340	340	L36-N39-N47	259	[$\gamma = 0.5$]	259
L2-N1-N23	240	[$\gamma = 169$]	[$\gamma = 8$]	L37-N41-N49	233	[$\gamma = 24.8$]	233
L3-N1-N25	433	[$\gamma = 16.6667$]	470	L38-N43-N53	55	[$\gamma = 28$]	[$\gamma = 1$]
L4-N1-N27	169	[$\gamma = 1$]	169	L39-N45-N51	226	[$\gamma = 131$]	[$\gamma = 4$]
L5-N3-N7	203	[$\gamma = 37.6667$]	[$\gamma = 2$]	L40-N47-N55	387	390	387
L6-N3-N43	203	[$\gamma = 41.1667$]	[$\gamma = 1$]	L41-N49-N53	55	[$\gamma = 28$]	[$\gamma = 1$]
L7-N5-N29	218	[$\gamma = 182$]	[$\gamma = 38$]	L42-N1-N2	88	[$\gamma = 88$]	[$\gamma = 88$]
L8-N5-N31	218	[$\gamma = 182$]	[$\gamma = 8$]	L43-N3-N4	36	[$\gamma = 36$]	[$\gamma = 36$]
L9-N7-N15	156	[$\gamma = 0.5$]	156	L44-N5-N6	69	[$\gamma = 69$]	[$\gamma = 69$]
L10-N7-N51	183	249	183	L45-N7-N8	28	[$\gamma = 28$]	[$\gamma = 28$]
L11-N9-N17	226	[$\gamma = 130$]	[$\gamma = 6$]	L46-N9-N10	104	[$\gamma = 104$]	[$\gamma = 104$]
L12-N9-N25	489	[$\gamma = 89.5$]	[$\gamma = 48$]	L47-N11-N12	59	[$\gamma = 59$]	[$\gamma = 59$]
L13-N9-N35	54	54	54	L48-N13-N14	55	[$\gamma = 55$]	[$\gamma = 55$]
L14-N9-N41	232	[$\gamma = 0.25$]	232	L49-N15-N16	38	[$\gamma = 38$]	[$\gamma = 38$]
L15-N9-N49	171	171	171	L50-N17-N18	39	[$\gamma = 39$]	[$\gamma = 39$]
L16-N11-N31	218	[$\gamma = 181.5$]	[$\gamma = 9$]	L51-N19-N20	29	[$\gamma = 29$]	[$\gamma = 29$]
L17-N11-N39	218	[$\gamma = 181.5$]	[$\gamma = 38$]	L52-N21-N22	105	[$\gamma = 105$]	[$\gamma = 105$]
L18-N13-N21	311	311	311	L53-N23-N24	141	[$\gamma = 141$]	[$\gamma = 141$]
L19-N13-N39	361	361	361	L54-N25-N26	101	[$\gamma = 101$]	[$\gamma = 101$]
L20-N15-N41	327	[$\gamma = 30.4$]	[$\gamma = 3.5$]	L55-N27-N28	156	[$\gamma = 156$]	[$\gamma = 156$]
L21-N15-N51	183	224	183	L56-N29-N30	65	[$\gamma = 65$]	[$\gamma = 65$]
L22-N17-N37	226	[$\gamma = 130$]	[$\gamma = 7$]	L57-N31-N32	63	[$\gamma = 63$]	[$\gamma = 63$]
L23-N19-N23	240	[$\gamma = 169$]	[$\gamma = 10$]	L58-N33-N34	106	[$\gamma = 106$]	[$\gamma = 106$]
L24-N19-N27	240	[$\gamma = 170$]	[$\gamma = 10$]	L59-N35-N36	109	[$\gamma = 109$]	[$\gamma = 109$]
L25-N21-N25	212	213	212	L60-N37-N38	38	[$\gamma = 38$]	[$\gamma = 38$]
L26-N21-N35	249	[$\gamma = 4.5$]	249	L61-N39-N40	66	[$\gamma = 66$]	[$\gamma = 66$]
L27-N21-N47	208	209	208	L62-N41-N42	57	[$\gamma = 57$]	[$\gamma = 57$]
L28-N27-N39	353	[$\gamma = 265$]	[$\gamma = 4$]	L63-N43-N44	93	[$\gamma = 93$]	[$\gamma = 93$]
L29-N29-N39	107	107	107	L64-N45-N46	54	[$\gamma = 54$]	[$\gamma = 54$]
L30-N29-N55	271	[$\gamma = 211.5$]	[$\gamma = 6.66667$]	L65-N47-N48	71	[$\gamma = 71$]	[$\gamma = 71$]
L31-N33-N35	210	217	210	L66-N49-N50	47	[$\gamma = 47$]	[$\gamma = 47$]
L32-N33-N43	242	[$\gamma = 42$]	242	L67-N51-N52	113	[$\gamma = 113$]	[$\gamma = 113$]
L33-N33-N55	398	[$\gamma = 88$]	[$\gamma = 8$]	L68-N53-N54	28	[$\gamma = 28$]	[$\gamma = 28$]
L34-N35-N49	311	[$\gamma = 32.2$]	311	L69-N55-N56	58	[$\gamma = 58$]	[$\gamma = 58$]
L35-N37-N45	226	[$\gamma = 130$]	[$\gamma = 7$]				

A.1.2 Instances with 90% maximum link load

p01

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N17-N15	52	[$\gamma = 1$]	52	L32-N9-N10	3	3	3
L2-N4-N5	57	[$\gamma = 2$]	[$\gamma = 2$]	L33-N8-N9	53	53	53
L3-N5-N6	57	[$\gamma = 2$]	[$\gamma = 2$]	L35-N11-N10	1	1	1
L4-N6-N7	54	54	54	L36-N11-N9	51	51	51
L5-N7-N12	52	53	52	L37-N8-N10	4	4	4
L6-N12-N16	55	55	55	L38-N8-N11	4	4	4
L7-N16-N15	54	54	54	L39-N7-N10	56	56	56
L9-N16-N13	53	53	53	L40-N7-N9	57	57	57
L10-N13-N14	50	50	50	L41-N12-N8	2	2	2
L12-N17-N1	53	55	53	L42-N14-N11	4	4	4
L14-N12-N11	2	2	2	L44-N16-N11	5	5	5
L15-N11-N13	49	49	49	L45-N1-N2	57	57	57
L16-N7-N11	54	54	54	L48-N16-N10	52	52	52
L18-N12-N6	3	3	3	L49-N15-N14	51	53	51
L19-N5-N12	57	[$\gamma = 2$]	[$\gamma = 2$]	L50-N12-N9	3	3	3
L21-N3-N6	56	56	56	L54-N2-N15	54	54	54
L23-N17-N2	2	2	2	L55-N15-N12	57	57	57
L25-N2-N7	55	55	55	L56-N2-N12	57	57	57
L26-N4-N6	57	57	57	L57-N1-N3	53	53	53
L27-N6-N8	57	57	57	L58-N1-N4	4	4	4
L28-N7-N8	57	57	57	L59-N1-N15	2	2	2
L29-N3-N8	4	4	4	L60-N2-N3	55	55	55
L30-N5-N7	57	[$\gamma = 2$]	[$\gamma = 2$]	L61-N3-N4	53	53	53
L31-N4-N7	19	19	19				

p07

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L45-N1-N11	1	1	1	L49-N3-N6	36	36	36
L112-N9-N15	48	[$\gamma = 15$]	[$\gamma = 3$]	L118-N11-N13	38	38	38
L76-N6-N11	20	20	20	L67-N4-N10	69	69	69
L63-N4-N9	16	16	16	L82-N6-N14	28	28	28
L37-N2-N1	48	66	48	L44-N2-N11	45	45	45
L85-N7-N1	40	40	40	L114-N10-N6	40	40	40
L119-N11-N14	18	18	18	L111-N9-N13	24	24	24
L36-N1-N11	60	60	60	L35-N1-N2	49	49	49
L48-N3-N6	14	14	14	L32-N15-N9	8	8	8
L22-N10-N6	30	30	30	L40-N2-N5	52	[$\gamma = 38$]	52
L73-N6-N9	1	1	1	L104-N8-N11	34	34	34
L71-N6-N9	1	1	1	L46-N1-N11	6	6	6
L53-N4-N3	16	16	16	L10-N6-N3	1	1	1
L51-N3-N8	81	81	81	L70-N5-N11	80	[$\gamma = 0.8$]	[$\gamma = 0.8$]
L5-N4-N6	1	1	1	L77-N6-N11	70	70	70
L125-N14-N13	6	6	6	L92-N7-N8	65	65	65
L103-N8-N6	58	58	58	L99-N8-N4	60	60	60
L8-N5-N11	19	19	19	L14-N6-N13	28	28	28
L58-N4-N6	43	43	43	L69-N5-N11	4	4	4
L96-N8-N1	77	77	77	L13-N6-N12	10	10	10
L74-N6-N9	72	72	72	L123-N13-N15	8	8	8
L64-N4-N10	2	2	2	L47-N1-N11	1	1	1
L24-N11-N13	16	16	16	L1-N1-N11	1	1	1
L100-N8-N6	63	63	63	L87-N7-N4	32	33	32
L57-N4-N6	1	1	1	L98-N8-N4	1	1	1
L86-N7-N4	11	11	11	L18-N8-N3	19	19	19
L7-N4-N10	28	28	28	L124-N14-N11	22	22	22
L72-N6-N9	1	1	1	L122-N13-N14	14	14	14
L15-N7-N1	18	18	18	L83-N8-N4	1	1	1
L56-N4-N6	1	1	1	L52-N3-N10	14	14	14
L30-N14-N6	16	16	16	L97-N8-N3	1	1	1
L61-N4-N9	8	8	8	L60-N4-N8	1	1	1
L129-N15-N13	48	[$\gamma = 16$]	[$\gamma = 3$]	L68-N5-N11	5	5	5
L109-N9-N13	5	5	5	L26-N12-N9	46	[$\gamma = 4$]	46
L108-N8-N11	64	64	64	L93-N8-N1	35	35	35
L80-N6-N13	52	52	52	L55-N4-N3	71	71	71
L9-N6-N12	49	[$\gamma = 12.5$]	49	L27-N12-N9	9	9	9
L110-N9-N13	1	1	1	L113-N10-N3	45	45	45
L88-N7-N8	22	22	22				

p09

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N14	7	7	7	L75-N5-N13	5	5	5
L2-N2-N6	5	5	5	L76-N5-N13	5	5	5
L3-N2-N14	7	7	7	L77-N5-N13	5	5	5
L4-N3-N10	5	5	5	L78-N5-N13	6	6	6
L5-N4-N5	159	159	159	L79-N6-N14	5	5	5
L6-N5-N8	7	7	7	L80-N6-N14	5	5	5
L7-N5-N12	5	5	5	L81-N6-N14	5	5	5
L8-N5-N13	204	204	204	L82-N7-N4	5	5	5
L9-N6-N14	144	[$\gamma = 2$]	144	L83-N7-N5	4	4	4
L10-N7-N4	4	4	4	L84-N8-N12	13	13	13
L11-N7-N5	5	5	5	L85-N8-N12	7	7	7
L12-N8-N4	7	7	7	L86-N8-N12	6	6	6
L13-N8-N12	7	7	7	L87-N8-N12	6	6	6
L14-N8-N13	6	6	6	L88-N8-N13	5	5	5
L15-N8-N16	152	152	152	L89-N8-N14	12	12	12
L16-N8-N17	7	7	7	L90-N8-N14	7	7	7
L17-N9-N1	3	3	3	L91-N8-N16	67	67	67
L18-N9-N3	129	129	129	L92-N8-N17	9	9	9
L19-N9-N5	5	5	5	L93-N8-N17	7	7	7
L20-N9-N10	49	49	49	L94-N8-N17	6	6	6
L21-N10-N4	5	5	5	L95-N8-N18	5	5	5
L22-N10-N8	6	6	6	L96-N8-N18	6	6	6
L23-N10-N14	7	7	7	L97-N10-N5	5	5	5
L24-N10-N15	5	5	5	L98-N9-N1	118	139	118
L25-N11-N10	25	25	25	L99-N9-N1	5	5	5
L26-N11-N10	282	[$\gamma = 22$]	282	L100-N9-N3	16	16	16
L27-N11-N14	307	[$\gamma = 108$]	307	L101-N9-N5	5	5	5
L28-N12-N16	68	68	68	L102-N9-N5	5	5	5
L29-N13-N4	58	58	58	L103-N9-N10	3	3	3
L30-N13-N8	6	6	6	L104-N9-N10	4	4	4
L31-N14-N8	7	7	7	L105-N9-N10	6	6	6
L32-N14-N17	7	7	7	L106-N9-N10	5	5	5
L33-N14-N18	4	4	4	L107-N9-N10	8	8	8
L34-N15-N1	4	4	4	L108-N10-N1	7	7	7
L35-N15-N14	5	5	5	L109-N10-N1	21	21	21
L36-N17-N12	7	7	7	L110-N10-N1	6	6	6
L37-N17-N19	5	5	5	L111-N10-N1	6	6	6
L38-N18-N8	7	7	7	L112-N10-N4	8	8	8
L39-N18-N17	176	[$\gamma = 0.133333$]	176	L113-N10-N5	5	5	5
L40-N19-N12	171	[$\gamma = 6$]	171	L114-N10-N5	5	5	5
L41-N1-N2	5	5	5	L115-N10-N8	130	130	130
L42-N1-N2	40	40	40	L116-N10-N8	5	5	5
L43-N1-N2	71	71	71	L117-N10-N8	6	6	6
L44-N1-N14	7	7	7	L118-N10-N8	5	5	5
L45-N2-N1	36	36	36	L119-N10-N14	12	12	12
L46-N2-N6	125	[$\gamma = 6$]	125	L120-N10-N14	8	8	8
L47-N2-N6	5	5	5	L121-N10-N14	7	7	7
L48-N2-N6	6	6	6	L122-N10-N14	6	6	6
L49-N2-N14	127	127	127	L123-N10-N14	6	6	6
L50-N2-N14	6	6	6	L124-N10-N15	5	5	5
L51-N2-N14	5	5	5	L125-N10-N15	5	5	5
L52-N2-N14	6	6	6	L126-N12-N16	131	131	131
L53-N1-N14	6	6	6	L127-N12-N17	5	5	5
L54-N1-N14	5	5	5	L128-N12-N17	5	5	5
L55-N1-N14	6	6	6	L129-N12-N17	6	6	6
L56-N1-N15	1	1	1	L130-N12-N19	5	5	5
L57-N3-N10	118	118	118	L131-N13-N4	6	6	6
L58-N3-N10	18	18	18	L132-N13-N8	168	168	168
L59-N4-N8	7	7	7	L133-N14-N6	6	6	6
L60-N4-N8	6	6	6	L134-N14-N8	6	6	6
L61-N4-N10	173	173	173	L135-N14-N17	8	8	8
L62-N4-N10	5	5	5	L136-N14-N17	7	7	7
L63-N4-N13	6	6	6	L137-N14-N18	168	168	168
L64-N5-N4	7	7	7	L138-N14-N18	23	23	23
L65-N5-N4	4	4	4	L139-N15-N14	5	5	5
L66-N5-N4	6	6	6	L140-N17-N14	6	6	6
L67-N5-N8	9	9	9	L141-N17-N18	6	6	6
L68-N5-N8	7	7	7	L142-N17-N19	5	5	5
L69-N5-N8	7	7	7	L143-N18-N14	5	5	5
L70-N5-N9	63	69	63	L144-N18-N17	6	6	6
L71-N5-N10	8	8	8	L145-N19-N12	5	5	5
L72-N5-N12	4	4	4	L146-N19-N17	176	[$\gamma = 7$]	176
L73-N5-N12	6	6	6	L147-N19-N12	5	5	5
L74-N5-N12	6	6	6	L148-N19-N17	6	6	6

p20							
		DIVERS.	NO DIVERS.			DIVERS.	NO DIVERS.
LINK	INITIAL FLOW	CHANGES	CHANGES	LINK	INITIAL FLOW	CHANGES	CHANGES
L1-N12-N2	31	31	31	L48-N18-N3	27	27	27
L2-N13-N4	51	51	51	L49-N18-N10	66	66	66
L3-N13-N1	39	39	39	L50-N15-N24	118	122	118
L4-N11-N6	65	65	65	L51-N15-N23	22	22	22
L5-N11-N15	100	100	100	L52-N2-N3	25	25	25
L6-N11-N7	64	64	64	L53-N2-N9	34	34	34
L7-N11-N21	7	7	7	L54-N9-N5	38	38	38
L8-N11-N17	2	2	2	L55-N5-N21	27	35	27
L9-N11-N14	9	9	9	L56-N13-N2	3	3	3
L10-N21-N20	48	48	48	L57-N10-N21	47	47	47
L11-N21-N18	44	44	44	L58-N11-N10	57	57	57
L12-N12-N13	14	14	14	L59-N16-N22	20	20	20
L13-N20-N24	22	22	22	L60-N1-N14	38	40	38
L14-N7-N15	99	162	99	L61-N8-N6	82	[$\gamma = 14$]	84
L15-N7-N6	170	170	170	L62-N17-N11	38	38	38
L16-N7-N17	4	4	4	L63-N12-N22	2	2	2
L17-N18-N2	4	4	4	L64-N12-N16	3	3	3
L18-N18-N5	26	26	26	L65-N13-N18	3	3	3
L19-N18-N10	18	18	18	L66-N13-N6	3	3	3
L20-N15-N24	60	60	60	L67-N13-N18	48	48	48
L21-N15-N23	3	3	3	L68-N13-N4	34	34	34
L22-N15-N19	3	3	3	L69-N11-N15	3	3	3
L23-N12-N7	27	27	27	L70-N11-N7	4	4	4
L24-N2-N3	2	2	2	L71-N11-N14	3	3	3
L25-N2-N9	35	35	35	L72-N21-N18	3	3	3
L26-N11-N18	45	45	45	L73-N20-N24	11	11	11
L27-N12-N13	82	82	82	L74-N7-N6	3	3	3
L28-N12-N1	28	28	28	L75-N7-N17	3	3	3
L29-N12-N16	30	30	30	L76-N18-N2	3	3	3
L30-N12-N4	30	33	30	L77-N18-N5	19	19	19
L31-N13-N2	87	87	87	L78-N13-N11	83	83	83
L32-N13-N1	49	49	49	L79-N18-N10	3	3	3
L33-N13-N6	93	93	93	L80-N15-N24	3	3	3
L34-N12-N22	27	27	27	L81-N15-N23	22	22	22
L35-N4-N8	27	27	27	L82-N15-N19	28	28	28
L36-N17-N6	44	[$\gamma = 3$]	44	L83-N2-N9	3	3	3
L37-N19-N15	111	[$\gamma = 0.666667$]	111	L84-N12-N22	3	3	3
L38-N11-N21	125	149	125	L85-N22-N2	6	6	6
L39-N11-N23	21	[$\gamma = 12$]	21	L86-N13-N14	15	15	15
L40-N11-N14	53	53	53	L87-N14-N11	3	3	3
L41-N11-N19	114	[$\gamma = 36$]	114	L88-N5-N9	3	3	3
L42-N21-N20	119	[$\gamma = 34$]	119	L89-N13-N6	6	6	6
L43-N20-N24	138	[$\gamma = 28.6667$]	138	L90-N16-N22	2	2	2
L44-N7-N8	75	[$\gamma = 13$]	82	L91-N22-N2	24	[$\gamma = 1$]	24
L45-N12-N16	13	13	13	L92-N5-N9	3	3	3
L46-N7-N17	73	73	73	L93-N1-N14	8	8	8
L47-N18-N2	87	87	87	L94-N14-N10	33	33	33

p30

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N23	375	[$\gamma = 2$]	[$\gamma = 2$]	L36-N29-N25	1360	[$\gamma = 31.8571$]	[$\gamma = 2$]
L2-N3-N21	370	[$\gamma = 153$]	[$\gamma = 44$]	L37-N29-N26	1360	[$\gamma = 31.8571$]	[$\gamma = 2$]
L3-N3-N45	370	[$\gamma = 153$]	[$\gamma = 44$]	L38-N30-N26	899	[$\gamma = 334$]	[$\gamma = 11.5$]
L4-N6-N32	424	[$\gamma = 78.5$]	[$\gamma = 34$]	L44-N35-N23	371	[$\gamma = 124$]	[$\gamma = 3$]
L5-N6-N40	414	[$\gamma = 88.5$]	414	L45-N35-N41	363	[$\gamma = 123$]	[$\gamma = 7$]
L7-N8-N23	725	[$\gamma = 161$]	[$\gamma = 34$]	L48-N36-N13	350	[$\gamma = 63$]	[$\gamma = 36$]
L8-N8-N38	53	[$\gamma = 23$]	53	L49-N36-N26	353	[$\gamma = 62$]	[$\gamma = 6$]
L9-N8-N42	670	670	670	L50-N37-N19	88	[$\gamma = 35$]	[$\gamma = 11$]
L10-N10-N19	94	[$\gamma = 36$]	[$\gamma = 11$]	L53-N42-N38	29	[$\gamma = 22$]	[$\gamma = 4$]
L11-N11-N27	92	[$\gamma = 35$]	[$\gamma = 6$]	L54-N42-N40	699	[$\gamma = 162$]	[$\gamma = 7$]
L12-N11-N37	84	[$\gamma = 35$]	[$\gamma = 10$]	L55-N43-N51	625	[$\gamma = 238$]	[$\gamma = 54$]
L13-N12-N50	586	[$\gamma = 238$]	[$\gamma = 9$]	L56-N44-N2	216	[$\gamma = 38$]	[$\gamma = 18$]
L15-N13-N10	344	[$\gamma = 63$]	[$\gamma = 45$]	L57-N46-N1	375	[$\gamma = 2$]	[$\gamma = 2$]
L16-N14-N4	213	[$\gamma = 44$]	[$\gamma = 19$]	L58-N46-N26	375	[$\gamma = 2$]	375
L17-N14-N16	213	[$\gamma = 44$]	[$\gamma = 18$]	L60-N47-N23	444	[$\gamma = 155$]	[$\gamma = 15$]
L20-N16-N40	227	[$\gamma = 44$]	[$\gamma = 18$]	L61-N47-N45	384	[$\gamma = 154$]	[$\gamma = 34$]
L23-N20-N12	559	[$\gamma = 238$]	[$\gamma = 45$]	L65-N50-N2	222	[$\gamma = 38$]	[$\gamma = 12$]
L24-N20-N43	627	[$\gamma = 238$]	[$\gamma = 54$]	L66-N50-N32	397	[$\gamma = 79$]	[$\gamma = 41$]
L25-N21-N51	619	[$\gamma = 238$]	[$\gamma = 53$]	L67-N52-N4	227	[$\gamma = 44$]	[$\gamma = 15$]
L26-N22-N44	208	[$\gamma = 38$]	[$\gamma = 21$]	L68-N52-N21	639	[$\gamma = 5$]	639
L27-N23-N25	1360	[$\gamma = 31.8571$]	[$\gamma = 1$]	L70-N52-N22	218	[$\gamma = 38$]	[$\gamma = 9$]
L30-N23-N28	358	[$\gamma = 52$]	358	L72-N52-N30	905	[$\gamma = 337$]	[$\gamma = 11$]
L34-N27-N26	171	[$\gamma = 1$]	171	L74-N54-N10	371	[$\gamma = 124$]	[$\gamma = 20$]
L35-N27-N28	338	[$\gamma = 51$]	[$\gamma = 33$]	L75-N54-N41	365	[$\gamma = 123$]	[$\gamma = 17$]

p44

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N3	1557	[$\gamma = 677.5$]	[$\gamma = 677.5$]	L15-N15-N2	1471	[$\gamma = 221.667$]	[$\gamma = 167.667$]
L2-N2-N11	1814	[$\gamma = 677.5$]	[$\gamma = 677.5$]	L16-N15-N10	453	[$\gamma = 160$]	[$\gamma = 160$]
L4-N4-N2	832	[$\gamma = 386.5$]	[$\gamma = 382.5$]	L17-N15-N12	453	[$\gamma = 276$]	[$\gamma = 52$]
L5-N4-N12	913	[$\gamma = 411$]	[$\gamma = 407$]	L18-N15-N13	1557	[$\gamma = 677.5$]	[$\gamma = 677.5$]
L7-N5-N12	499	[$\gamma = 217$]	[$\gamma = 217$]	L19-N15-N17	820	[$\gamma = 384.5$]	[$\gamma = 384.5$]
L8-N6-N5	499	[$\gamma = 217$]	[$\gamma = 217$]	L21-N16-N8	781	[$\gamma = 329$]	[$\gamma = 228.5$]
L9-N6-N12	439	[$\gamma = 128$]	[$\gamma = 124$]	L22-N19-N7	475	[$\gamma = 242$]	[$\gamma = 86.5$]
L10-N8-N12	1467	[$\gamma = 648.5$]	[$\gamma = 575.5$]	L24-N19-N20	1116	[$\gamma = 328.5$]	[$\gamma = 282.5$]
L11-N9-N12	1644	[$\gamma = 748$]	[$\gamma = 712$]	L25-N20-N11	1332	[$\gamma = 436.5$]	[$\gamma = 398.5$]
L12-N13-N3	1557	[$\gamma = 677.5$]	[$\gamma = 677.5$]	L26-N1-N7	958	[$\gamma = 378$]	[$\gamma = 360$]
L13-N14-N9	1042	[$\gamma = 448$]	[$\gamma = 411$]	L28-N10-N17	373	[$\gamma = 161$]	[$\gamma = 160$]
L14-N14-N16	362	[$\gamma = 322$]	[$\gamma = 214$]				

p45

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N3	453	[$\gamma = 440$]	[$\gamma = 270$]	L14-N14-N16	531	[$\gamma = 419$]	[$\gamma = 260$]
L2-N2-N11	869	[$\gamma = 445$]	[$\gamma = 359$]	L16-N15-N10	195	[$\gamma = 36$]	[$\gamma = 24$]
L4-N4-N2	1206	[$\gamma = 577.5$]	[$\gamma = 567.5$]	L17-N15-N12	1014	[$\gamma = 479.5$]	[$\gamma = 337.5$]
L5-N4-N12	1202	[$\gamma = 575.5$]	[$\gamma = 565.5$]	L18-N15-N13	453	[$\gamma = 440$]	[$\gamma = 270$]
L7-N5-N12	156	[$\gamma = 156$]	[$\gamma = 156$]	L19-N15-N17	136	[$\gamma = 37$]	[$\gamma = 32$]
L8-N6-N5	156	[$\gamma = 156$]	[$\gamma = 156$]	L21-N16-N8	557	[$\gamma = 396$]	[$\gamma = 266$]
L9-N6-N12	196	[$\gamma = 156$]	[$\gamma = 82$]	L22-N19-N7	666	[$\gamma = 445$]	[$\gamma = 266$]
L10-N8-N12	673	[$\gamma = 394$]	[$\gamma = 311$]	L24-N19-N20	932	[$\gamma = 441$]	[$\gamma = 374.5$]
L11-N9-N12	453	[$\gamma = 394$]	[$\gamma = 238$]	L25-N20-N11	922	[$\gamma = 444$]	[$\gamma = 367.5$]
L12-N13-N3	453	[$\gamma = 440$]	[$\gamma = 270$]	L26-N1-N7	563	[$\gamma = 444$]	[$\gamma = 270$]
L13-N14-N9	453	[$\gamma = 395$]	[$\gamma = 266$]	L28-N10-N17	77	[$\gamma = 37$]	[$\gamma = 34$]

p69

LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES	LINK	INITIAL FLOW	DIVERS. CHANGES	NO DIVERS. CHANGES
L1-N1-N13	276	[$\gamma = 7.75$]	294	L36-N39-N47	257	[$\gamma = 7.2$]	279
L2-N1-N23	198	[$\gamma = 170$]	[$\gamma = 48$]	L37-N41-N49	255	[$\gamma = 57$]	[$\gamma = 15.5$]
L3-N1-N25	275	[$\gamma = 10.9091$]	317	L38-N43-N53	167	[$\gamma = 28$]	[$\gamma = 1$]
L4-N1-N27	229	[$\gamma = 32$]	229	L39-N45-N51	173	[$\gamma = 130$]	[$\gamma = 27$]
L5-N3-N7	174	[$\gamma = 56.5$]	[$\gamma = 2$]	L40-N47-N55	387	[$\gamma = 39.25$]	439
L6-N3-N43	174	[$\gamma = 56.6667$]	[$\gamma = 8.66667$]	L41-N49-N53	167	[$\gamma = 28$]	[$\gamma = 1$]
L7-N5-N29	243	[$\gamma = 182$]	[$\gamma = 38$]	L42-N1-N2	88	[$\gamma = 88$]	[$\gamma = 88$]
L8-N5-N31	243	[$\gamma = 182$]	[$\gamma = 8$]	L43-N3-N4	36	[$\gamma = 36$]	[$\gamma = 36$]
L9-N7-N15	114	[$\gamma = 7.6$]	114	L44-N5-N6	69	[$\gamma = 69$]	[$\gamma = 69$]
L10-N7-N51	126	[$\gamma = 17.5$]	[$\gamma = 5.33333$]	L45-N7-N8	28	[$\gamma = 28$]	[$\gamma = 28$]
L11-N9-N17	173	[$\gamma = 130$]	[$\gamma = 27$]	L46-N9-N10	104	[$\gamma = 104$]	[$\gamma = 104$]
L12-N9-N25	352	[$\gamma = 62$]	[$\gamma = 48$]	L47-N11-N12	59	[$\gamma = 59$]	[$\gamma = 59$]
L13-N9-N35	138	139	138	L48-N13-N14	55	[$\gamma = 55$]	[$\gamma = 55$]
L14-N9-N41	178	[$\gamma = 1$]	178	L49-N15-N16	38	[$\gamma = 38$]	[$\gamma = 38$]
L15-N9-N49	31	31	31	L50-N17-N18	39	[$\gamma = 39$]	[$\gamma = 39$]
L16-N11-N31	243	[$\gamma = 182$]	[$\gamma = 18.6667$]	L51-N19-N20	29	[$\gamma = 29$]	[$\gamma = 29$]
L17-N11-N39	243	[$\gamma = 182$]	[$\gamma = 46.6667$]	L52-N21-N22	105	[$\gamma = 105$]	[$\gamma = 105$]
L18-N13-N21	382	[$\gamma = 36$]	[$\gamma = 4.5$]	L53-N23-N24	141	[$\gamma = 141$]	[$\gamma = 141$]
L19-N13-N39	262	[$\gamma = 2$]	262	L54-N25-N26	101	[$\gamma = 101$]	[$\gamma = 101$]
L20-N15-N41	291	[$\gamma = 73.3333$]	[$\gamma = 62$]	L55-N27-N28	156	[$\gamma = 156$]	[$\gamma = 156$]
L21-N15-N51	191	[$\gamma = 36.5$]	[$\gamma = 32$]	L56-N29-N30	65	[$\gamma = 65$]	[$\gamma = 65$]
L22-N17-N37	173	[$\gamma = 130$]	[$\gamma = 7$]	L57-N31-N32	63	[$\gamma = 63$]	[$\gamma = 63$]
L23-N19-N23	198	[$\gamma = 170$]	[$\gamma = 23$]	L58-N33-N34	106	[$\gamma = 106$]	[$\gamma = 106$]
L24-N19-N27	198	[$\gamma = 170$]	[$\gamma = 48$]	L59-N35-N36	109	[$\gamma = 109$]	[$\gamma = 109$]
L25-N21-N25	223	256	223	L60-N37-N38	38	[$\gamma = 38$]	[$\gamma = 38$]
L26-N21-N35	329	[$\gamma = 48$]	[$\gamma = 48$]	L61-N39-N40	66	[$\gamma = 66$]	[$\gamma = 66$]
L27-N21-N47	264	[$\gamma = 5.6$]	264	L62-N41-N42	57	[$\gamma = 57$]	[$\gamma = 57$]
L28-N27-N39	345	[$\gamma = 263$]	[$\gamma = 17.3333$]	L63-N43-N44	93	[$\gamma = 93$]	[$\gamma = 93$]
L29-N29-N39	31	31	31	L64-N45-N46	54	[$\gamma = 54$]	[$\gamma = 54$]
L30-N29-N55	246	[$\gamma = 212$]	[$\gamma = 46.6667$]	L65-N47-N48	71	[$\gamma = 71$]	[$\gamma = 71$]
L31-N33-N35	208	[$\gamma = 1.42857$]	208	L66-N49-N50	47	[$\gamma = 47$]	[$\gamma = 47$]
L32-N33-N43	241	[$\gamma = 14.8$]	[$\gamma = 3.6$]	L67-N51-N52	113	[$\gamma = 113$]	[$\gamma = 113$]
L33-N33-N55	361	[$\gamma = 67$]	[$\gamma = 61.3333$]	L68-N53-N54	28	[$\gamma = 28$]	[$\gamma = 28$]
L34-N35-N49	291	[$\gamma = 46$]	[$\gamma = 19.6$]	L69-N55-N56	58	[$\gamma = 58$]	[$\gamma = 58$]
L35-N37-N45	173	[$\gamma = 131$]	[$\gamma = 7$]				

A.2 Adding new demands

ADDING NEW DEMANDS – 90% MAX. INITIAL LINK LOAD

NET	MAX NEW [%]	DEMANDS	VALUES	ACTUAL NEW [%]	ACTUAL NEW [ABS.]	OPT. FRAC. SOLUTION	CHANGES	TIME
p01	10	436	862	1.89	16	16	16	00:00:00
	20	458	906	7.09	60	60	60	00:00:00
	30	516	1022	20.80	176	186	186	00:00:01
	40	538	1066	26.00	220	[$\gamma = 2$]	—	00:00:01
	50	637	1264	49.41	418	[$\gamma = 11$]	—	00:00:01
p07	10	829	1658	7.11	110	110	110	00:00:00
	20	909	1818	17.44	270	270	270	00:00:00
	30	988	1976	27.65	428	428	428	00:00:00
	40	1068	2136	37.98	588	590	590	00:00:01
	50	1160	2320	49.87	772	[$\gamma = 0.5$]	—	00:00:01
p09	10	990	1980	6.68	124	124	124	00:00:00
	20	1081	2162	16.49	306	306	306	00:00:02
	30	1175	2350	26.62	494	494	494	00:00:06
	40	1266	2532	36.42	676	676	676	00:00:11
	50	1391	2782	49.89	926	926	926	00:00:22
p20	10	897	1794	4.06	70	70	70	00:00:00
	20	972	1944	12.76	220	220	220	00:00:00
	30	1067	2134	23.78	410	410	410	00:00:01
	40	1142	2284	32.48	560	560	560	00:00:02
	50	1292	2584	49.88	860	[$\gamma = 0.444444$]	—	00:00:02
p30	10	2823	4354	4.84	201	201	201	00:00:00
	20	3059	4704	13.27	551	551	551	00:00:01
	30	3273	5028	21.07	875	875	875	00:00:03
	40	3592	5533	33.23	1380	1385.5	1386	00:00:17
	50	3894	5993	44.31	1840	1894.5	1895	00:00:32
p44	10	8782	9672	2.10	199	199	199	00:00:00
	20	9002	9972	5.27	499	499	499	00:00:01
	30	9221	10276	8.48	803	824	824	00:00:02
	40	9443	10578	11.66	1105	1152	1152	00:00:03
	50	9765	11022	16.35	1549	1637	1637	00:00:12
	60	9916	11225	18.49	1752	1752	1752	00:00:00
p45	10	2843	3950	7.66	281	281	281	00:00:00
	20	3086	4295	17.06	626	[$\gamma = 20$]	—	00:00:00
	30	3331	4638	26.41	969	[$\gamma = 63$]	—	00:00:00
	40	3574	4983	35.81	1314	[$\gamma = 106.5$]	—	00:00:00
	50	3874	5413	47.53	1744	[$\gamma = 156$]	—	00:00:00
p69	10	3211	4221	4.69	189	189	189	00:00:00
	20	3475	4544	12.70	512	512	512	00:00:00
	30	3724	4842	20.09	810	810	810	00:00:01
	40	4059	5306	31.60	1274	[$\gamma = 13.1429$]	—	00:00:01
	50	4403	5790	43.60	1758	[$\gamma = 42$]	—	00:00:01

A.3 Shortening routing paths

SHORTENING PATH – 80% MAX. INITIAL LINK LOAD							
NETWORK	SP+	SKIPPED INIT. PATHS	PATH LIMIT	LIMIT REACHED	OPT.FRAC. SOLUTION	CHANGES	SOLUTION TIME
p01	3	0	100	0	0	0	00:00:00
	2	10	100	0	10	10	00:00:00
	1	INITIALIZER DETECTED INFEASIBILITY					
p07	6	0	100	0	0	0	00:00:00
	5	2	100	0	2	2	00:00:00
	4	8	100	0	8	8	00:00:00
	3	INITIALIZER DETECTED INFEASIBILITY					
p09	7	0	100	0	0	0	00:00:00
	6	11	100	0	11	11	00:00:00
	5	21	100	0	21	21	00:00:00
	4	45	100	0	45	45	00:00:00
	3	104	100	0	104	104	00:00:10
	2	INITIALIZER DETECTED INFEASIBILITY					
p20	4	0	100	0	0	0	00:00:00
	3	10	100	0	10	10	00:00:00
	2	32	100	4	32	32	00:00:27
	1	INITIALIZER DETECTED INFEASIBILITY					
p30	10	45	100	0	45	45	00:00:00
	9	50	100	0	50	50	00:00:00
	8	202	100	0	202	202	00:00:00
	7	321	100	0	321	321	00:00:00
	6	INITIALIZER DETECTED INFEASIBILITY					
p44	9	0	100	0	0	0	00:00:00
	8	4	100	0	4	4	00:00:00
	7	4	100	0	4	4	00:00:00
	6	INITIALIZER DETECTED INFEASIBILITY					
p45	9	0	100	0	0	0	00:00:00
	8	INITIALIZER DETECTED INFEASIBILITY					
p69	10	9	100	0	9	9	00:00:00
	9	15	100	0	15	15	00:00:00
	8	INITIALIZER DETECTED INFEASIBILITY					

A.4 Link load reduction

LINK LOAD REDUCTION – 80% MAX. INITIAL LINK LOAD

NET	MAX CHNG	ACT. CHNG	LINK LOAD	GAP [%]	SOLUTION TIME	BnB NODES IT 1	IT 2	IT 3	IT 4	IT 5	IT 6
p01	0		0.796875								
	20	19	0.796875	1.90	01:51:19	2486	568	467	815	706	4451
	30	29	0.78125	0.00	01:15:25	1112	1009	857	889	1210	3314
	50	50	0.78125	0.00	00:15:00	1109	1111	41	0	0	41
	10000	120	0.78125	0.00	00:00:03	14	0	0	0	0	14
p07	0		0.796875								
	20	19	0.792969	0.84	01:50:24	2040	2438	1893	2318	2139	13466
	30	30	0.792969	0.84	01:50:33	2305	1778	1556	1516	2001	9247
	50	49	0.792969	0.84	01:50:37	1820	1342	1268	1177	1133	7802
	10000	52	0.792969	0.84	01:50:26	1979	1523	1526	1587	1495	8850
p09	0		0.75								
	20	20	0.75	11.63	01:21:04	40054	40003	0	0	0	246680
	30	28	0.75	17.98	01:51:04	40063	37494	37485	37634	41680	226641
	50	40	0.625	6.01	01:50:44	11055	10660	10508	10099	10219	57796
	10000	667	0.3125	8.60	01:51:19	723	919	888	546	609	1648
p20	0		0.796875								
	20	20	0.703125	1.47	01:50:47	40373	16223	17217	17175	14092	168621
	30	29	0.6875	2.62	01:51:01	14332	14308	18612	15463	16865	101700
	50	50	0.65625	2.48	01:50:31	1907	1699	1653	1607	1675	9774
	10000	200	0.640625	1.21	01:50:22	2247	2069	2023	1838	2010	11011
p30	0		0.753968								
	20	20	0.714286	0.00	00:00:02	1	0	0	0	0	1
	30	30	0.694444	0.00	00:00:02	1	0	0	0	0	1
	50	50	0.660714	0.08	00:00:08	4	0	0	0	0	4
	10000	1115	0.412698	0.32	01:50:58	500	480	472	458	463	2876
p44	0		0.799603								
	20	20	0.793651	0.00	00:00:23	1	0	0	0	0	1
	30	30	0.793651	0.00	00:00:51	6	0	0	0	0	6
	50	50	0.793651	0.00	00:00:23	1	0	0	0	0	1
	10000	261	0.793651	0.00	00:00:22	1	0	0	0	0	1
p45	0		0.799603								
	20	9	0.781746	0.00	00:00:02	1	0	0	0	0	1
	30	9	0.781746	0.00	00:00:02	1	0	0	0	0	1
	50	9	0.781746	0.00	00:00:02	1	0	0	0	0	1
	10000	9	0.781746	0.00	00:00:02	1	0	0	0	0	1
p69	0		0.78								
	20	19	0.760714	0.25	01:51:18	396	226	253	243	191	1846
	30	30	0.757143	0.24	01:51:34	120	116	102	117	96	797
	50	49	0.757143	0.24	01:41:05	115	103	100	105	0	928
	10000	182	0.757143	0.24	01:50:58	141	138	137	138	138	689

LINK LOAD REDUCTION – 80% MAX. INITIAL LINK LOAD (NO DIVERSIFICATION)											
						BnB NODES					
NET	MAX CHNG	ACT. CHNG	LINK LOAD	GAP [%]	SOLUTION TIME	IT 1	IT 2	IT 3	IT 4	IT 5	IT 6
p01	0		0.796875								
	20	20	0.78125	0.00	01:08:06	4352	1952	2267	1947	2015	2776
	30	30	0.78125	0.00	00:00:57	231	58	0	0	0	58
	50	50	0.78125	0.00	00:00:02	14	0	0	0	0	14
	10000	129	0.78125	0.00	00:00:01	8	0	0	0	0	8
p07	0		0.796875								
	20	20	0.785156	0.66	01:50:37	5052	5149	4472	4630	4807	32685
	30	30	0.78125	0.78	01:50:34	2577	4447	4256	3601	3480	23509
	50	50	0.769531	0.41	01:50:27	2931	3756	3594	3835	3674	22327
	10000	262	0.765625	0.76	01:40:18	3553	5318	3904	0	0	15619
p09	0		0.75								
	20	20	0.75	11.63	01:50:25	101573	100962	96608	93161	97973	703873
	30	26	0.75	17.98	01:51:10	67523	66326	71568	64926	61080	480087
	50	43	0.625	6.01	01:40:20	53845	71815	71129	71182	0	483759
	10000	815	0.25	4.83	01:50:38	1853	3303	2778	2672	2372	16330
p20	0		0.796875								
	20	20	0.703125	1.47	01:50:13	63388	54677	59268	60999	47617	473450
	30	30	0.679688	1.46	01:50:33	38550	47831	40575	40588	56489	300094
	50	49	0.640625	0.66	01:50:25	6855	7888	6714	7474	19213	101976
	10000	278	0.609375	1.38	01:50:21	2844	2636	3064	2809	3153	18323
p30	0		0.753968								
	20	20	0.714286	0.00	00:00:01	5	0	0	0	0	5
	30	30	0.694444	0.00	00:00:01	1	0	0	0	0	1
	50	50	0.660714	0.08	00:00:03	4	0	0	0	0	4
	10000	1435	0.345238	0.29	01:30:27	659	665	664	0	0	4057
p44	0		0.799603								
	20	20	0.787698	0.03	00:00:19	4	0	0	0	0	4
	30	29	0.78373	0.08	01:50:34	374	809	881	871	883	2794
	50	50	0.774802	0.06	01:30:27	519	564	564	0	0	3259
	10000	731	0.767857	0.03	00:00:52	6	0	0	0	0	6
p45	0		0.799603								
	20	20	0.759921	0.00	00:00:00	1	0	0	0	0	1
	30	30	0.740079	0.00	00:00:00	1	0	0	0	0	1
	50	50	0.700397	0.00	00:00:00	1	0	0	0	0	1
	10000	1085	0.456349	0.00	00:00:04	5	0	0	0	0	5
p69	0		0.78								
	20	20	0.758929	0.16	01:50:31	445	409	323	336	346	2775
	30	30	0.753125	0.25	01:50:38	340	282	255	278	249	2511
	50	50	0.7425	0.22	01:50:31	275	232	233	242	360	2180
	10000	779	0.665625	0.11	01:30:24	318	317	319	0	0	1915

LINK LOAD REDUCTION – 90% MAX. INITIAL LINK LOAD (NO DIVERSIFICATION)

NET	MAX CHNG	ACT. CHNG	LINK LOAD	GAP [%]	SOLUTION TIME	BnB NODES IT 1	IT 2	IT 3	IT 4	IT 5	IT 6
p01	0		0.890625								
	20	20	0.84375	0.28	01:50:49	6301	6558	5009	5193	4990	28580
	30	30	0.828125	0.32	01:50:47	5039	4956	4227	3913	3876	23048
	50	49	0.8125	1.56	01:51:01	4835	5625	4513	4148	4332	24631
	10000	150	0.78125	0.00	00:02:08	63	1017	85	0	0	85
p07	0		0.875								
	20	20	0.78125	2.85	01:40:34	74927	88045	141353	141368	0	979318
	30	30	0.734375	1.48	01:30:19	65313	108031	94268	0	0	508714
	50	48	0.6875	1.24	01:30:11	20283	29461	30972	0	0	181558
	10000	108	0.6875	1.85	01:30:06	17314	17666	22616	0	0	117664
p09	0		0.875								
	20	20	0.833333	6.82	01:50:22	50549	64116	66249	55690	56604	336358
	30	30	0.833333	11.61	01:50:41	112172	125139	121910	125734	123955	793592
	50	44	0.75	6.87	01:50:22	35898	56755	55654	52432	55103	348579
	10000	680	0.375	10.53	01:40:28	2220	1958	1527	1686	0	14154
p20	0		0.890625								
	20	17	0.8125	1.42	01:20:11	42533	47833	0	0	0	286176
	30	30	0.78125	0.50	01:50:14	48894	37536	38673	38181	37999	224351
	50	49	0.75	0.84	01:50:13	21108	21165	22228	22393	22432	154069
	10000	350	0.640625	1.65	01:50:33	1981	4161	3306	4374	3278	22452
p30	0		0.880952								
	20	20	0.84127	0.00	00:00:02	1	0	0	0	0	1
	30	30	0.821429	0.00	00:00:02	1	0	0	0	0	1
	50	50	0.781746	0.00	00:00:03	1	0	0	0	0	1
	10000	1073	0.553571	0.18	01:30:46	156	155	155	0	0	924
p44	0		0.899802								
	20	20	0.889881	0.00	00:00:12	3	0	0	0	0	3
	30	30	0.884921	0.00	00:00:12	1	0	0	0	0	1
	50	50	0.875	0.00	00:00:12	1	0	0	0	0	1
	10000	762	0.83631	0.03	00:01:16	6	0	0	0	0	6
p45	0		0.89881								
	20	20	0.878968	0.00	00:00:01	1	0	0	0	0	1
	30	30	0.869048	0.00	00:00:01	1	0	0	0	0	1
	50	50	0.849206	0.00	00:00:01	1	0	0	0	0	1
	10000	306	0.650794	0.06	00:00:13	6	0	0	0	0	6
p69	0		0.89								
	20	20	0.857143	0.24	01:50:38	482	892	847	618	641	3866
	30	28	0.85	0.37	01:50:43	457	304	611	360	358	3612
	50	50	0.8375	0.33	01:50:57	182	175	165	186	261	1677
	10000	393	0.795	0.18	01:20:22	310	307	0	0	0	1813

List of Abbreviations

ATM	Asynchronous Transportation Module	page 6
B-ISDN	Broadband Integrated Services Digital Network	page 6
DMP	dual master program	page 33
DRMP	dual restricted master program	page 33
IP	integer linear program	page 19
KSP	K shortest paths	page 58
MIP	mixed-integer linear program	page 19
MP	master program	page 33
NOS	Normal Operating State	page 8
PDH	Plesiochronous Digital Hierarchy	page 5
RMP	restricted master program	page 33
SDH	Synchronized Digital Hierarchy	page 5
SNCP	Subnetwork Connection Protection	page 9
STM	Synchronous Transport Modules	page 6
VC	Virtual Container	page 7

List of Figures

List of Tables

Bibliography

[BHV00] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.*, 48(2):318–326, 2000.

[BS95] A. W. Brander and Mark C. Sinclair. A comparative study of k-shortest path algorithms. In *Proc. 11th UK Performance Engineering Worksh. for Computer and Telecommunications Systems*, September 1995.

[Chv83] V. Chvátal. *Linear programming.* A series of books in the mathematical sciences. New York - San Francisco: W. H. Freeman and Company, 1983.

[DL05] J. Desrosiers and M.E. Lübbecke. A primer in column generation. In G.Desaulniers, J.Desrosiers, and M.M. Solomon, editors, *Column Generation.* Kluwer Academic Publishers, Boston, MA, 2005.

[Epp94] David Eppstein. Finding the k shortest paths. In *IEEE Symposium on Foundations of Computer Science*, pages 154–165, 1994.

[Grö04] M. Grötschel. *Lineare Optimierung, Skriptum zur Vorlesung im WS 2003/2004.* Technische Universität Berlin, 2004. Version vom 13. Januar 2004.

[Kor02] Bernhard Korte. *Combinatorial optimization: theory and algorithms.* Algorithm and combinatorics. Berlin - Heidelberg - New York: Springer, 2002.

[Krö03] Alexander Kröller. Network optimization: Integration of hardware configuration and capacity dimensioning. diploma thesis, Technische Universität Berlin, 2003.

[Kya93] Othmar Kyas. *ATM-Netzwerke: Aufbau, Funktion, Performance.* DATACOM-Fachbuchreihe. Bergheim: DATACOM-Verlag, 1993.

[LD02] M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. Les Cahiers du GERAD G-2002-64, HEC Montréal, Canada, 2002. To appear in *Oper. Res.*

[MPS98] E.Q.V. Martins, M.M.B. Pascoal, and J.L.E. Santos. The k shortest loopless paths problem, 1998.

[Orl03] Sebastian Orlowski. Local and global restoration of node and link failures in telecommunication networks. diploma thesis, Technische Universität Berlin, February 2003.

[SA05] ILOG SA. *ILOG CPLEX 9.1 Reference Manual*. 2005.

[Sie93] Gerd Siegmund. *ATM — Die Technic des Breitband-ISDN*. Kommunikation & Technik bei R. v. Decker. Heidelberg: v. Decker, 1993.

[Wes00] R. Wessäly. *DImensioning Survivable Capacitated NETworks*. PhD thesis, ZIB and TU Berlin, 2000.

[Wol98] L.A. Wolsey. *Integer Programming*. New York: John Wiley and Sons, 1998.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt die selbstständige und eigenhändige Anfertigung dieser Diplomarbeit.

(Thomas Thevis)