Evolutionary Design of Corporate Networks under Uncertainty

Volker Gerd Fischer

Institut für Informatik
der Technischen Universität München
Lehrstuhl für Informatik VIII
Rechnerkommunikation und maschinelle Deduktion

Evolutionary Design of Corporate Networks under Uncertainty

Volker Gerd Fischer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Chr. Zenger
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. E. Jessen
2.	Univ.-Prof. Dr. J. Eberspächer

Die Dissertation wurde am 03.04.2000 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 26.06.2000 angenommen.

Contents

1 Motivation and Outline

Motivation

Designing Corporate Telecommunication Networks[1] is a critical task for every company which has to rely on telecommunication services. Not only do the costs of the network play a major role, but the survival of the whole company may depend on it. It is a known fact that most of the fortune 100 companies would go bankrupt if their data communication fails for more than 48 hours.

However, optimizing corporate networks to satisfy the demands of a company while keeping the expenses low is an difficult problem. Nowadays, network design is subject to a rapidly changing planning environment: New improved network technologies are becoming available in short intervals, the deregulated market and competition leads to a variety of different tariffs and services. Moreover, traffic patterns and characteristics in data communication change considerably from day to day, while new applications (such as the World-Wide-Web only five years ago) can introduce new network traffic characteristics and overall traffic volume still shows exponential growth.

More specifically, today there is an almost exclusive demand for IP services in data networks. Data traffic is expected to outgrow voice traffic in 2002 [Odl98a]. Many analysts expect that exponential growth in data communication will continue for the next 10 years at least. The characteristics of multimedia traffic may enforce the introduction of next generation protocols and hardware, while the ability of efficiently multiplexing traffic streams still remains unclear.

Network design (sometimes also referred to as **network planning**), the task of planning and managing communication networks, comprises a variety of techniques and knowledge evolving from many different fields of science. These sciences include optimization, theory of graphs, forecasting, simulation & modeling, knowledge representation, decision theory, theory of finance, electrical engineering, and computer science. In general, one could classify network design as *Operations Research* and model network design problems as *Mathematical Programs*.

However, the designer often has only very vague ideas about the environment in which the network design takes place. Sometimes not even the traffic requirements between the network nodes are known. This **incomplete knowledge** has to be transformed into a proper model that can be tackled by one of the techniques mentioned above.

Even if there is perfect knowledge about a present-day network design available, the rapidly changing environment makes it necessary to predict and take into account developments of the future. An optimally designed network being in operation now can fail badly after only a few months if the designer does not take into account the future developments. Therefore, every responsible designer has to concern himself with how to incorporate incomplete knowledge in his[2] planning task.

The purpose of this thesis is to give an overview of the current state of the art of network design and to address the problem of how to efficiently incorporate incomplete knowledge. Furthermore, the task of designing an optimized data network over time in a rapidly changing environment, which is an unsolved as well as mainly unaddressed problem, is tackled.

[1] A corporate telecommunication network is a company wide telecommunication network delivering different kinds of services like telephone and IP. From here on the term **network** will denote a corporate telecommunication network.

[2] respectively her

Outline of the Thesis

Chapter 2 introduces the concepts and the terminology necessary to formulate the "static", "incremental", and "evolutionary" (i.e. network planning over time) telecommunication network design problem and gives an overview of the dimensions of planning involved in network design problems.

A decision theoretic framework for network design problems which is capable of coping with all types of network design problems is introduced in Chapter 3. This framework structures the design problem in an appropriate way, providing the designer with uniform guidelines about how to proceed in an arbitrary planning environment, and is capable of incorporating incomplete knowledge.

Especially when dealing with evolutionary network design problems, the incorporation of uncertainty (e.g. about the development of traffic growth) becomes the key part in the design process. A thorough examination and classification of uncertainties in network design and their "resolution" is the topic of Chapter 4. Mathematical techniques that are able to incorporate uncertainties are reviewed with respect to their qualification to deal with (evolutionary) network design problems.

Resulting from the theoretical concepts, an object-oriented application framework for telecommunication network design (*FooNet*) is presented in Chapter 5. *FooNet* supports code and design reuse on different levels of abstraction. Furthermore, it makes use of XML (Extended Modeling Language) as an application independent data exchange format.

The first part of Chapter 6 gives an overview and assessment of heuristic optimization algorithms with respect to network design problems. It is empirically justified that meta-heuristic search algorithms are a promising solution technique. The second part of the chapter exemplarily applies the concepts by the example of a realistic setting: the German Wissenschaftsnetz.

Chapter 7 concludes the thesis with a discussion of the results obtained and gives an outlook on future work.

The appendices provide supplementary material on existing planning algorithms (Appendix A), available uncertainty formalisms (Appendix B), mathematics of corporate finance (Appendix C), as well as some notes on UML (Appendix D).

Acknowledgements

I am indebted to many people for support, guidance, and advice during the course of my research. First of all, I want to thank my advisor Prof. Dr.-Ing. Eike Jessen for giving me the opportunity to work in one of the most interesting fields of computer science and for the support of this work. Furthermore, I want to thank Prof. Dr.-Ing. Jörg Eberspächer for his support and willingness evaluate this thesis. I also want to express my gratitude to the DFN-Verein (especially the B-WiN Lab in Erlangen) for supporting me with the necessary data.

Several colleagues have contributed to this research through important discussions and valuable comments, particularly Thomas Erlebach (by the way, good luck in Switzerland), Manfred Jobmann, Hans-Peter Schwefel, Helmut Gogl, Gernot Riegert, and Christian Kreibich. Thanks also to Christian Schwingenschlögl, Jochen Frings, and Hemant Sharma for reading the half-finished texts and "lending me their ears".

Last but not least, I am grateful to my parents, Heidi & Peter, and my wife Sylvia for their support and for reminding me that there is a life beside the doctoral thesis. The thesis is dedicated to them.

München, March 2000

Volker Gerd Fischer

2 Overview of Network Design

2.1 Overview

The author wants to begin this thesis with a quotation taken from a treatise of the International Telecommunication Union (ITU) about network planning [ITU83]:

> "Network planning consists of the use of scientific methods for optimizing investments and for dimensioning equipment in a unified way ...
> Thus the prime definition of network planning is to provide the right equipment, at the right place, at the right time, and the right cost in order to satisfy expected demand and give an acceptable grade of service."

There are a lot of definitions of the term "network design" available that heavily depend on the context in which the term is used. Here the following definition[3] (adapted from [HHMK89]) will be used:

Definition of Concept 1 (Network Design)
Network design *(also sometimes referred to as* ***network planning****) deals with the initial setup of a network as well as the ongoing reconfiguration and redesign necessary to accommodate traffic, services and technology to satisfy demands imposed by the users of the network together with optimization of the network towards a certain goal (such as costs or performance).*

Furthermore, four different design tasks will be distinguished:

- **Network Operation Planning** (*NOP*): aims at supporting network provision activities and detailed resource planning. Its emphasis is on day-to-day planning activities and reactions to small changes in the environment that have to be monitored on an appropriate time-scale and detail. In the future, so called *self-sizing* or *self-organizing* networks [↑][4] may have the ability to adapt to changing situations by configuring themselves to some degree. *NOP* will not be subject to further research in this thesis.
- **Static Network Design** (*SND*): denotes the network design process on the "green field", i.e. the design of a network without taking into account large-scale traffic variations, or an already deployed[5]network. Many of the existing network design algorithms deal with static network problems. A formal treatise is given in Section 2.4.
- **Incremental Network Design** (*IND*): deals with the problem of how an existing network should be upgraded in order to meet an altered environment. In most cases an upgrade is necessary due to increased traffic requirements. *IND* will be discussed in Section 2.5.
- **Strategic** or **Evolutionary Network Design** (*END*): takes into account the medium and long term aspects and focuses on the fundamental structure of the network. *END* is more function and technology driven and covers aspects including [GB96]:
  - network architecture
  - network traffic prediction
  - required transport network structure and equipment

[3]The term definition will not only be used in a strict mathematical sense, but also as a classification of terms and an introduction of concepts.
[4]The symbol [↑] means that there is an entry about this subject in the glossary.
[5]The facilities that are already installed in an existing network design are denoted as **deployed facilities**.

- identifying network evolution strategies.

END will be covered in Section 2.6 and Chapter 4.

The following sections will give deeper insights in the problems involved. Structuring the planning process will result in guidelines for solving network design problems and finally lead to a tractable framework of modeling current problems in network design.

2.2 Corporate Networks

This thesis focuses on the design of corporate networks. Corporate networks are built to satisfy data & multimedia (including voice and video) communication requirements (i.e. **traffic**) of an enterprise. Depending on the size, the geographical locations of the departments, and the deployment of network services in that enterprise its corporate network may have to connect up to a few hundred sites via wide area connections. The business trend is toward interconnection of all company resources into a seamless enterprise wide network (*Intranet* [↑]). The paradigm is often called "the corporation is the network" [MS99]. However, such interconnections can be prohibitively expensive unless set up correctly.

In contrast to public networks provided by national telephone companies (*Carriers* or *BCs* [↑]) or Internet Service Providers (*ISPs* [↑]), corporate networks do not make a measurable "revenue" but are designed to meet the enterprise's goals. Designers of corporate networks may rent or lease services from BCs or ISPs, however, rarely own wide area network facilities themselves. The ultimate goal of a corporate network is to meet the requirements while keeping expenses low.

One peculiarity of designing corporate networks lies in the condition that the planner **has** to meet all traffic requirements with a certain QoS [↑]. No shortage of productivity due to congestion in the network is accepted[6]. This situation differs from that of e.g. Internet Service Providers which have to balance the yield against the costs, often tolerating not to meet some requirements.

It has to be mentioned that there is no unique corporate network design problem since every enterprise has different requirements. A financial institution almost surely has other requirements on than a military or a scientific one.

An overview of the recent development of Internet architectures is given in [MS99]. Five generations of corporate networks have been deployed in the last decades:

First-generation corporate networks came into usage in the early 80s typically employing low speed (from 2.4 to 9.6 kbps) analog lines to support mission-critical functions such as database access. The network was realized by point-to-point lines and different departments of the company usually used different networks.

Second-generation corporate networks began to use digital transmission facilities (such as T1/E1-lines [↑SDH]) and started in the mid-80s. Considerations for cost efficiency led to transporting aggregated data over backbone networks taking into account economy of scale considerations. Still there was a strict separation of voice and data.

Third-generation corporate networks at the end of the 80s introduced the idea of packet switching and packet routing instead of (synchronous) multiplexing. This was founded on the hope for additional cost effectiveness in form of statistical multiplexing while the demand was rapidly growing.

[6] This does not mean that no congestion does ever appear in the realized network, only that the planner is not allowed to deliberately ignore requirements, e.g. due to higher cost.

Fourth-generation corporate networks came into being as the traffic load and delay of the existing networks became prohibitive. The demand of high-end throughput while keeping the connectivity [↑] of the network small led to the introduction of frame-relay services at the beginning of the 90s.

Fifth-generation networks became necessary when the bandwidth demand was measured in hundreds of Megabits per second, leading to new technologies such as ATM [↑] or Gigabit [↑Gbps] IP routers. This development started in the mid 90s and is still continuing. In the future, ISPs and BCs will provide services such as VPNs [↑] that merge Inter- and Intranet [↑] services.

2.3 Modeling Network Design Problems

2.3.1 Mathematical Concepts

A real-world problem has to be represented in an appropriate form before any mathematical techniques can be used. This process is called modeling.

Definition of Concept 2 (Mathematical Model, Level of Detail)
*A **mathematical model**[7], sometimes also called **mathematical program** [↑MP], $(\mathcal{P}, \mathcal{Z}, \mathcal{V}, o)$ is a set of mathematical relationships (e.g. equalities, inequalities, logical conditions) which represents an abstraction of the real world problem under consideration. A mathematical model consists of the following objects (the terms in parentheses list the terminology used later in the thesis):*

- *input data or input parameters [$\mathcal{P}$]*
- *variables (continuous, semi-continuous, binary, integer) [$\mathcal{V}$], defining the solution space $S_{\mathcal{V}}$ (see Definition 3)*
- *constraints (equalities, inequalities) [$\mathcal{Z}$]*
- *objective function (or simply objective) [$o : S_{\mathcal{V}} \rightarrow \mathbb{R}$]*

*A most important aspect is the **level of detail** in which the problem is modeled. A manageable level of detail is often necessary for computational tractability while the characteristics of the real-world problem should be maintained.*

A model is therefore an appropriate abstract representation of a real system at the chosen level of detail. The values of parameters fix an instance of a problem. They may represent costs, demands, and so on. The variables represent the degrees of freedom. The constraints represent quality relations, equipment capabilities, etc. which the solution has to fulfill. Finally, the objective function expresses the goal such as to minimize costs.

Mathematical models can be distinguished by the following characteristics:

- their level of detail
- their comprehensiveness, i.e. the ability to express real-world problems. In the context of a given problem, comprehensiveness is sometimes also denoted as *credibility*.
- their comprehensibility, i.e. the easiness with which the model is understood by non-experts.

[7] adapted from [KW97]

- their extensibility, composibility and adaptability.
- their computational tractability, i.e. the time and space complexity which is usually denoted by the Landau-symbol $O(\cdot)$ ([↑Landau-Symbol]).

Definition of Concept 3 (Solution, Feasibility, Solution Space)
*A **solution** of a mathematical model is an assignment of values to each of its variables $\mathcal{V}$ honoring their domains. A solution is said to be **feasible** if it fulfills all constraints in $\mathcal{Z}$. The set of all feasible solutions, i.e. the **solution space**, is denoted as $S_{\mathcal{V}}$.*

*A solution x^* is said to be **optimal** subject to an objective function, o, if it is feasible and there exists no other feasible solution $\llcorner x \in S_{\mathcal{V}} \lrcorner$ that has a "better" (smaller or higher) objective function value $o(x)$.*

To deal with the complexity of the design, beside the level of detail, two fundamental concepts are used: *abstraction* and *structuring*. After classifying the problem and building a model, a *solver*, i.e. a piece of software which implements an algorithm capable of finding a "good" feasible solution, is needed.

Definition of Concept 4 (Solver)
*A **solver** is a proven method which finds a "sufficiently good" solution to a specific class of problems which are represented by a certain model.*

To capture different aspects of different solvers, sometimes more than one model is used. Therefore the models have to be designed compatibly. Such models are called *composite models*. The linkage between models can be sequential, parallel, feedback or iterative, embedded or multi-level. Another approach is the *decomposition* of a model, which refers to the breaking down of a large complicated problem into many smaller "solvable" ones, thereby reducing complexity while (generally) sacrificing optimality.

The selection of a specific solver is mainly driven by the required functionality which usually depends on the complexity, input requirements, applicability, or other appraisal criteria.

2.3.2 Concepts used in Network Design

It is a non-trivial task to formulate a network design problem. Due to the fact that a network must satisfy the needs of an enterprise, whereas every enterprise has different requirements on communication, network design is a context sensitive problem.

To illustrate the (sub-)problems arising in network design, the planning task of the Wissenschaftsnetz (*WiN*) provided by Deutsches Forschungsnetz Verein (*DFN-Verein*) is regarded as an example throughout the thesis. This network is built to satisfy the demands of the German research community. On the one hand this design problem is not too specific, on the other hand full information concerning the planning task is available.

Generally, networks are designed using *hierarchical* structures (sometimes denoted as *stratification*). Two different types of hierarchies can be identified: The *topological* or *vertical* hierarchy imposed by the different network layers (see Figure 1) and the *logical* or *horizontal* hierarchy implied by tiers (see Definition 5).

Accepting the following statements:

1. Network design as a whole, i.e. *overall network design*, is much too complicated to be solved in one step.

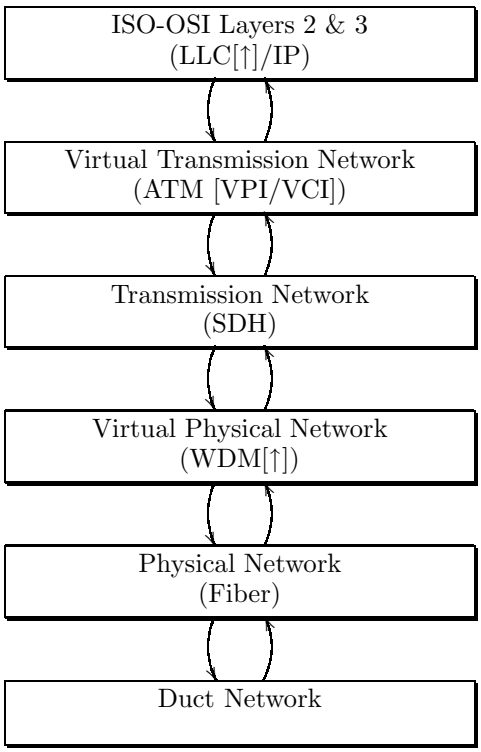

Figure 1: Example of a Topological Hierarchy

2. An obvious "optimum" solution[8] does not exist in general due to multi-criteria objective functions and incomplete knowledge.
3. Network design is an iterative, user controlled procedure.
4. Networks exist within enterprises and must be adjusted to the goals of the enterprise.

leads to *hierarchical decomposition planning* (see for example Figure 2 and [FB98]) with subtasks being well defined and *independent* to the largest possible degree combined with *alternate optimization*, i.e. optimization with recursion, and interaction with the designer.

The hierarchical decomposition planning process must be embedded in the topological hierarchy of the network, for example executed for each topological layer. The more hierarchy levels a designer wants to take into account simultaneously, the more complex the planning problem, respectively its mathematical model, becomes. Only a joint optimization over all layers and tiers (see Definition 5) safeguards an optimal solution, but usually results in an intractable complexity. Optimality is therefore sacrificed for tractability.

A *hierarchical network design* having two tiers is proposed throughout this thesis[9].

Definition of Concept 5 (Two Tier Hierarchical Network Design)
The two tier hierarchical design splits up the network into two layers, denoted as ***tiers****. The first* ***access-tier*** *contains the* ***access-*** *or* ***end-nodes****. Each end-node represents a canonical source that sends traffic into the network. The traffic originating from the end-nodes may be the output of a single source like a personal computer, or a already multiplexed traffic stream (e.g. of a department of a corporation). The second* ***backbone-tier*** *contains*

[8] optimum is here for once not used as mathematical term

[9] It should be mentioned that a hierarchical design is no restriction of generality. Putting all nodes in one tier reduces the hierarchical problem to a non-hierarchical.

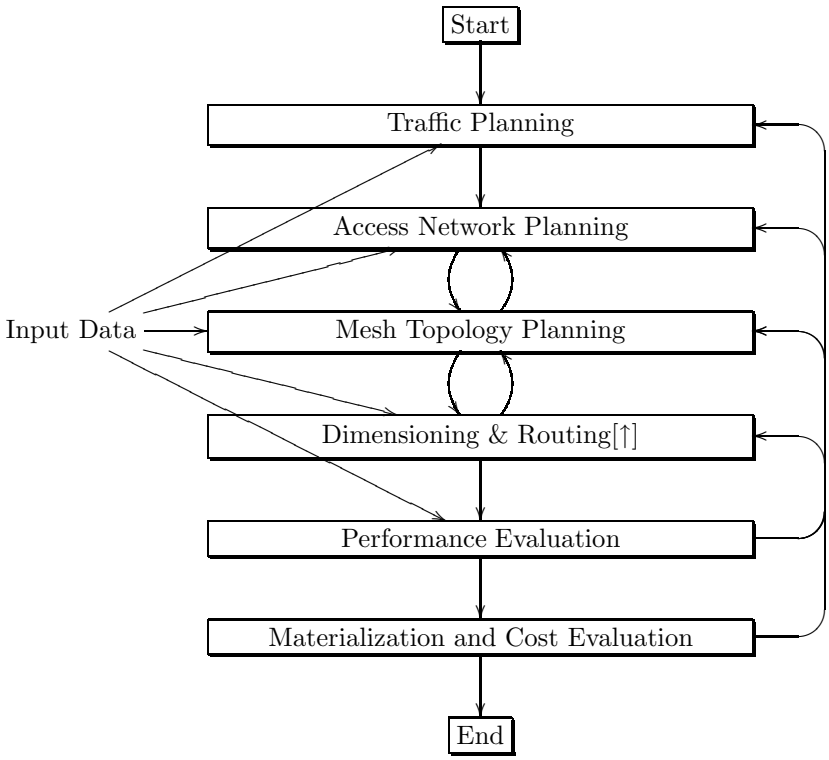

Figure 2: Topological Hierarchical Decomposition Planning (adapted from [BPT98])

*the **backbone-nodes**. Each end-node sends to and receives its traffic from exactly one backbone-node*[10]*. End-nodes and backbone-nodes are allowed to be coincident, i.e. need not be exclusive.*

Definition of Concept 6 (Topology, Configuration, Facilities & Network Design)
*A **facility** is an arbitrary network element such as a router or an arbitrary service such as a leased line. The set of all available facilities is denoted as F.*

*A **topology** $\mathcal{T}$ in the context of network design is the physical or logical layout of network facilities. Common topologies include star, ring, bus, or mesh. A common representation of the network topology is a **graph** $\llcorner G = (V, E) \lrcorner$ with the set of nodes V and the set of edges $\llcorner E \subseteq V \times V \lrcorner$ (see for example [Jun87] and [CH94] for a detailed introduction to concepts and terminology).*

*A **configuration** $\mathcal{C}$ in the context of network design is the selection and initialization of all facilities $\llcorner f \in F \lrcorner$ used in the network (e.g. the setting up of routing [↑] tables). A configuration of a certain facility f is denoted as $\mathcal{C}_f$.*

*A **network design**[11] $\mathcal{D}$ is the specification of the topology and configuration forming a complete network, i.e. $\llcorner \mathcal{D} = (\mathcal{T}, \mathcal{C}) \lrcorner$. To put it in other words, the topology specifies where to put the facilities and how to interconnect them, the configuration specifies which facility is used at each topological element and how to set it up. The set of all used facilities in a certain design is denoted as $F_{\mathcal{D}}$.*

[10] Note that this is also no restriction in generality due to the fact that end-nodes which require a higher connectivity can be thought as backbone-nodes.

[11] In contrast to network design task, here a specific network design, i.e. a realization of a single network, is meant. From the context it should be immediately clear to which of the two meanings it is referred.

Remark 1
The number of all possible graphs to connect n nodes is of the order $O(2^{n(n-1)/2})$.

2.4 Static Network Design

Definition of Concept 7 (Static Network Design (*SND*))
*Network design is said to be **static** if variations in network traffic over time are not taken into account.*

Definition of Concept 8 (Static Network Design Problem (*SNDP*))
*A **static network design problem** is given by:*

- *a set V_e of end-nodes representing sources and destinations of communication requirements. An end-node is a traffic concentration point from/to which traffic is transported to/from various other end-nodes in V_e.*
- *a set V_b of possible backbone-nodes that may be a subset of V_e or any set of points in a (continuous) area.*
- *a traffic matrix R of the dimension $|V_e|^2$ representing (peak) communication **requirements**[12] $r_{i,j}$ (r_t-dimensional for r_t different traffic classes) between the end-node $\llcorner i \in V_e \lrcorner$ and $\llcorner j \in V_e \lrcorner$ (usually of **statistical** nature). R is not necessarily symmetric. If the requirements are not known in advance, estimates made by experts may have to substitute this incomplete knowledge.*
- *a network technology together with a set of available facilities F*
- *a routing [↑] scheme*
- *a cost structure $\llcorner C = (C^r, C^{nr}) \lrcorner$ assigning to every available facility $\llcorner f \in F \lrcorner$ two cost functions[13]:*

$$\begin{aligned} C^r &: F \times \mathcal{B}_0^+ \to \mathbb{R}_0^+ \\ C^{nr} &: F \to \mathbb{R}_0^+ \end{aligned}$$

The first cost function $\llcorner C^r(f,b) =: C_f^r(b) \lrcorner$ is a continuous or discrete function depending on the facility $\llcorner f \in F \lrcorner$ and the time set $\llcorner b \in \mathcal{B}_0^+ \lrcorner$. It expresses the discounted cash flow (DCF) (see Appendix C) of the reoccurring (e.g. in regular periods) expenses for a deployed facility. The second cost function $\llcorner C^{nr}(f) =: C_f^{nr} \lrcorner$ depends on $\llcorner f \in F \lrcorner$ and expresses the non-reoccurring costs, such as installation costs[14]. If there exists more than one provider or vendor for a facility, the cost structure may even be multi-dimensional.

Optimize the objective function value $\llcorner o : S_{\mathcal{D}} \to \mathbb{R}_0^+ \lrcorner$ over the design $\mathcal{D}$ subject to

- *hardware technology constraints, e.g. implied by the throughput of facilities*
- *software technology constraints, e.g. implied by the routing scheme*
- *QoS constraints e.g. on blocking, delay or packet loss*
- *reliability/survivability [↑], e.g. 2-connectivity or diversification [↑]*
- *robustness, for example the ability to carry a slightly different traffic pattern*

[12] sometimes also called **commodities**

[13] $\mathcal{B}_0^+$ denotes all measurable subsets of $\mathbb{R}_0^+$ and is a subset of the Borel set $\mathcal{B}$.

[14] In some situations it is possible to trade setup and reoccurring costs against each other, for example the acquisition can be distributed over the life time of the equipment and therefore represented as reoccurring costs, see again Appendix C for further details.

- *constraints implied by the management (OAM [↑])*
- *other constraints (e.g. already owned lines, fixed backbone-nodes, security etc.)*

Remark 2

Representing an SNDP *by a mathematical model* $(\mathcal{P}, \mathcal{Z}, \mathcal{V}, o)$ *results in the following assignments:*

- $\mathcal{P} = (V_e, V_b, R, F, C)$
- $\mathcal{Z}$ *"is given by a functional mathematical representation of all constraints imposed on the solution (technology and user constraints), e.g. by relations between the parameters"*
- $\mathcal{V} = \mathcal{D}$
- o *"is a function* $\llcorner S_{\mathcal{V}} \to \mathbb{R}_0^+ \lrcorner$ *and usually depends on* $\mathcal{P}$*"*

It can easily be seen that the representation of constraints is by far the most difficult part of the modeling task and should be performed[15] *by an experienced designer.*

It has been shown (e.g. in [MS81]) that even subproblems of the *SNDP* are strongly $\mathcal{NP}$-complete[16]. Another important fact is that there is no generic method of taking into account the various constraints imposed on the problem.

A special case of static network design problems are multi-hour network designs, i.e. designs that make use of different static traffic matrices for each hour a day, respectively, each day a week. A multi-hour design can yield considerable savings in comparison to a single-hour design (as shown in e.g. [Dut94, Bau97a]), but depends on precise knowledge of user behavior. This situation occurs mainly in voice telecommunication networks and therefore will be of no further concern in this thesis (see Section 2.7.1.5).

Definition of Concept 9 (Observation Period)

The ***observation period****, sometimes also denoted as* ***observation interval****, is an ordered set* T*, where* T *can be a discrete set of dates or a continuous closed time-interval fixed by its begin and end time. An element* $\llcorner t \in T \lrcorner$ *is called an* ***observation time****. If not mentioned explicitly, throughout this thesis the observation period is assumed to be a discrete, finite set of monthly, weekly or daily units*[17] *over a certain period. For notational convenience let* $\llcorner t_0 = \min T := 0 \lrcorner$ *and* $\llcorner t_{\max} = \max T \lrcorner$.

Remark 3

The total discounted[18] *costs* $C(\mathcal{D}, T)$ *of a* static *design* $\mathcal{D}$ *over the complete observation period* $\llcorner T = [t_0, t_{\max}] \lrcorner$ *are:*

$$C(\mathcal{D}, T) := \sum_{f \in F_{\mathcal{D}}} \left[C_f^{nr} + C_f^r(\llcorner [t_0, t_{\max}) \lrcorner) \right] \tag{1}$$

[15] modeling is sometimes regarded as an art

[16] which means that under the assumption $\mathcal{P} \neq \mathcal{NP}$ this problem is not solvable in an efficient way and even worse there exists no efficient pseudo-polynomial approximation algorithm (see e.g. [BDG88, Hoc95]).

[17] This stands reason, because in the context of corporate network design the contracts concerning renting or leasing facilities are generally in the unit of months or weeks.

[18] see Appendix C

2.5 Incremental Network Design

Corporations grow, employees move and network requirements will increase almost certainly each time that new applications and services become available. So it is likely that even a perfectly designed network reaches its limits after a surprisingly short time period.

To quote it with more illustrative words [Cah98]:

> "It's like the shoes of a child. At the moment they fit perfectly, but what about in 3 weeks ?"

This leads to the necessity of upgrading a network design in more or less regular intervals.

Definition of Concept 10 (Incremental Network Design (*IND*))
Incremental Network Design *deals with the problem of upgrading or reconfiguring a network and the timing decision involved. It has to find answers to the following questions:*

- *when is the right moment to take action (i.e. the* ***trigger events****)*
- *what sort of actions to take:*

  reconfigure *, i.e. change the configuration $\mathcal{C}$ of some facilities $F_{\mathcal{D}}$.*

  upgrade *, i.e. add/remove facilities or replace already deployed facilities while preserving the existing topology $\mathcal{T}$ as far as possible (sometimes also called* ***augmentation****).*

  redesign *, i.e. solve the* SNDP *that is given by the altered environment without (explicitly) taking the deployed design $\mathcal{D}$ into account.*

IND is a time driven process. The most important point is to figure out trigger events that initiate a new *IND* consideration. Examples of trigger events are:

- regular time periods
- a contract with a provider ends
- specified QoS violations (e.g. measured throughput or user complaints)
- addition or removal of end-nodes
- introduction of new services
- disruption of services (e.g. vendor going out of business)

Of course, any combination of the situations above can serve as trigger event. After a trigger has been pulled, usually, one of two different situations has occurred:

Situation 1: There are bottlenecks on some network facilities which throttle performance, whereas other facilities in the network are under-utilized. This situation can occur when traffic shifts, e.g. because of a new service provided at a end-node. A specific augmentation of single affected facilities having "negligible" costs, or reconfiguration by moving traffic to less utilized facilities may solve this problem.

Situation 2: The overall traffic growth exceeds the throughput [↑] of the network as a whole. Then a larger-scale upgrade or redesign of the network will be necessary. No simple reconfiguration can relieve the insufficient performance of the network. The task of the designer is to decide whether an augmentation of network facilities is profitable or a major change is necessary.

Situation 1 usually passes over to situation 2 after some time. A common (but unsatisfactory) strategy is to "muddle through" until a complete redesign is necessary, i.e. wait until situation 2 comes true and in the meantime making only reconfigurations that are free of charge. Reconfiguration measures belong to the task of *NOP*.

Definition of Concept 11 (Incremental Network Design Problem (*INDP*))

*An **Incremental Network Design Problem** is entailed with the upgrade of an existing network design $\mathcal{D}$ to meet altered requirements and is given by*

- *a set of available facilities F'*
- *a set V'_e of end-nodes*
- *a set V'_b of backbone nodes*
- *a traffic matrix R' with the dimension $|V'_e|^2$*
- *a given (already deployed) network design $\llcorner\mathcal{D} = (\mathcal{T}, \mathcal{C})\lrcorner$ together with the set of facilities $F_\mathcal{D}$*
- *a cost structure $\llcorner C' = (C^r_f, C^{nr}_f, C^\nabla_{f_1,f_2})\lrcorner$, where*

$$C^\nabla_{f_1,f_2} : F_\mathcal{D} \times F' \to \mathbb{R}^+_0$$

expresses the costs for upgrading or downgrading the facility $\llcorner f_1 \in F\lrcorner$ to the facility $\llcorner f_2 \in F'\lrcorner$. If this upgrade is not possible or does not make sense, a prohibitive, fictitious value like $+\infty$ can be assigned. For notational convenience, the following simplifications are made[19]:

$$\begin{aligned} C^\nabla_{\oslash,f_2} &:= C^{nr}_{f_2} && f_2 \in F' \\ C^\nabla_{f_1,\oslash} &:= \textit{termination costs of } f_1 && f_1 \in F_\mathcal{D} \\ C^\nabla_{f,f} &:= 0 && f \in F_\mathcal{D} \wedge f \in F' \end{aligned} \tag{2}$$

- *a set of constraints as described in Definition 8*

*The objective is to find a feasible network configuration $\llcorner\mathcal{D}' := (\mathcal{T}', \mathcal{C}')\lrcorner$ meeting the constraints such that the **upgrade costs** of design $\mathcal{D}$ are minimal.*

Remark 4

Note that the INDP *is deliberately formulated to regard only costs as the single objective criterion but the designer is free to use other objectives, too.*

Remark 5

It can be shown easily that under real-world conditions, i.e. credible tariffs [↑] (see Definition 23) and strict traffic growth (see Section 2.7.5 and Appendix C.4), the upgrade of a design is cheapest in terms of costs when it is deployed at the last moment possible.

Definition of Concept 12 (Set of Changes ∇)

*The **set of changes** ∇ is the set of tuples $\llcorner(f'_1, f'_2) \in \{F_\mathcal{D} \cup \{\oslash\}\} \times \{F'_{\mathcal{D}'} \cup \{\oslash\}\}\lrcorner$ that describe the change of each network element in an incremental design. ∇ consists of the following elements:*

i) for each facility $\llcorner f_1 \in F_\mathcal{D}\lrcorner$ there exists exactly one element $\llcorner(f'_1, f'_2) \in \nabla\lrcorner$ with $\llcorner f'_1 = f_1\lrcorner$. f'_2 is either the successor of f_1 or $\oslash$ if no successor exists.

[19] The symbol $\oslash$ represents the state where no facility is present.

ii) for each facility $\llcorner f_2 \in F_{\mathcal{D}'} \lrcorner$ *there exists exactly one element* $\llcorner (f_1', f_2') \in \nabla \lrcorner$ *with* $\llcorner f_2' = f_2 \lrcorner$. f_1' *is either the predecessor of* f_1 *or* $\oslash$ *if no predecessor exists.*

Remark 6

The total discounted costs $C(\mathcal{D}', T')$ *of a design* $\mathcal{D}'$ *over the new observation period* $T' = [t_0', t_{\max}']$ *(which is usually the remaining observation period from* T*) are:*

$$C(\mathcal{D}', T') \;:=\; \sum_{f \in F_{\mathcal{D}'}} C_f^r(\llcorner [t_0', t_{\max}') \lrcorner) + \sum_{(f_1', f_2') \in \nabla} C_{f_1', f_2'}^{\nabla} \tag{3}$$

IND is a *short-sighted* (*my-optic*) or *reactive* network design strategy. The designer reacts whenever one of the trigger events occurs. Then the designer has to choose between reconfiguration or redesign, whichever is economically justified.

Definition of Concept 13 (Slack Costs)

The ***slack costs*** *of a design* $\mathcal{D}'$ *is the difference of the reoccurring costs* C^r *between an* INDP*-solution and the corresponding* SNDP*-solution of a complete redesign.*

The more the requirement matrix R' differs from the initial requirement matrix R for which $\mathcal{D}$ was designed, the more economically attractive a redesign of the whole network becomes. The *slack costs* of an *IND* are an indicator of how "far off" the "upgraded" solution compared to the best solution of a complete redesign is. One indicator for evaluating *IND*s is the **pay-back period**, i.e. the time span $t_{\text{pay-back}}$ for which a redesign $\mathcal{D}_{\text{new}}$ yields savings compared to upgrading.

Definition of Concept 14 (Pay-back Period)

$$t_{pay\text{-}back} = \min \{ t \in T' \,:\, C(\mathcal{D}', \llcorner [t_0', t) \lrcorner) \geq C(\mathcal{D}_{new}, \llcorner [t_0', t) \lrcorner) \} \tag{4}$$

2.6 Evolutionary Network Design

2.6.1 Motivation

In contrast to *IND*, Evolutionary Network Design (***END***) is a *far-sighted* strategy. It is motivated by the observation that every reconfiguration involves fixed upgrade-costs (i.e. the considerable amount of money the providers charge for making changes in design) that could be avoided by taking *foreseeable* future developments into account.

To illustrate this statement, the following simplified setting is considered:

- one requirement between two end-nodes
- an observation period T, $\llcorner \Delta_1, \Delta_2 \in T \lrcorner$, $\llcorner \Delta_1 < \Delta_2 \lrcorner$
- the expected traffic between end-nodes is described by the dashed line in Figure 3 and the assumption that the traffic develops as expected
- line-facilities $\llcorner F = \{f_1, f_2, f_3\} \lrcorner$ with capacities $\llcorner C_1, C_2, C_3 \lrcorner$, respectively, cost functions $\llcorner C_{f_i, f_j}^{\nabla}, \llcorner f_i, f_j \in F \cup \oslash \lrcorner \lrcorner$ corresponding to Definition 11 and $\llcorner C_{f_i}^r, i \in 1, 2, 3 \lrcorner$
- trigger events at time $\llcorner \Delta_1, \Delta_2 \in T \lrcorner$ that are pulled by the over-utilization of the line capacity C_1 and C_2, respectively
- already deployed facility f_1

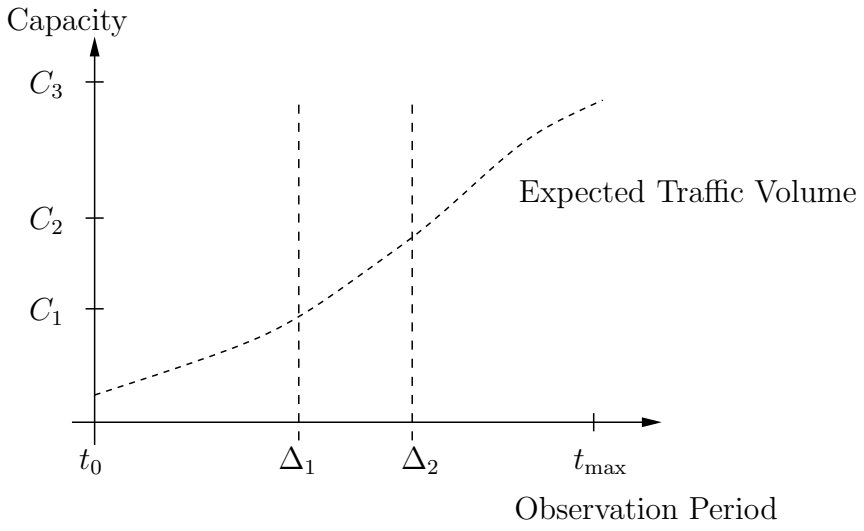

Figure 3: One-Line Traffic Growth

IND as defined in the previous section would "wait until the trigger event at time Δ_1 and then upgrade f_1 to f_2", the same way it would upgrade f_2 to f_3 at time Δ_2. Here the *IND*-solution is denoted as E_1 and has the discounted costs[20]:

$$\begin{aligned} C_{E_1} &= \mathrm{PV}(C^{\nabla}_{f_1,f_2}, \Delta_1) + \mathrm{PV}(C^{\nabla}_{f_2,f_3}, \Delta_2) + \\ &\quad C^r_{f_1}(\llcorner[t_0, \Delta_1)\lrcorner) + C^r_{f_2}(\llcorner[\Delta_1, \Delta_2)\lrcorner) + C^r_{f_3}(\llcorner[\Delta_2, t_{\max})\lrcorner) \end{aligned} \tag{5}$$

Definition of Concept 15 (Sequence)
*A **sequence** denotes a chronological succession of events, decisions, etc.*

Remark 7
A sequence of successive (i.e. at the trigger events) IND*-solutions always form a feasible solution to the corresponding* END*-problem.*

Continuing the example, in contrast to *IND*, *END* could also consider to deploy f_3 at time Δ_1 covering the requirements of the period $[\Delta_1, t_{\max})$ by just one upgrade. This solution is denoted as E_2 and has the discounted costs:

$$C_{E_2} = \mathrm{PV}(C^{\nabla}_{f_1,f_3}, \Delta_1) + C^r_{f_1}(\llcorner[t_0, \Delta_1)\lrcorner) + C^r_{f_3}(\llcorner[\Delta_1, t_{\max})\lrcorner) \tag{6}$$

A network designer taking discounted costs as the single objective, should vote for solution E_2 if

$$C_{E_1} > C_{E_2} \tag{7}$$

In the case of a time linear cost function C^r_f the decision criterion becomes:

$$\mathrm{PV}(C^{\nabla}_{f_1,f_2}, \Delta_1) + \mathrm{PV}(C^{\nabla}_{f_2,f_3}, \Delta_2) + C^r_{f_2}(\llcorner[\Delta_1, \Delta_2)\lrcorner) > \mathrm{PV}(C^{\nabla}_{f_1,f_3}, \Delta_1) + C^r_{f_3}(\llcorner[\Delta_1, \Delta_2)\lrcorner) \tag{8}$$

[20]Information of the function PV can be found in Definition 58 (Appendix C).

2.6.2 Introduction of Concepts

2.6.2.1 Planning Horizon Evolutionary Network Design is a **multi-period** or **multi-staged** design problem, i.e. it considers more than one network upgrade at a time. Therefore it makes use of *dynamic models*, i.e. models that take time explicitly into account. Dynamic models can be classified as follows [ITU83]:

Defined Horizon Models: The design of the network at the end of the observation period has been defined a priori by the designer. The objective of the optimization process is to determine the optimum *evolution*, i.e. a set of network designs for each observation time of the observation period, realizing the predetermined design at the end of the observation period and which is consistent with the network conditions at the start of the observation period. The observation period T in defined horizon models is covered by a closed interval $\llcorner T \subseteq [0, t_{\text{end}}] \lrcorner$ and is either a continuous interval or a finite set of points with $\llcorner 0, t_{\text{end}} \in T \lrcorner$.

Undefined Horizon Models: In contrast to defined horizon models, in undefined horizon models the solution at the end of the observation period is not defined a priori but determined as a result of the dynamic study. In this sense, a sequence of *IND*-problems corresponds with an undefined horizon model. For the observation period T in undefined horizon models holds: $\llcorner T \subseteq [0, +\infty) \lrcorner$.

From a conceptual viewpoint, the undefined horizon model better represents the real network planning problem, however, it is more difficult to solve. In this thesis we will focus on defined horizon models. However, a predetermined design at the end of the observation time is **not** assumed, merely that all requirements and constraints for each $\llcorner t \in T \lrcorner$ are met.

2.6.2.2 Forecasting The quality of an Evolutionary Network Design depends critically on the quality of the prediction, i.e. *forecasts*, of the future traffic requirements. The more *uncertain* the future is, the more imprecise the forecasted traffic requirements will be, the shorter the observation period of a defined horizon model has to be to yield useful results. In voice telecommunication e.g. where the expected growth rates are small, forecasts can cover more than a decade.

Forecasts for Corporate Networks are subject to the following influence factors:

- the traffic demand per user and application
- the number of users or services at each end-node
- the number and location of end-nodes
- the external traffic at the peering points [↑] of the network
- the requirements of enterprise wide applications (ERP [↑])

If the growth of the network traffic is considerable, a short study period may be appropriate because significant errors in forecasts become more likely. A more detailed discussion on forecasting techniques will be conducted in Section 4.4.2.3.

All following considerations are based on the assumption "*traffic develops as forecasted*", but in most real-world situations forecasts are entailed with significant errors. This is especially true for networks carrying IP traffic that were showing exponential traffic growth (with varying exponents!) for the last few years. A credible design should take this knowledge into account. Network design under uncertainty is the main topic of Chapter 4.

2.6.2.3 Network Sensitivity Testing A first assessment of how a designed network can cope with changing traffic patterns is to determine how much additional traffic can be carried before any of the performance related trigger events are set off. This process is called *network sensitivity testing*.

Given a design $\mathcal{D}$, the following strategy for performing sensitivity testing is proposed:

Definition of Concept 16 (Network Sensitivity Testing)
Network sensitivity testing consists of two steps:

- *select a sequence of* $\llcorner s \geq 2, s \in \mathbb{N} \lrcorner$ *points* $\llcorner S = \{t_0^s, t_1^s, \ldots, t_{s-1}^s\} \lrcorner$ *from the observation period* T *with* $\llcorner t_0^s < t_1^s < \ldots < t_{s-2}^s < t_{s-1}^s \lrcorner$
- *forecast the traffic matrix* $R(t)$ *for each* $\llcorner t \in S \lrcorner$ *and determine the first* $\llcorner \Delta_1 \in S \lrcorner$ *that pulls a performance related trigger of the design* $\mathcal{D}$ *(which can be decided e.g. by* ***simulation*** *or by the use of an appropriate* ***network loader****).*

A design $\mathcal{D}_1$ *that pulls the trigger earlier than a design* $\mathcal{D}_2$ *is denoted as more sensitive.*

Definition of Concept 17 (Routing Scheme, Network Loader, Overload)
A ***routing scheme*** *[↑Routing] is a set of rules that determine the paths taken by a data flow from the source to the destination usually based on some optimality metric.*

Given a network design $\mathcal{D}$ *and a traffic matrix* R*, a* ***network loader*** *[Cah98] determines how much of the traffic flows over each facility* $\llcorner f \in F_{\mathcal{D}} \lrcorner$ *(respectively topological element) according to a (static) routing scheme.*

If the amount of traffic that flows over a facility according to a network loader exceeds the capacity of that facility the difference between the traffic flow and the capacity is denoted as ***overload****.*

Definition of Concept 18 (Slack Capacity, Over-Provisioning)
The ***slack capacity*** *of a network is a weighted sum of spare capacities over all network facilities* $\llcorner f \in F_{\mathcal{D}} \lrcorner$ *that the network design does not need according to a network loader. Slack capacities usually arise due to the fact that capacities are available only in discrete quantities, i.e. blend, or due to the over-provisioning of facilities.*

Over-provisioning *denotes the selection of facilities having a much higher capacity than it would be necessary to transport the traffic which is loaded on it.*

2.6.2.4 Problem Formulation

Definition of Concept 19 (Evolutionary Network Design Problem (ENDP))
An ***evolutionary network design problem (ENDP)****, also called* ***multi-period*** *or* ***multi-staged*** *network design problem, is given by the following setting:*

- *an observation period* T
- *a (static) network design problem for each* $\llcorner t \in T \lrcorner$ *with the parameters: end-nodes* $V_e(t)$*, potential backbone nodes* $V_b(t)$*, requirements* $R(t)$*, available facilities* $F(t)$*, and a set of constraints* $\mathcal{Z}(t)$ *corresponding to Definition 8.*
- *a cost structure* $\llcorner C^t = (C_f^{r,t}, C_f^{nr,t}, C_{f_1,f_2}^{\nabla,t}) \lrcorner$ *for each* $\llcorner t \in T \lrcorner$*, corresponding to Definition 8 and Definition 11. Sometimes it makes sense to have a cost structure that depends on the time a facility is deployed, since the costs for certain facilities or services are likely to change during the observation period.*

Find

- *an initial network design* $\llcorner\mathcal{D}(\Delta_0), \Delta_0 := t_0\lrcorner$
- *a set of* $d \in \mathbb{N}$ *upgrade times* $\llcorner\{\Delta_1, \dots \Delta_d\} \subseteq T\lrcorner$ *together with network designs* $\llcorner\mathcal{D}(\Delta_i) = (\mathcal{T}(\Delta_i), \mathcal{C}(\Delta_i)),\ i \in \{0, 1, \dots, d\}\lrcorner$ *such that if*

$$\mathcal{D}(t) := \begin{cases} \mathcal{D}(\Delta_{i-1}) & \text{if } \Delta_{i-1} \leq t < \Delta_i \\ \mathcal{D}(\Delta_d) & \text{if } \Delta_d \leq t \end{cases} \qquad t \in T \tag{9}$$

$\llcorner\{\mathcal{D}(t) : t \in T\} =: \mathcal{D}^T\lrcorner$ *is a sequence of feasible solutions to each* SNDP *at time* t *for all* $\llcorner t \in T\lrcorner$.

subject to an objective function o *and over the design variables* $\mathcal{V}$ *given by*

$$\llcorner\mathcal{D}(t) = (\mathcal{T}(t), \mathcal{C}(t)),\ \forall t \in T\lrcorner\,.$$

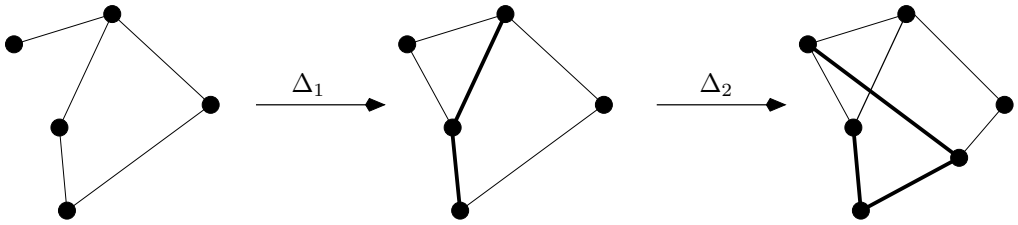

Figure 4: Example of a Network Evolution

Definition of Concept 20 (Solution, Feasibility of *ENDP*s)
A ***solution*** $\mathcal{D}^T$ *of an* ENDP *(also denoted as* ***evolution****) is said to be feasible, if* $\mathcal{D}(t)$ *is a feasible solution to the corresponding* SNDP *of time* t *for all* $\llcorner t \in T\lrcorner$.

An illustration of a network evolution with two upgrade times $\llcorner\Delta_1, \Delta_2\lrcorner$ is shown in Figure 4.

Remark 8
Please note that the ENDP *is formulated assuming complete knowledge, i.e. all future settings such as* $R(t)$ *or* $C(t)$ *are known in advance. A problem having full knowledge of a system during the observation period* T *will be denoted as* ***problem under complete knowledge****, whereas a problem that has to deal with factors afflicted with uncertainties will be denoted as a* ***problem under incomplete knowledge****.*

Remark 9
Sometimes the initial network design $\mathcal{D}(t_0)$ *is given. If his is the case we will denote the problem as an* ENDP *with* ***given starting conditions****.*

Remark 10
Remark 5 applies to ENDP *in a similar fashion. It can be easily shown that under real-world conditions upgrades are deployed at the last possible moment.*

Definition of Concept 21 (Set of Changes ∇_i)
The set ∇_i $(\llcorner i \in \{1 \dots d\}\lrcorner)$ *contains all tuples*

$$(f_1, f_2) \in \{F_{\mathcal{D}(\Delta_{i-1})} \cup \{\oslash\}\} \times \{F_{\mathcal{D}(\Delta_i)} \cup \{\oslash\}\}$$

that describe the change of each network element at time Δ_i in an evolutionary design. ∇_i honors the properties of Definition 12.

Remark 11
The total discounted costs $C(\mathcal{D}^T)$ of a design $\mathcal{D}^T$ over the observation time T and $\llcorner\Delta_d := t_{\max}\lrcorner$ are (see Appendix C.3):

$$C(\mathcal{D}^T) := \sum_{f\in F_{\mathcal{D}_0}} C_f^{nr} + \sum_{i=1}^{d}\left[\sum_{(f_1,f_2)\in\nabla_i} PV(C_{f_1,f_2}^{\nabla,\Delta_i},\Delta_i) + PV\Big(\sum_{f\in F_{\mathcal{D}(\Delta_{i-1})}} C_f^r(\llcorner[\Delta_{i-1},\Delta_i)_\lrcorner),\Delta_{i-1}\Big)\right] \quad (10)$$

$$\text{where } \nabla_0 := \begin{cases} \bigcup_{f\in F_{\mathcal{D}_0}}(f,f) & \text{in an ENDP with given starting conditions} \\ \bigcup_{f\in F_{\mathcal{D}_0}}(\oslash,f) & \text{otherwise} \end{cases}$$

Remark 12
Analogous to Remark 2, an ENDP *which is represented as mathematical model $(\mathcal{P},\mathcal{Z},\mathcal{V},o)$ leads to the following assignments:*

- $\mathcal{P}^T := \mathcal{P}(t) = (V_e(t), V_b(t), R(t), F(t), C(t))$
- $\mathcal{Z}^T := \mathcal{Z}(t)$ *"is a functional mathematical representation of all constraints imposed on the solution (technology and user constraints) at each time $t \in T$"*
- $\mathcal{V}^T := \mathcal{V}(t) = \mathcal{D}^T$
- $o(D^T) := C(\mathcal{D}^T)$ *"if discounted costs are the only objective criterion, otherwise a function $\llcorner S_{\mathcal{V}^T} \to \mathbb{R}^+\lrcorner$"*

2.6.2.5 Summary Depending on the cost functions, the requirements, and the observation period, *ENDP*s are very difficult to solve even if all future developments are known in advance, i.e. with complete knowledge. An optimal solution $\mathcal{D}^T$ of an *ENDP* consists of a sequence of *SNDP* solutions $\mathcal{D}(t)$ at each time $\llcorner t \in T\lrcorner$ but none of these $\mathcal{D}(t)$ may be necessarily an optimal *SNDP* design. Depending on the level of detail and on the objective function, changes between successive network designs $\llcorner\mathcal{D}(\Delta_{i-1})$ and $\mathcal{D}(\Delta_i)$, $i \in \{1,\ldots,d\}\lrcorner$ may consist of small changes in configuration that are subject to *NOP*, minor augmentations of some facilities or even a major change.

2.7 Dimensions & Degrees of Freedom

To achieve a better insight into network design problems with their overwhelming number of variations and variables in general, a first step consists in having a look at the factors that may influence the problem and its modeling. The following sections will give a brief overview of the "*dimensions of network design*".

2.7.1 Network Traffic

2.7.1.1 Overview Network traffic modeling is the core of network design. Without having a good (i.e. accurate) idea of how the traffic will look like, e.g. by measurements or traffic models, a reasonable network design is extremely difficult.

The traffic requirements between the end-nodes (denoted with the symbol R) are a measure of the communication demand. In the context of voice communication, R is usually measured in terms of Erlangs [↑] in the *busy hour* [↑]. The focus in this thesis, however, is on data communication, especially IP data traffic. In most networks today, IP data traffic includes also the multimedia traffic (encapsulated by IP packets). The IP traffic requirements are usually given in terms of transported volume (in Megabytes) or peak bandwidth (in Mbps [↑]) during a certain time period. In a native ATM environment, the requirements R may also specify the number of connections of each traffic type during the busy hour. The deployed communication network must be able to transport the requirements R meeting the desired QoS and other constraints. Given a fixed topology and a routing scheme, the problem is transformed into the appropriate *dimensioning* of all facilities (i.e. links, switches and routers) in the network.

In the next sections a short review of different techniques used for (link-)dimensioning is given.

2.7.1.2 Circuit Switched Networks In the context of circuit switched networks, e.g. Public Switched Telephone Networks (*PSTN* [↑]) or ATM, multidimensional Erlang formulas (see [ITU83, Kau81, LH92] for examples and [Sie96, Bau97b] for applications to network design) are used for the dimensioning of the facilities meeting the QoS constraints (such as blocking) given the topology and routing.

2.7.1.3 Packet Switched Networks In packet switched (IP) networks the nature of the transported traffic is of highest interest to calculate the bandwidth needed to transport the requirements honoring a given QoS. Measurements of IP traffic in the B-WiN[21] [Gro99] show that the average traffic per hour in comparison with the traffic in the "busiest" 60 minutes of a day, i.e. the *busy traffic hour*, differ by the factor 1.8. The DFN-Verein [Jes99] calculates with a *burstiness surcharge factor*, i.e. the over-provisioning factor of facilities in proportion to the average traffic, of 2 to 3 to transport the average (busy-hour) traffic with an acceptable QoS. Research aims at better understanding of network traffic to give a scientifically justified value for this factor (which significantly depends on the mixture of the applications that use the network). Depending on the time scale at which the network is regarded (e.g. cell-level, burst-level, call-level), other burstiness surcharge factors can be derived to meet the requirements.

In classical heuristic network design algorithms this factor is almost always 2 (which coincides with 50% utilization) and is justified by queuing theory assuming Markovian arrivals with exponential packet lengths (M/M/1–queuing formula, see e.g. [Kle75, Ker93]). Recent research [LTWW94, PF95, Rim99, Gog00], however, rises doubts about the correctness of this "rule of thumb". A quotation taken from [PF95] says:

> "Modeling network traffic using Poisson assumptions will result in analyzes that significantly underestimate performance measures such as packet delay."

The statistical nature of packet switched network traffic is subject of ongoing research. Catch words like self-similar, fractal or chaotic traffic patterns dominate the current literature, but there is still little knowledge to what degree these observations are facts or artifacts. An overview of traffic statistics illustrating different types of traffic streams, each of which belongs to a different application can be found in [Gro99, Gog00].

[21] The B-WiN was put in operation 1996 and is the predecessor of the G-WiN.

2.7.1.4 Cell Switched Networks ATM technology as an example of a cell switched network is a cross-breeding of packet and circuit switched networks. A wide-spread technique for calculations concerning ATM network traffic are *Effective Bandwidth*-formulas (e.g. introduced in [Kel96], a survey can be found in [PE96] and [JRF99]). Effective Bandwidth is a concept that aims at providing a measure of bandwidth which adequately represents a trade-off between sources of different types, taking proper account of their varying statistical characteristics and QoS requirements.

Definition of Concept 22 (Effective Bandwidth (EB) [Kel96])
Given J independent traffic sources X_j with "traffic descriptors" F_j (i.e. a set of statistical properties of the sources), the link capacity C, and constants $\llcorner\gamma, K\lrcorner$ ($\llcorner\gamma, K\lrcorner$ depend on the desired cell loss probability and given buffer size). Is it possible to impose conditions such that $\llcorner P\{\sum_{j=1}^{J} X_j \geq C\} \leq e^{-\gamma}\lrcorner$?
If the answer can be given such that if $\llcorner\sum_{j\in J} B(F_j) \leq K\lrcorner$ is satisfied, the above statement is also satisfied then $B(F_j)$ is called ***Effective Bandwidth****.*

Important properties of EB *are :*

i) $\overline{X}_j \leq B(X_j) \leq \max X_j$ *i.e. the* EB *of a source lies between its peak cell rate and mean cell rate*

ii) $B(X_i + X_j) = B(X_i) + B(X_j)$ *i.e. additivity if multiplexing different* EB*-sources*

EB-formulas therefore are a way of summarizing the statistical information of a source in a single parameter and the requirements of superimposed sources simply add up[22].

An appealing, popular conversion of this effective bandwidth concept in simple mathematical formulas is given in [Lin94]. Its practical usability in the context of a DFN-like setting is demonstrated in [FJSP99].

However, statistical traffic modeling (an overview can be found e.g. in [Gör97]) is an ongoing field of research and still has to prove its practical relevance.

2.7.1.5 IP Traffic Models Measurements on the MCI Internet Backbone show [Odl98a, Odl98b, Odl98c] that the connection facilities have light average utilization (10-15%) and some private corporate networks seem to have even less (close to 5%). The congestion experienced by the users is caused mainly by the limits of throughput in the exchange points between networks (*peering points*) and by server overload. Average utilization therefore seems irrelevant to designers of private networks when it is economically efficient to transport large chunks of data at very high speed only a few times a day. These corporate networks are built to satisfy "bursty" [↑Burstiness] requirements. Unlike voice communication, even a single connection between two hosts can occupy a high speed transmission facility all by itself. Another problem lies in the fact that in contrast to PSTN networks no busy hour can be observed. Short bursts of packets generate peaks that can be a magnitude larger than the average traffic level. With IP, one large file download can cause a huge burst in the traffic. Moreover, such small time-scale peaks can occur at any time during the day and the busy traffic hour may not contain the busiest second [BCRH$^+$99]. Therefore, the concept of a busy hour does not work well with IP traffic.

Analysis of large corporate IP network traffic indicates [BCRH$^+$99] that the variations of traffic over the time of day from one day to another is extremely high so that it could not be described

[22] There are *EB*-formulas that deliberately calculate the effective bandwidth of each mix of superimposed sources separately to achieve a better, i.e. more precise, result. But in general this improvement is payed for by the loss of linearity and therefore is not regarded here.

in terms of "extreme value engineering" (see Section 4.5.7), much less in a busy hour. A representation of **traffic variations** that does not have a busy hour is proposed: Figure 5 and Figure 6 show the distribution of the average transported traffic volume within a quarter-hour and 2-second interval which was measured on the Munich backbone-node facility of the B-WiN in spring 1999. This distribution can be described by a *normal distribution* or a composition of two normal distributions with good accuracy. The distributions of the most important underlying applications are shown in Figure 7. The first percentage value in the brackets represents the fraction of the traffic volume and the second value the fraction of the number of connections.

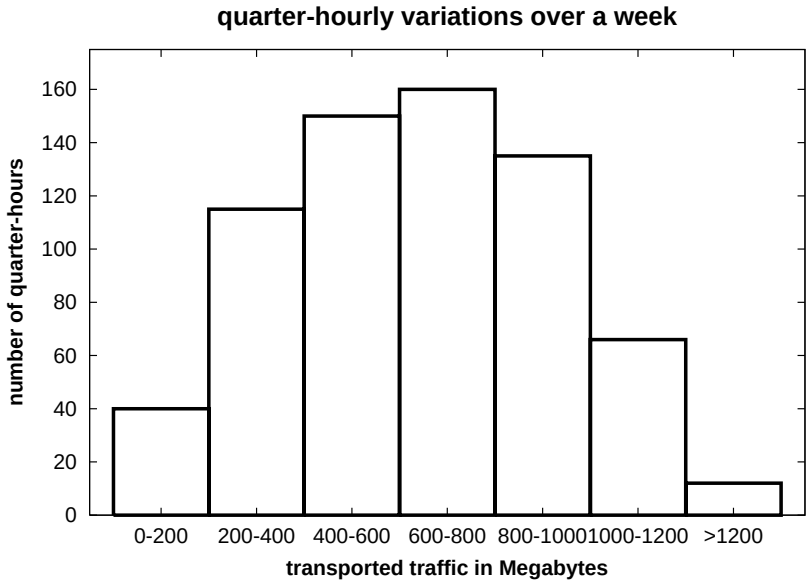

Figure 5: Load-Variations on the B-WiN (quarter-hour Interval)

The problem of dimensioning a (single) facility to transport a traffic stream with the above characteristics could be carried out by calculating the required bandwidth such that it can cope with a certain fraction of the traffic (for example meeting 95% of all requirements), and let flow-control handle overload situations, i.e. *congestion* [↑]. Regarding Figure 6, a reasonable dimensioning of the facility would be such that it can transport on average 3.5 Mbps in a 2-second interval[23].

Additional complexity will be imposed on the modeling approach when QoS transport services are introduced to TCP/IP networks (e.g. by RSVP [↑] or Differentiated Services [↑]). These subjects are open to future research.

Another, largely disregarded fact lies in the TCP/IP windowing flow-control mechanism. When an IP network is lightly loaded an application may seek more bandwidth. However, as soon as congestion occurs in an IP network packets are dropped and the flow-control may throttle throughput to a lower rate.

2.7.1.6 Summary of Traffic Modeling Throughout this thesis a requirement matrix R representing the commodities between all end-nodes in an appropriate form is assumed to be given (e.g. by the bandwidth demand in Mbps). Whether this is achieved by the use of *EB*-formulas, burstiness surcharge factors or IP models will be of no further concern. However,

[23] To achieve this, again a burstiness surcharge factor (for two-second intervals) is necessary.

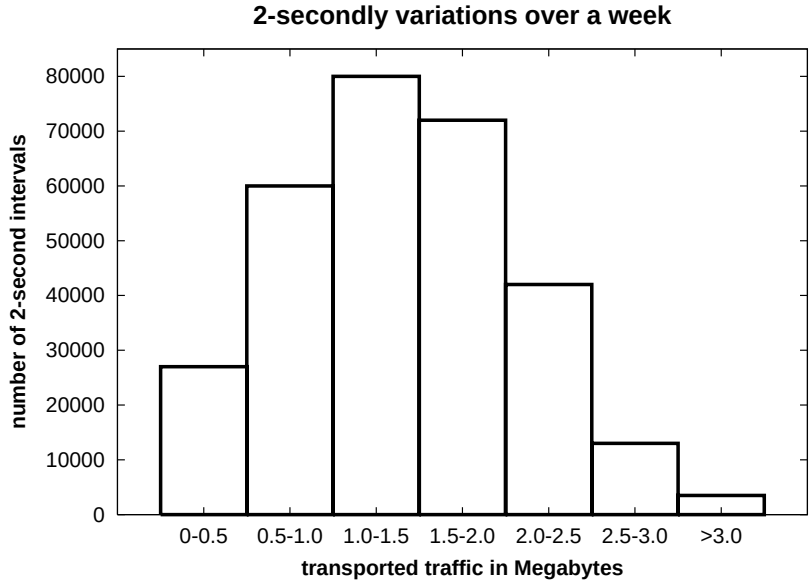

Figure 6: Load-Variations on the B-WiN (2-second Interval)

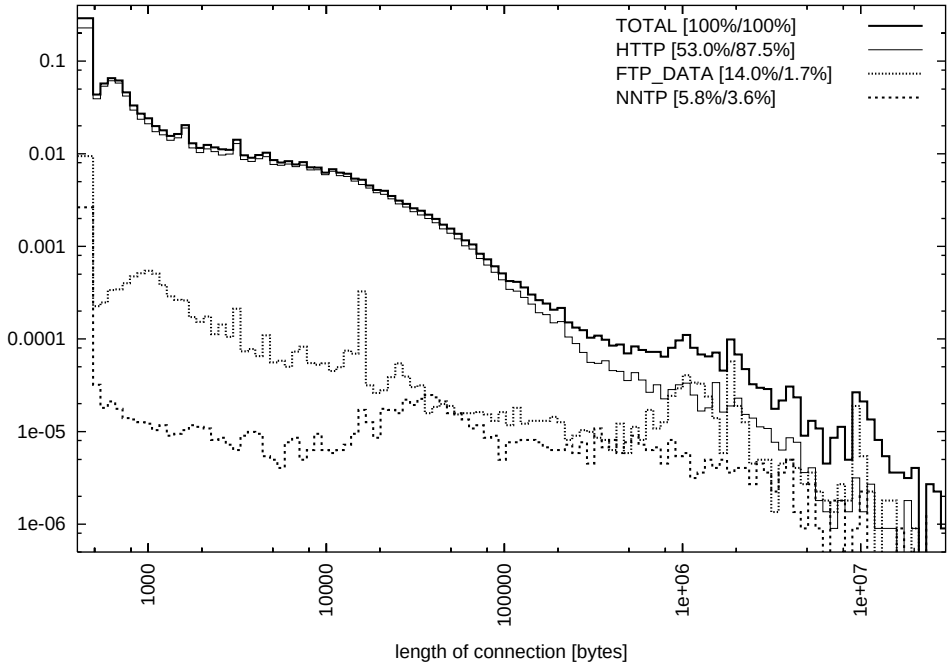

Figure 7: Empirical log-log-scale Connection Volume Histogram of Connection Lengths

it has to be emphasized that the quality of a design rises and falls with the quality of the requirement analysis.

Remark 13

In some situations, R consists of r_t-dimensional requirements $\llcorner r_{ij} \in \mathbb{R}^{r_t}, r_t \in \mathbb{N}\lrcorner$ that represent different application types or requirements at different time periods (hourly or daily variations).

2.7.2 Network Technology

Technology is a user-dependent design decision. The success of a certain network technology significantly depends on its promotion by groups or vendors and on the installed basis. Sometimes it is also a question of faith. The current wide area network technologies include Frame Relay, ATM, SDH, WDM. New photonic network technologies [↑] in general are likely to emerge in the future (see e.g. [Spä99]). The choice of a technology by a designer may also depend on its special features, such as reliability. Another, very important matter is the interoperability of a special hardware with existing facilities. The ability of a smooth migration between technologies must not be underestimated and contributes to the flexibility of the design. However, the training costs for the staff entailed with introducing new technologies and services are a factor that is usually left out in design decisions, but can sum up to a considerable amount of money.

In some cases, it may be a design constraint not to use proprietary hardware to avoid dependence on a single vendor or, on the contrary, to rely on a special feature which is provided only by a single vendor.

The choice of network technology additionally depends on the one hand on the topological hierarchical layer for which the network is planned, and on the other hand on the level of detail the corporation wants to operate its network with. For example, renting dark-fiber services entails a network planner to design the network from the physical layer including all switching and routing equipment, whereas renting IP services frees him from most of these tasks.

Choosing between network technologies is a task that is usually involves the comparison between different scenarios and under the aspect of already existing equipment. Valid scenarios for transporting IP traffic requirements could be for instance

- IP over fiber/WDM
- IP over SDH
- IP over ATM
- IP over ISPs (i.e. rent IP services from a provider)

The deployment of a network technology or certain components may impose some additional constraints on the optimization process such as ring topology or capacity limits. Usually, the deployed network technology is subject to abstraction of the modeling process, in the course of which only the key characteristics are considered.

2.7.3 Network Protocols

Most corporate networks today are designed for the efficient transport of IP traffic. Selecting a standard (e.g. standardized by the IETF [↑]) routing protocol such as OSPF imposes additional constraints on the network configuration. OSPF is a dynamic link-state routing protocol (i.e. it

routes in sink trees) and prohibits bifurcated routing. RIP [↑] as a second alternative for an IP routing protocol is as a static distance-vector routing protocol even more restrictive.

In network design, generally a static routing scheme is assumed, because it allows the direct evaluation of flows. It was shown that at *steady state* flow patterns of good adaptive routing policies are close to optimal fixed policies (see e.g. [GK77]).

New developments such as MPLS [↑], IPv6 [↑], or vendor specific protocols such as IP Switching [RFC96] and NetFlow [CIS99] may overcome some of the restrictions imposed by standard IP routing protocols while imposing new constraints on the design problem.

Emerging support for the introduction of QoS services (namely Integrated and Differentiated Services [↑]) and the demand for a higher flexibility in network design, e.g. by self sizing networks [↑] or intelligent networks [↑], introduce new paradigms such as "route at the edge, switch at the core". Vendor products like Siemens Corporate Scaled Internetwork [CSI98] or Cisco's Dynamic Transport Protocol [CIS96] take this development into account. The influence of these technologies and paradigms on the network design is still unclear. An example of a very pragmatic solution to QoS-based problems is the paradigm "throw bandwidth on it", i.e. massive *over-provisioning.*

2.7.4 Existing Facilities

Network design is usually not done on the green field. Existing facilities have to be taken into account. Why ignore an existing, already owned or leased facility that can be used to reduce network costs? Also, this can sometimes prevent a clear cut when migrating from one technology to another[24]. Considering existing facilities becomes even more important in the context of incremental or evolutionary planning that have to adopt most of the existing network structures.

2.7.5 Providers, Tariffs & Costs

Renting a single line for a short period of time is comparatively easy, renting a multi-million € network for many years is quite another thing. First of all, prices may not exist for these networks and often are open to negotiations. Second, bulk discounts play a major role and often networks are sold "as a whole" at a much lower price than the single facilities. Prices for transport facilities, generally denoted as **tariffs**[25] [↑], are quite variable, e.g. the tariff for a transatlantic connection having 90 Mbps in 1997 dropped by more than a half in 1998. (PSTN) Connection tariffs always were notorious for their odd behavior. Not long ago a telephone call from Germany to North America was three times more expensive than in the other direction. Tariffs are said to obey "Grosch's law" ([↑EoS]) which proposes that the tariff for higher capacity grows with the square root. While this may not be exactly true in practice, it still shows the general tendency that wider lines charge less per bit. This non-linearity is one of the reasons why network design problems are most difficult.

In practice, algorithms often approximate tariffs by the use of generic cost models (see for example Figure 38 in Appendix C.4) or cost generators depending e.g. on bandwidth and distance (see Section 5.3.2.4). Some network design algorithms depend critically on certain properties of the cost model, e.g. convexity. A short discussion about models of tariffs can be found in Appendix C.4).

We will assume that a tariff at least has the following characteristics:

[24] One could take the migration from Ethernet to ATM as an example.

[25] A tariff is charged by a provider for a certain service. If that service is used, its costs are imposed accordingly.

Definition of Concept 23 (Credible Tariffs)
A tariff (or costs) of a set of facilities F belonging to a node or an edge is said to be ***credible****, if it*

- *increases with capacity*
- *does not increase with time*

The topology $\mathcal{T}$ of the network depends rather critically on the underlying cost structure. Examples of two extreme cases (minimum spanning tree in case of having an Euclidian distance-dependent cost model and fully meshed in case of having a bandwidth-dependent cost model) of least cost network topologies are shown in Figure 8.

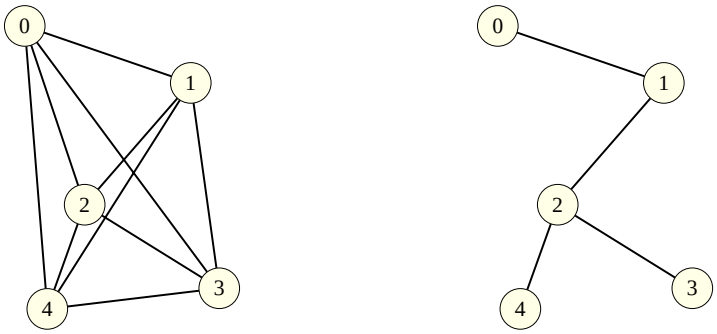

Figure 8: Example Topologies of Least-Cost Networks

2.7.6 Time

As already mentioned in the section on traffic dimensioning, the time period for which a network is designed is a very important factor in network design. By definition, static design takes into account only the present traffic requirements. However, in the last few years Internet traffic demands have been doubling each year at least. Under these circumstances a perfectly designed static minimum-cost network is likely to break down after only a surprisingly short time period. Network sensitivity testing can evaluate a design in this respect.

Also the variability in the traffic matrix may play a major role. A new host containing an intensely used service may result in a permanently, considerably changed traffic matrix. As the enterprise develops, the set of end-nodes V_e is also subject to changes by the opening or closing of departments.

Definition of Concept 24 (Variability of Traffic)
The ***variability of traffic*** *denotes its likeliness to change within a traffic matrix, i.e. the fluctuations of traffic requirements between two end-nodes.*

The classical answer to this type of problems is a massive *over-provisioning* of network facilities. The research results of e.g. [Odl98c] supply evidence for this situation impressively.

Additionally, costs and tariffs for equipment and services tend to decrease over time in an expanding and deregulated market.

Whenever projections of the future become necessary, planning under incomplete knowledge has to be a major concern. Due to its significance, a real-world design that ignores this influence is of limited use. *NOP* can cope with small-scale traffic variations as long as there is enough slack capacity in the deployed design. *IND* is a simple reactive strategy offering solutions for network design problems over time, but due to its limitations cannot take advantage of the knowledge of future developments and thus can lead to unnecessarily expensive designs. Finally, *END* is a strategy that takes future developments explicitly into account, however, it has to cope with the "resolution of uncertainties". The concepts necessary for designing cost efficient networks over time using evolutionary designs is the major concern of Chapter 4.

2.7.7 Classical & Other Constraints

Survivability is a classical constraint in network design. The outage of a single facility should not lead to an impairment of network services. Survivability can be dealt with on different layers of the topological hierarchy (see Figure 1). For example, deploying a fault tolerant SDH-APS [↑APS] technology resolves survivability on the physical layer and leads to a ring-based physical layer topology, whereas OSPF-routing deals with link failures on network layer and requires for instance an appropriately dimensioned 2-connected network layer topology.

Other, so called *organizational* constraints include e.g. the location of backbone nodes or peering-points to other networks. *Environmental* constraints can exist e.g. due to regulatory conditions or politics in general.

Sometimes, security demands burden additional constraints on networks (e.g. not to use radio-links or only use dedicated lines for a part of the traffic).

A design rule taken from [Cah98] sums it up:

> "The designer needs to be inventive and agile when dealing with unusual constraints."

2.7.8 Objective Criterion

2.7.8.1 Overview Mathematical programming aims at finding the "best" possible solution to a given problem. The *objective function* allows the comparison between two solutions with respect to the *goals* of the optimization process. The extend of meeting the goals is expressed numerically by one or more *criteria*. The objective function is a (not necessarily linear) composition of all criteria. The word optimization, in nontechnical language, is often used in the sense of **improving** while the original meaning of the word is related to finding the best. The optimal solution therefore is defined as the solution which has the best objective function value, i.e. is nearest to the goals of the optimization (see Definition 3), whereas an **optimized** solution does not claim to be optimal.

Costs in terms of currency units are the objective criterion which is used most, but not necessarily the best. In some situations, a tradeoff between two contradictory goals is sought. This might be the case if the designer looks for obtaining good QoS for a reasonable amount of money and there are a priori no restrictive *hard* constraints (see Definition 25) on the amount of money to spend or on the QoS which has to be guaranteed. Obviously, minimizing costs while maximizing QoS is contradictory.

Definition of Concept 25 (Hard Constraint)
A constraint is said to be ***hard*** *if it has to be fulfilled under all circumstances, i.e. all feasible solutions have to meet this constraint.*

2.7.8.2 Multi-Criteria Objectives The situation described above is a typical situation in network design. There are *multi-criteria*, $\llcorner(c_1, \ldots, c_n)$, $n \in \mathbb{N}_\lrcorner$ given, whereby each criterion $\llcorner i \in \{1, \ldots, n\}_\lrcorner$ is expressed as a function $\llcorner c_i : S_\mathcal{V} \to \mathbb{R}_\lrcorner$. The goal is to find a solution which fulfills all criteria to some degree. In the literature, problems with multi-criteria objectives are more graphically denoted as *frustrated problems*.

One approach to solve multi-criteria problems is to express all objectives in terms of a common measure of quality, e.g. costs, and reduce the problem to a *mono-criteria* objective

$$o : S_\mathcal{V} \to \mathbb{R} \quad x \to \sum_{i=1}^{n} c_i(x) \tag{11}$$

but this is not always possible.

Another, widely used method is *goal programming*. Two basic approaches exist:

- In the *Archimedian* approach, *penalties* are applied for not reaching the goal for each goal i as criterion c_i. The objective function aims to minimize each penalty. The problem, however, consists in weighing the importance of goals against each other, i.e. the relative importance w_i of goal c_i:

$$o : S_\mathcal{V} \to \mathbb{R} \quad x \to \sum_{i=1}^{n} w_i \cdot c_i(x) \tag{12}$$

  The weight w_i may also depend on the units in which the goals are measured.
- In *lexicographic* goal programming, the goals are ordered according to importance and priority: Criterion c_1 is more important than c_2 and so on. The goals at a higher priority level are considered to be infinitely more important than the goals in the next lower level imposing a total (strict or weak) order $\llcorner\succ: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{B}_\lrcorner$ on the solution space.

A technique extending Archimedian goal-programming is the introduction of *soft constraints*:

Definition of Concept 26 (Soft Constraint)
Soft constraints are constraints which may be broken provided a ***penalty*** *is payed in the objective function.*

A somewhat less restrictive approach than lexicographic goal programming is the concept of Pareto optimality, that imposes a partial (strict or weak) order $\rhd$ on the solution space.

Definition of Concept 27 (Pareto Optimality)
A solution is called ***Pareto-optimal*** *if there exists no other solution that is at least as good according to every objective value, and is strictly better according to at least one objective value. Having two comparable solutions $x \rhd y$, y is said to be* ***dominated*** *by x.*

Pareto optimality, however, confronts the designer with choosing between several Pareto-optimal solutions.

The interested reader may turn to [Fre86, KW97] for a further introduction to this subject.

2.7.8.3 Objectives in Network Design Generally, network design is a trading of costs versus *performance* (e.g. measured in delay, throughput, robustness etc.). The idea is to get the

"best" network justifying a certain amount of money. Whenever costs are the only objective criterion performance is usually incorporated in the model as a *hard* constraint[26]. But sometimes a slightly more expensive network offers a considerably better performance. Therefore it makes sense to use a multi-criterion objective function that represents a tradeoff between costs and performance-measures, i.e. includes performance as a *soft* constraint. The problem of appropriate performance measures (for example flexibility is hard to measure, see Section 4.5.7.2) and weighing between costs and performance remains unsolved. The following objective functions are proposed in the literature:

- maximize QoS given investment costs as a hard constraint
- minimize investment costs given QoS as a hard constraint
- maximize revenue

2.7.9 Resources

Often the network designer has limited access to resources like man-power, time (e.g. for doing exact measurements), or simply investment capital. Existing resources have to be taken into account. Limited resources usually result in constraints that prune the solution space or by an (increased) uncertainty in the planning environment.

2.7.10 Level of Detail

The designer has to specify the level of detail he requests. A greater level of detail almost always implies a higher complexity which is the reason why a tradeoff between detail and tractability is necessary. Instead of designing the network as a whole, a common approach is to design the network in succession of horizontal or vertical hierarchies (see Section 2.3.2), implicitly imposing a model decomposition.

Another level of detail is given by modeling the facilities F and their costs C_f. Due to the underlying complexity a common approach is to use generic models, i.e. model abstraction, for different types of equipment or tariffs relieving the designer (and the underlying algorithm) from the task of modeling each facility separately.

2.7.11 Available Algorithms

The repertoire of existing *SND* algorithms for the most part consists of heuristics and techniques adapted from the field of combinatorial optimization. It is not a trivial task to select a model incorporating all desirable details together with an appropriate *solver* (i.e. an optimization algorithm) which produces a feasible network design in a reasonable time, let alone finding the optimal solution. Sometimes even checking a solution for feasibility is a $\mathcal{NP}$-complete task itself. Therefore, the shortcomings of existing algorithms place a major burden on the network design process.

As already mentioned, optimization in its original meaning aims at finding the optimal solution, but in practice most network design algorithms try to find a sufficiently good (e.g. locally optimal) optimized solution.

A classification of algorithms in network design is given in Appendix A.

[26] This may not be instantly obvious, e.g. having traffic requirements that are calculated by the use of a burstiness surcharge factor.

2.7.12 Discussion

A review of the dimensions of network design discussed above results in a classification of all variables and factors into three disjoint sets: design environment, design decisions and design variables.

Definition of Concept 28 (Design Environment)
*The **design environment** $\mathcal{E}$ comprises all knowledge about input parameters and factors that is given in a network design problem. This knowledge may include the type of network traffic & traffic matrices, classical constraints, providers & tariffs, owned facilities, planning horizon (i.e. the time span over which the network will be planned), selection of objective criteria, available resources, and technology. Sometimes, there is no complete knowledge about the design environment available (e.g. the traffic matrices are not known in advance). In this case, the absent knowledge has to be completed to form a **valid** environment.*

Definition of Concept 29 (Design Decisions)
***Design decisions** are decisions that have to be made by the designer before a mathematical model can be built. Design decisions typically include selection of network technologies, network protocols, level of detail, weighing of objective criteria, and a solver. Design decisions cannot be changed by the optimization process, rather the mathematical model depends on the design decisions. Usually, design decisions are made in the course of a human-driven decision process where the network planner tries to find the best settings for a given network design problem. Decision theory and system theory, which are discussed in the next chapter, can be used to find the "best" set of design decisions.*

Remark 14
A valid environment together with a complete set of design decisions fix $\mathcal{P}$, $\mathcal{Z}$ and o of a mathematical model.

Definition of Concept 30 (Design Variables)
*The set of **design variables** $\mathcal{V}$ includes all factors that represent the degrees of freedom in the mathematical model chosen by the design decisions, i.e. form the solution space $S_{\mathcal{V}}$. Variables typically include facility selection, placement and configuration.*

It has to be mentioned that the assignment of each of the dimensions to one of the sets depends heavily on the specific context in which the planning takes place and is partly subjective. No clear distinction can be made or rules can be set up to decide which variable belongs to one set or another, e.g. the planning horizon can be in the design environment or a design variable.

The implication of the classification for the mathematical model can be summarized in the following rules:

- The design environment defines the fixed parameters of a mathematical model and contributes some constraints pruning the degrees of freedom.
- The design decisions contribute a set of constraints imposed on the solution, form the objective function, choose the level of detail, and together with the appropriate solver form the instance of a mathematical program.
- The design variables represent the degrees of freedom and together with all other constraints define the solution space.

3 A Decision Framework for Network Design Problems

3.1 Introduction

Taking the previous chapter into account, it can be derived easily that network design is a human-driven process which is very hard to automate. Frameworks for network design were already proposed in the literature (e.g. [Ber89], [TF97] and [HP98]), but few of these frameworks take time into account explicitly and most of the frameworks contain no or only superficial rules and instructions for handling uncertainties.

In the following, a framework for network design problems is proposed that provides guidelines for most situations a designer is likely to meet. In this chapter the framework is introduced having complete information about all factors that influence the design problem. The next chapter will drop this assumption, thus extending the framework to incorporate uncertainty.

3.2 The Core Planning Frame

The purpose of this section is to present a general course of action and a set of guidelines the network designer can follow that enables him to deal with all facets of network design problems. The main module of the framework is the design frame illustrated in Figure 9.

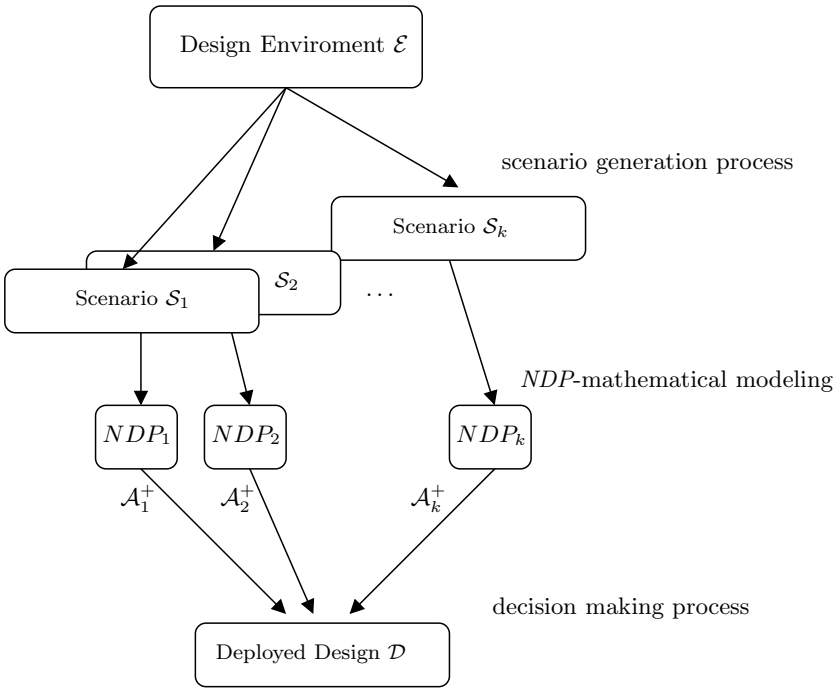

Figure 9: Design Frame $\mathcal{F}(\mathcal{E})$

First of all the following concepts have to be introduced:

Definition of Concept 31 (Scenario)
*A **Scenario** $\mathcal{S}$ is a valid design environment $\mathcal{E}$ (see Definition 28) together with a complete set of design decisions (see Definition 29). In other words, in a scenario all designer-dependent design dimensions are fixed and form a complete set of input parameters $\mathcal{P}$.*

Definition of Concept 32 (Network Design Problem (*NDP*))
A ***network design problem*** NDP *is the mathematical model of a scenario* $\mathcal{S}$ *together with a suitable solver for this model.*

Definition of Concept 33 (Feasibility, Optimality of *NDP*s)
A feasible NDP*-solution* $\llcorner x \in S_{\mathcal{V}} \lrcorner$ *is an assignment to all design variables* $\mathcal{V}$ *such that all constraints of the* NDP *hold (see Definition 3). If such an* x *exists, the* NDP *is said to be* ***feasible****. The set* $\llcorner \mathcal{A}^* \subseteq S_{\mathcal{V}} \lrcorner$ *is said to be the* ***optimal*** *set of solutions if each* $\llcorner x \in \mathcal{A}^* \lrcorner$ *has the best objective function value of all feasible solutions given a mono-criterion objective function, or is a Pareto-optimal solution given a multi-criteria objective. Note that an optimal solution is not necessarily unambiguous. In general, the solver does not produce a set of optimal solutions but a set of* optimized *solutions denoted as* $\llcorner \mathcal{A}^+ \subseteq S_{\mathcal{V}} \lrcorner$*. In fact* $\mathcal{A}^+$ *may even be empty in the case that no feasible solution is found by the solver.*

The next section will show how the design frame $\mathcal{F}(\mathcal{E})$ allows to solve an *SNDP* in a straightforward way.

3.3 An Illustration of an *SNDP*-Planning Process

To gain deeper insight into the general course of action presented in Figure 9, the WiN design will serve as an illustration for an *SNDP*.

3.3.1 Setting up the Design Environment

First of all, the designer has to collect all available knowledge for setting up the design environment $\mathcal{E}$. A specification of the WiN *SNDP* is presented in [Ull99a, Ull99b], whereas the relevant facts[27] of the valid environment are given by:

- 335 access nodes (in $\mathcal{P}$)
- busy hour requirements of IP traffic between access nodes (in $\mathcal{P}$)
- burstiness surcharge factor of 8 (in $\mathcal{Z}$)
- realization of 29 backbone nodes in each of the 29 access regions (in $\mathcal{Z}$)
- hop-limit: h hops (in $\mathcal{Z}$)
- 2-connectivity for all nodes having more than 34 Mbps traffic demand (in $\mathcal{Z}$)

3.3.2 Scenario Generation

$\mathcal{E}$ allows a multitude of scenarios, and the most promising ones should be generated by a *scenario generation process*[28]. Here, the following scenarios $\mathcal{S}_1$ and $\mathcal{S}_2$ are considered:

Scenario $\mathcal{S}_1$:

- ATM network technology, provided by the Deutsche Telekom AG

[27] Actually a slightly modified setting is used in this thesis due to the circumstance that parts of the planning task are sourced out, i.e. are left to the provider.

[28] Usually, the scenarios are manually set up by the designer or a semi-automated process like scenario analysis (see Section 4.5.2.2).

- MPOA [↑] routing
- objective criterion: a weighing between minimal discounted costs and minimal delay (for instance measured by the average number of hops)
- planning algorithm: MENTOR-II heuristic (see Appendix A.3)
- design variables $\mathcal{V}_1$: topology, capacity, routing

Scenario $\mathcal{S}_2$:

- SDH over WDM network technology, provided by Gasline
- IP point-to-point OSPF routing
- objective criterion: minimum discounted costs
- planning algorithm: Tabu Search heuristic (see Section 6.2.3)
- design variables $\mathcal{V}_2$: topology, capacity, OSPF-routing-parameters

Each scenario $\llcorner\mathcal{S}_i, i \in 1,2\lrcorner$ together with the environment $\mathcal{E}$ is transformed into a complete mathematical model and with its solver forms the instance of the static network design problem NDP_i. Note that the level of detail of the model and the capabilites of the solver may drastically influence the complexity and quality of the results.

Regarding NDP_2 in detail, the two-tier network design (see Definition 5) makes it attractive to decompose the problem into the following (independent) mathematical programs[29]:

Problem1: (star access design)

Given

- in- and outgoing traffic of each access node
- potential backbone node locations
- facility costs

minimize costs subject to

- each access node is connected to at least one backbone node
- QoS constraints are met (for instance no facilities are overloaded)

over the design variables $\llcorner\mathcal{V} \subseteq \mathcal{V}_2\lrcorner$ topology and capacity.

Problem2: (backbone design)

Given

- a set of backbone nodes
- accumulated traffic requirements between these backbone nodes
- additional potential backbone node locations

minimize costs subject to

- backbone nodes are 2-(edge)-connected
- QoS constraints are met (i.e. no facilities are overloaded)

over the design variables $\llcorner\mathcal{V} \subseteq \mathcal{V}_2\lrcorner$ topology, capacity and OSPF routing parameters.

[29]Note that the implication of the independence assumption can be weakened by recourse, i.e. an alternate optimization of the two subproblems, where results from one problem serve as input to the other until a local optimum is reached.

Remark 15

In general, the solution provided by a decomposed model is not globally optimal, even if the subtasks yield optimal solutions.

3.3.3 Decision Making Process

After all solutions $\mathcal{A}_i^+$ have been calculated, the decision making process takes over to select the scenario that will be deployed. It is reasonable that the objective functions of each scenario should be identical to safeguard comparability between the solutions. If there is a mono-criterion objective and $\llcorner \cup_{i=1}^{k} \mathcal{A}_i^+ \neq \emptyset \lrcorner$, choosing is easy by taking the best available solution, i.e. the solution $\llcorner x \in \cup_{i=1}^{k} \mathcal{A}_i^+ \lrcorner$ that has the best objective function value. In the case of Pareto-optimal solutions, it is the task of the network designer to choose one of the (incomparable) optimal solutions.

3.4 An *INDP* Framework

The *SNDP* framework can easily be expanded to an *INDP* framework by the use of two parallel design frames $\mathcal{F}$. As introduced in Section 2.5, an *INDP* has to be solved after a trigger event has occurred, i.e. a situation arises when a corporate network has to be adapted to a changed setting.

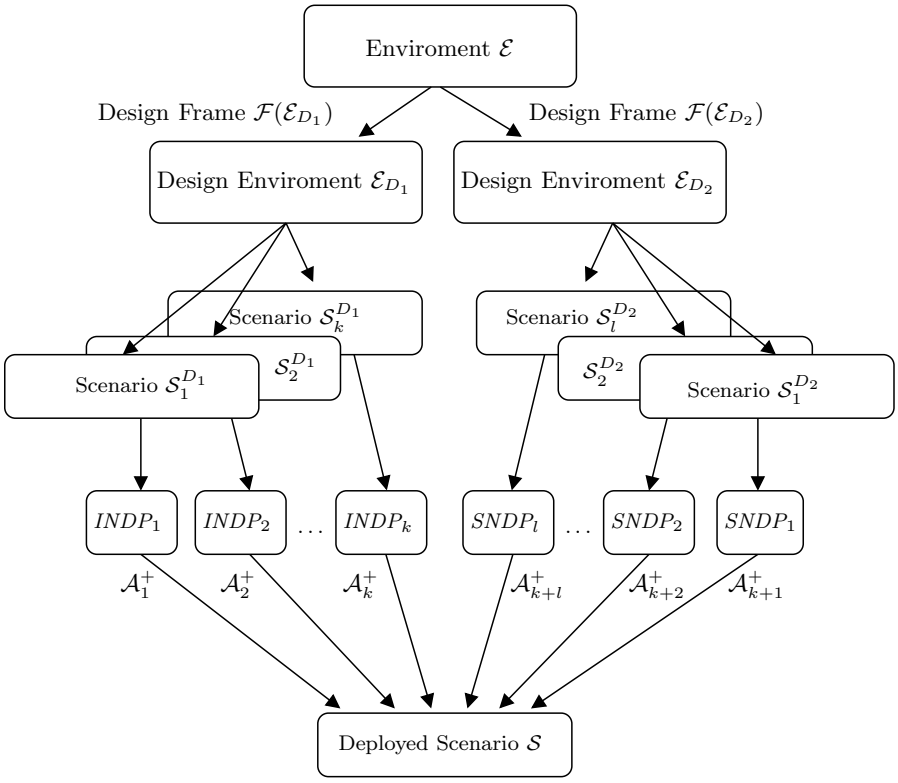

Figure 10: *IND* Design Framework

A designer has to decide between two alternatives:

D_1 Upgrade existing network; the design environment $\mathcal{E}_{D_1}$ is given by the possible upgrades of the deployed system.

D_2 Design a completely new network, i.e redesign; The environment $\mathcal{E}_{D_2}$ of this alternative consists of the *SNDP* planning environment of the changed setting[30].

The graphical framework is shown in Figure 10. The course of action coincides with the *SNDP* design framework described in the previous section.

Again it is useful to find a common objective function, e.g. costs, to compare the solutions $\mathcal{A}_i^+$ produced by the two alternatives. Given an observation period T, the associated pay-back period (see Definition 14) could serve as the decision criterion.

3.5 An *ENDP* Framework

The situation in evolutionary network design differs from *SND* and *IND* by the introduction of more than one stage that has to be taken into consideration. In the literature, this type of problems is denoted as *multi-stage* planning problems.

To propose a useful design framework for *ENDP*s some knowledge about concepts from the field of *decision analysis*[31] is necessary.

3.5.1 Decision Analysis

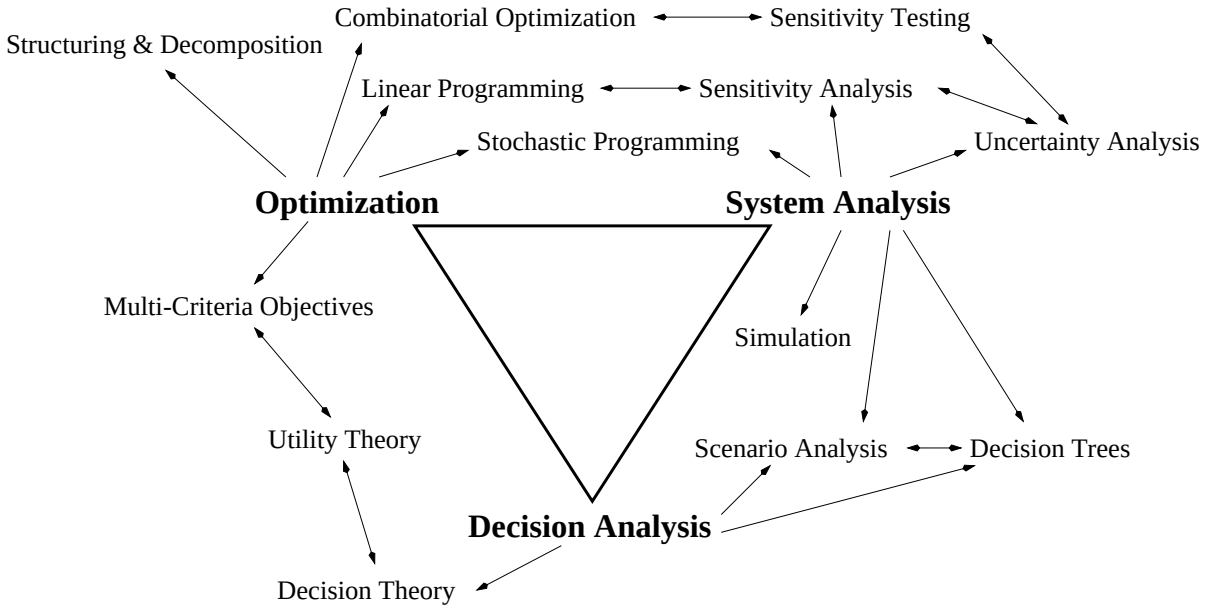

Figure 11: Overview of Concepts used in Decision Theory (adapted from [Ku95])

Definition of Concept 34 (Decision Analysis, Decision Theory)
Decision analysis *comprises the careful considerations that precede a decision. Decision analysis is a technique that structures the problem in terms of decisions and uncertain events. It encompasses a range of decision theoretic techniques, including decision trees, matrix models, and (multi-criteria) utility, whereby the term* ***utility*** *denotes a measure*

[30] Some of the already existing facilities can be naturally taken into consideration. In this way a smooth passing over from upgrade to redesign is possible!

[31] A detailed introduction into decision analysis from which the following text is derived can be found in [Ros76], [Fre86] and [Ku95].

*for a **subjective preference** for a decision. Utilities usually are a trade-off between stake, risk and gain entailed with a decision analysis.*

***Decision theory** is concerned with the task of supporting a decision maker in his task to select the "best" or "optimal" decision out of a number (but at least two) of possible alternatives (i.e. the **decision problem**). Decision theory is concerned with the understanding and formalization of decision making.*

*Optimization techniques are used most intensely in decision analysis, but decision analysis draws some of its qualities from **system analysis** that will be introduced in more detail in Section 4.5.2. The explicit representation of the decision maker's risk attitude and preferences (i.e. utility) distinguishes decision analysis from optimization.*

An overview of the relations between optimization, decision analysis, and system analysis is given in Figure 11.

Definition of Concept 35 (Multi-Stage, Multi-Attribute Decision Problem)
A decision problem is called

multi-staged or multi-period *if there is a sequence of at least two succeeding decision problems.*

multi-attribute or multi-criteria *if there is not a single but a set of decision criteria that all have to be fulfilled to some degree at least.*

Definition of Concept 36 (Decision Making Process)
The decision maker's task comprises the following steps:

1. *Structure the decision problem, i.e. find the set of **trigger events** or **stages** whenever a decision is necessary or useful over the time horizon (i.e. observation period) of the decision problem.*
2. *Define alternatives, i.e. find the set of possible **alternatives** for each stage over the observation period taking into account the impact or consequence of each alternative.*
3. *Scrutinize alternatives, i.e. check the alternatives and the **sequence** of alternatives over each stage with respect to **constraints** and possible **uncertainties**.*
4. *Assess alternatives, i.e. find a set of **criteria** that allow evaluation of alternatives at each stage and each sequence of alternatives.*
5. *Make the decision (this process is denoted as **choice**), i.e. choose one of the possible alternatives guided by a strategy (see Definition 37), or depending on other assessments, or personal preferences.*

(6.) Sensitivity analysis and testing, i.e. post-optimal analysis of how susceptible the decision is to errors in the environment the decision is made in (this will be made clear in Section 4.5.2.3).

Definition of Concept 37 (Decision Rule or Strategy)
*A **decision rule** or **strategy** is simply a specification of a procedure that prescribes exactly which **decision** (out of a set of possible alternatives) should be made in every situation a decision is necessary or useful. The "best" alternative is referred to as the **optimal decision**. Strategies in decision problems are equivalent to objective functions in optimization problems.*

According to this classification, an *ENDP* is a multi-stage, multi-criteria decision problem. Multi-staged because *END* has to solve a design problem at each of its trigger events and multi-criteria

due to the fact that network design problems have to take a lot of criteria into consideration (see Section 2.7.8). Decision trees and their extensions are the classical technique used in multi-staged decision analysis.

Definition of Concept 38 (Decision Tree, Complete Path)
Given a multi-stage decision problem with a finite number of stages and a finite number of choices, a ***decision tree*** *is a graphical representation of alternatives for and consequences of each decision.*

The mathematical formalization is given by an attributed (i.e. information is attached to nodes and edges), directed graph $\llcorner G_T = (V_T, E_T) \lrcorner$*, where* G_T *is a tree and the set of nodes* V_T *decomposes in three disjoint sets:*

- *decision point nodes* V_T^D *(symbolized by a* ***square*** $\square$*)*
  Decision point nodes represent the decision maker's choice between different alternatives. Each edge (in the context of trees also denoted as ***branch****)* $\llcorner e \in E_T \lrcorner$ *stemming from these points represents an action and is denoted as* ***action branch****. The root of the tree usually is a decision point node.*
- *chance nodes* V_T^C *(symbolized by a* ***circle*** $\circ$*)*
  Chance nodes are at the end of each action branch and edges stemming from chance nodes subdivide the tree by further edges $\llcorner e \in E_T \lrcorner$*, denoted as* ***chance branches****, one for each possible state or outcome.*
- *leaves* V_T^L *(symbolized by a* ***diamond*** $\diamond$*)*
  Leaves represent the ultimate outcome or consequence of a (unique) sequence of decisions and chances.

A ***complete path*** P_T *is a traversal through the decision tree beginning at the root and ending at a leaf.*

Remark 16
For the set of branches holds: $\llcorner E_T \subseteq (V_T^D \times V_T^C) \cup (V_T^C \times \{V_T^D \cup V_T^L\}) \lrcorner$

Remark 17
In the context of decision trees, a ***strategy****, as defined in Definition 37, provides rules to select a unique complete path in a decision tree, i.e. defines which branch to select at each decision point node.*

3.5.2 Application of Decision Trees to *ENDP*s

An *END* framework should put the designer in the position to choose between different possible deployments at time t_0 either in case of a completely new design or an upgrade of an existing design. The following information is assumed to be known in advance:

- the defined horizon model with finite observation period T being of no smaller granularity than changes in the network can be deployed.
- the valid design environment $\mathcal{E}$

The size and grade of detail of the decision tree is influenced by:

i) the granularity of T

ii) the choice of trigger events

iii) the uncertainty concerning e.g. the traffic matrix or cost structure

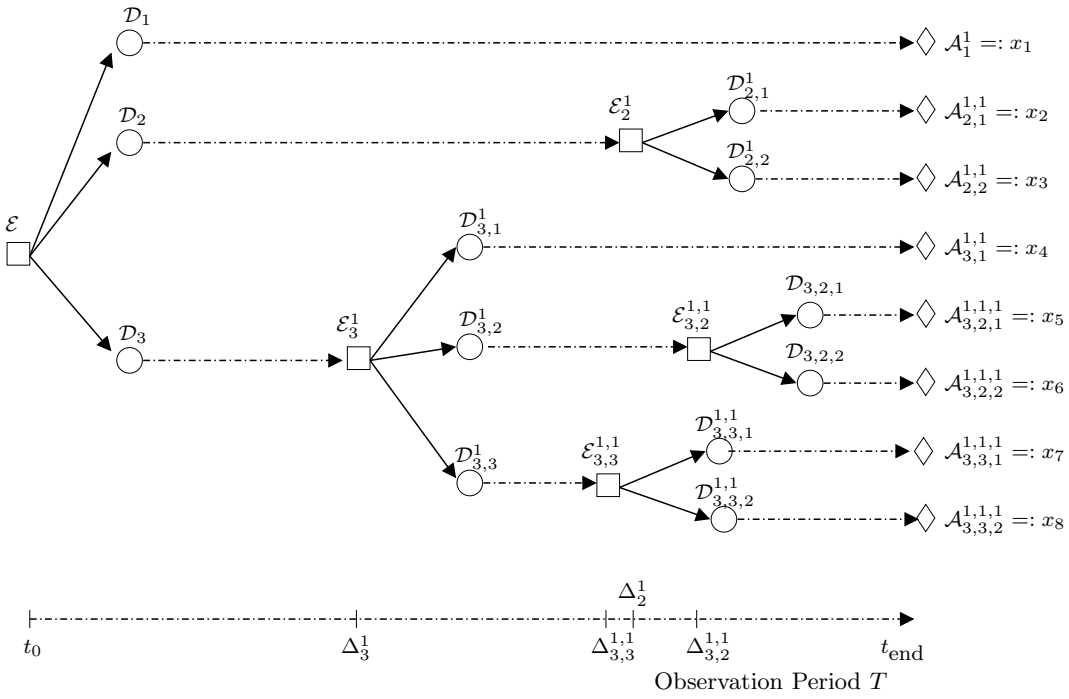

Figure 12: *END* Decision Tree (Deterministic Case)

An illustration of a decision tree for an *ENDP* is given in Figure 12. The (static) traffic matrices $\llcorner R(t),\ t \in T \lrcorner$ are known in advance and the set of trigger events consists of the violations of traffic related constraints specified in $\mathcal{E}$ which cannot be resolved by *NOP*.

Remark 18

Each decision node is labeled by the entailed environment $\mathcal{E}_y^x$ and by its observation time Δ_y^x. Each chance node is labeled by the entailed design $\mathcal{D}_{y'}^{x'}$ and each leaf by the entailed solution $\mathcal{A}_{y''}^{x''}$. The upper indices $\llcorner x, x', x'' \lrcorner$ are given by the sequence of chance branches and the lower indices $\llcorner y, y', y'' \lrcorner$ are determined by the sequence of action branches on the path from the root to that node. If the branches are not numbered explicitly, an implicit numeration from top to bottom is assumed.

Remark 19

The decision tree in Figure 12 is denoted as ***deterministic*** *because there are only deterministic chance nodes, i.e. exactly one chance branch stemming from each node $\llcorner v \in V_T^C \lrcorner$.*

Remark 20

The depth of a decision tree G_T in an ENDP *is bounded by the number of upgrade times (see Definition 19).*

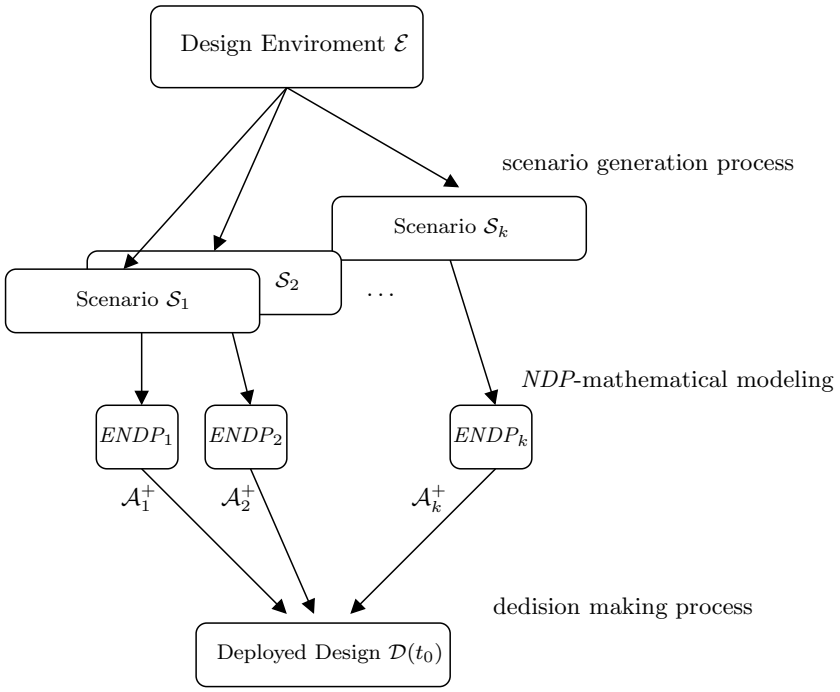

Figure 13: *END* Design Framework

The decision tree is a visualization of the sequence of (topologically different) designs $\mathcal{D}(t)$ belonging to each solution $\llcorner x \in \bigcup_{i=1}^{k} \mathcal{A}_i^+ \lrcorner$ of an *ENDP*. For every solution x there exists exactly one complete path P_T in the decision tree G_T.

The purpose of the decision tree is to help the decision maker in deciding which design to deploy at time t_0. This is achieved by the evaluation of the effects of the scenarios on the future development of the network as proposed in the framework in Figure 13. In the example above, the alternative designs at observation period t_0 are denoted as $\llcorner \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3 \lrcorner$ which again are part of at least one of the possible solutions $\llcorner x_i,\ i \in \{1, \ldots, 8\} \lrcorner$. The canonical strategy consists in deploying that design $\mathcal{D}_i$ at t_0 which lies on the path to the "best" solution x_i selected by the decision making process.

Remark 21

In real world network designs, at every trigger event Δ_y^x a new calculation of a decision tree is necessary because of a changed environment $\mathcal{E}_y^x$ and a changed framework $\mathcal{F}(\mathcal{E}_y^x)$, respectively. Examples of changes in the environment are:

- *the original observation period is prolonged*
- *new, more appropriate forecasts are available*
- *an unforseeable event has occurred that was not considered in $\mathcal{E}$, e.g. a new technology is available or a new provider enters the market.*

Sometimes it is possible to reuse some of the calculations done in the subtree rooted at the trigger event Δ_y^x.

The special quality of decision trees in environments that contain uncertain information will be discussed in Section 4.4.

3.6 Summary

Summarizing, a general framework that is applicable to all kinds of network design problems is proposed to guide the designer through this decision making process:

- In the first stage of the framework, the designer has to collect all available knowledge and eventually add some of his *expert* knowledge forming a valid environment $\mathcal{E}$.
- From $\mathcal{E}$ possible scenarios $\mathcal{S}_i$ representing different (technical) realizations are derived. Again, the designer has to contribute his expert knowledge to be able to accomplish a variety of reasonable scenarios.
- In the next step, each scenario $\mathcal{S}_i$ has to be transformed into a mathematical program together with an appropriate solver in order to calculate possible solutions of the network design problem $\mathcal{A}_i^+$. In general, these calculations do not safeguard optimality, since optimal solutions in general may not be computationally tractable.
- In the last stage, the designer has to vote for one of the designs to be deployed. A common objective function for all scenarios eases that decision.

4 Planning Networks under Uncertainty

4.1 Introduction

The author would like to begin this chapter with a quotation taken from [JV87]:

> "Rechensysteme werden anhaltend mit erheblicher Unsicherheit entworfen und beurteilt."

The task of planning and decision making almost inevitably has to deal with uncertainties. Uncertainty is a frequent subject in operations research, but approaches to handle uncertainty are still not satisfying and the problem may remain unsolved as a whole because many uncertainties are of problem specific nature. Uncertainties are one of the main reasons why planning is difficult and why strategies cannot safeguard optimal solutions. It is important to identify and understand uncertainties, especially the major ones (i.e. the *key uncertainties*), on the one hand to be aware of negative consequences, on the other hand to find out where it is useful to invest additional efforts to reduce uncertainty. But when dealing with highly uncertain environments, it may even make sense to trade optimality against "flexibility" and "robustness" to hedge against unpredictable changes.

4.2 Uncertainty

4.3 Overview

An introduction to uncertainty in the context of decision making can be found in [Ros76, Fre86, Som92, Sal98], applications of the uncertainty formalizations to real-world problems are given in [Ku95, Isu99] and the aspects of uncertainty related to artificial intelligence as well as appropriate formalisms can be found in [DAA95, RN95, KW98]. In the following, the necessary terminology is reviewed briefly and the results of the research gained in other fields of science (especially from [Ku95, Isu99]) is adapted to the context of network design problems.

4.3.1 Definition of Uncertainty

Uncertainty is a generic term used to describe something whose exact influence is not known at the present time. Uncertainty arises because of incomplete information[32] such as impreciseness or missing information. Such incomplete information may also result from simplifications and approximations that are necessary to make models tractable (*model uncertainty*). Uncertainty also refers to randomness in nature or variability in data.

Definition of Concept 39 (Internal & External Factors)
*The term "**factor**" refers to all information that may affect a decision.*

There are two kinds of factors:

1. *Factors that enter the planning process, i.e. belong to the input parameters of a scenario $\mathcal{S}$, are denoted as **planning** or **internal factors**, and as such have to be specified, approximated, or predicted beforehand although their "real value" may be quite different from their estimate.*

[32] The terms **information** and **knowledge** are used synonymously.

2. *Factors or events (some of these events can even be trigger events) that do not enter the planning model are denoted as* ***external factors****. It may be quite possible that some of the external factors cannot be predicted or even foreseen at all.*

In the literature, the terms "uncertainty" and "risk" are often used interchangeably. Here the following distinction will be made:

Definition of Concept 40 (Weak & Strict Uncertainty)
A factor is called ***weakly uncertain*** *or* ***risky***[33] *if there is additional information available, e.g. its probability distribution function (PDF) is either known or can be predicted, otherwise the factor is called* ***strictly uncertain****.*

Remark 22
As a consequence of Definition 40, weak uncertainty is a generalization of certainty, i.e. every certain factor can be regarded as weakly uncertain with precise information about its value.

It is obvious that all factors which enter a model, i.e. that are part of $\mathcal{P}$ or $\mathcal{Z}$, must be of the ***weak*** *uncertain type to be useful. Therefore, all internal uncertainties are always assumed to be of the weak type.*

Referring to the core design frame $\mathcal{F}(\mathcal{E})$ in Section 3.2, factors that enter the model used in an NDP *have to be formalized either by the (valid) environment $\mathcal{E}$ or by the derived scenario $\mathcal{S}$.*

4.3.2 Taxonomy & Classification

A taxonomy of "dis-information" which addresses all forms of "faulty" information is shown in Figure 14.

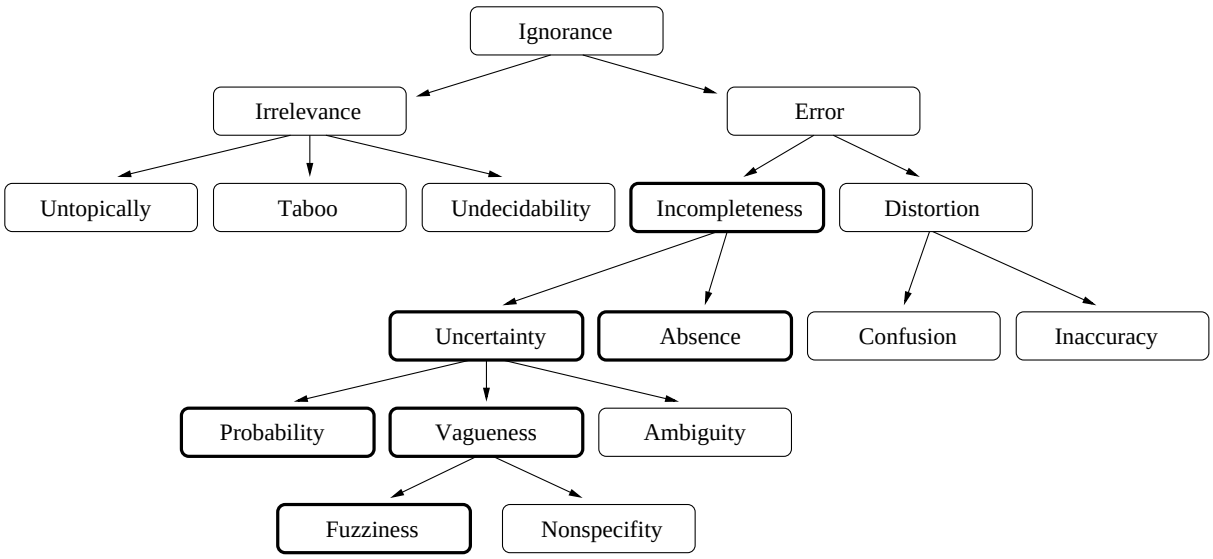

Figure 14: Taxonomy of Dis-Information according to [HP98]

Since this thesis deals with the specific problem of corporate network design, the research will focus on the uncertainties arising there. For a detailed description of the taxonomy the interested

[33] Risk as defined here is not necessarily entailed with a negative outcome.

reader may refer to [HP98].

Uncertainties will be classified using the following attributes:

Definition of Concept 41 (Uncertainty Classification)

internal vs. external: *see Definition 39.*

weak vs. strict: *see Definition 40.*

reducible vs. irreducible: *Reducible uncertainties can be resolved, at least to some degree, by additional expenditures, whereas irreducible uncertainties are unavoidably uncertain.* ***Inherently irreducible*** *uncertainties can possibly be reduced but cannot be resolved as a whole. Factors based on sampling methods are usually candidates for reducible uncertainties, whereas measurements often depend on factors that are inherently stochastic and thus of the irreducible type. The distinction between reducible and irreducible uncertainty is often a matter of convention, since it may not be feasible to reduce uncertainty of a factor beyond a certain level. Furthermore, what is perceived as irreducible natural uncertainty may be quantified e.g. with statistical methods.*

short-term vs. long-term: *Short-term in contrast to long-term uncertainties in network design apply to factors which are substantially shorter than the shortest time period to react (e.g. by reconfiguration or upgrading).*

static vs. dynamic: *Static uncertainties will not change in their qualities over time while dynamic uncertainties change with time. Traffic growth is a typical example of a (long-term) dynamic uncertainty, while the amount of (day-to-day) variablity of the traffic may be of the static type.*

smooth vs. singular or disruptive: *Uncertainties can be of the smooth type, i.e. slowly changing, or of the singular or disruptive, discontinuous type.*

quantifiable vs. non-quantifiable: *Quantifiable uncertainties can be quantified by some uncertainty formalism, whereas non-quantifiable cannot.*

predictable vs. unpredictable: *Predictable uncertainties can be assessed by their quantity and by their* ***time*** *of occurrence in the future (e.g. by forecasting), whereas unpredictable cannot.*

As it can be seen easily, most of the attributes are not of the black-and-white type but grey and "in-betweens" are possible. Besides, attributes overlap to some degree. Classification of a factor according to these attributes will give hints in which way it should be handled by the decision making process. To be useful for the decision making process, uncertainties have to be **resolved**:

Definition of Concept 42 (Resolution of Uncertainties)

The term ***resolution of uncertainty*** *denotes the description of an uncertainty factor in an appropriate way.*

Depending on the type of the uncertainty, its importance, and the abilities of the used mathematical model (i.e. its comprehensiveness), a resolution can be achieved for example by:

- *an uncertainty formalism*
- *ignorance*
- *a projection in a non-uncertain expression (e.g. its expected value or any other quantile)*

- *a modeling assumption*
- *a design decision*

4.3.3 Uncertainties in Planning and Design

The following sources of uncertainties can be identified in network design problems:

Environmental uncertainties: include all uncertainties imposed by the environment in which the planning takes place. The planning environment may include uncertainties of the non-quantifiable, unpredictable, or singular type. In the context of network design, non-quantifiable uncertainties have to be resolved to form a valid environment. Factors that are of the non-quantifiable or unpredictable type are usually left out of the model and dealt with outside of the actual design process or will serve as trigger events.

Parametric/data uncertainties: include all uncertain parameters that are input to the model. Data uncertainties have to be weak, predictable, or quantifiable to be usable in the model and may contain reducible and irreducible components. In a given scenario, the mathematical program has to be able to cope with these uncertainties.

Model uncertainty: is introduced by deciding on a certain model in the modeling process. The original uncertainty of the underlying reality may be shadowed by the model. This uncertainty is caused by limited (spatial or temporal) resolution, the level of detail, or by the modeling assumptions in general. Model uncertainties are of the internal type.

Designer uncertainties: are introduced by the influences of the designer in the planning process such as his lack of experience, selection of type of uncertainty formalization, selection of a certain modeling technique, and his personal preferences in general.

4.3.4 Uncertainty Formalisms

This section summarizes the survey of uncertainty formalisms given in Appendix B. The following concepts are addressed:

- classical set theory
- fuzzy sets
- probability theory
- evidence theory and fuzzy measure theory
- interval arithmetics

Depending on the type of uncertainty entailed with a factor, the suitability of the formalisms varies. As shown in Figure 14, there are different types of uncertainties, i.e. uncertainty is a multidimensional concept. Choosing one formalization, the modeling necessarily becomes limited to the expressiveness of that formalization. Obviously, a more general theory is capable of capturing more faithfully than specialized theories.

In particular, the following conclusions can be drawn:

Nonspecifity can be captured by classical set theory and its generalization, fuzzy sets.

Vagueness as well as **ambiguity** (i.e. lack of distinction) can be captured by fuzzy sets.

Probability can be captured by probability theory, as well as generalizations such as fuzzy measurement theory.

Absence as another instance of uncertainty falls into the category "strict uncertainty" (see Definition 40). It can be resolved in different ways, such as ignorance, the principle of indifference, or the use of default values. It exists outside of the classification of Definition 41 and its resolution is the task of the designer.

It is interesting to realize that as a result of the famous Gödel theorem, even mathematics itself has to deal with incomplete of information (however, with different context).

Today, probability theory and fuzzy set theory are the techniques chosen most often for coping with uncertainty. Nevertheless, a uniform theory of uncertainty incorporating all of its possible dimensions is still missing. Generalizations such as fuzzy measure theory are likely to replace concepts that are restricted to single types of uncertainty, but the theoretical development does not meet the practical needs due to lack of understanding and concepts are still subject to ongoing research.

4.4 Uncertainties in Network Design Problems

4.4.1 Motivation

A lot of publications addressing the problem of how network technologies, network traffic, applications etc. will develop is available. Almost every important company, research institution, or single expert has different ideas. Some authors have concerned themselves with the problem of uncertainties in network design (e.g. [GP87, Min93, CP95, TF97, DH98]), but no real progress has been made except addressing the problem and giving some either very general, or very specific concepts handling it. As an illustration, [DH98] is quoted:

> "A multitude of choices, the unpredictability of future technologies and traffics growth, the emerging of new applications, and the need to operate in heterogenous environments demand a concept taking these in the decision making process and allowing evaluation of different alternatives.
> Exponential growth [with an uncertain exponent!] coupled with a high degree of uncertainty in future traffic patterns call for new approaches to planning network technologies, topologies and capacities."

However, a thorough overview and classification of uncertainties arising in network design problems is missing in literature. Here the author wants to give an overview to fill this gap.

4.4.2 Classification and Assessment of Uncertainties

4.4.2.1 Introduction Uncertainties accompany a designer in almost every step of his design task. The following sections will elaborate the uncertainties arising in network design guided by

the dimensions of network design introduced in Section 2.7. Equally important to the formalization of uncertainties is their assessment with regard to their effects on the network design process and, correspondingly, either their resolution by the designer, by the model, or by the design process.

4.4.2.2 Network Traffic The stochastic nature of network traffic is influenced by user behavior as well as protocols, respectively applications. Therefore network traffic is an inherently irreducible but predictable uncertain planning factor.

The first question a designer has to answer is how much bandwidth is necessary to transport a given traffic stream over a facility with a certain QoS. The characterization of the statistical nature of traffic is an ongoing research topic over many decades beginning with works by Erlang 1909 and Kleinrock 1964.

Modeling network traffic, i.e. network traffic characterization, is done by selecting appropriate (statistical) models and calibrating them by measured parameters, if available. Obtaining reliable estimates of network traffic is the most burdensome task in network design. When measurements are lacking, assessments of experts (e.g. by the network designer himself) have to substitute this absence of information. However, due to the changing mix of applications in the network (e.g. the portion of multimedia applications is expected to grow in the near future in many corporations) the statistical nature of traffic is a dynamic uncertainty. Therefore, the designer has to verify whether the traffic characteristics conform with the applied traffic models in regular intervals. Almost unconsciously, this type of uncertainty has been addressed and resolved by *static* network design in Section 2.7.1. In summary, the characterization of traffic is resolved by the selection of statistical traffic models and is therefore both, a designer and a model uncertainty.

Another problem lies in the fact that not only the overall network traffic is important, but also its distribution among the end-nodes.

Definition of Concept 43 (Periodic & Persistent Variability in Traffic Patterns)
Traffic variability (see Definition 24) can follow a periodic pattern (denoted as ***periodic variability****), e.g. a over a day or week, or can cause a disruptive, permanent change in the traffic matrix (denoted as* ***persistent variability****), e.g. due to introduction or request of a single bandwidth intensive service.*

Periodic variability is (like the network traffic itself) an inherently irreducible, predictable uncertainty and can be used by sophisticated network design algorithms to achieve more cost efficient designs when several busy-periods are observed (e.g. leading to multi-hour designs in voice networks). As already mentioned in Section 2.7.1, data networks lack the concept of a strongly developed "busy-period". The periodic variability is usually resolved "on-line" by dynamic routing schemes or over-provisioning of bandwidth.

The persistent variability of traffic is an unpredictable uncertainty which likewise belongs to the traffic and time dimension. Whenever a new service goes online or an existing service moves from one location to another, this can result in a considerably changed traffic matrix. Persistent variability also arises when end-nodes come online, move, or shut down. *NOP* can cope with this development to some degree, but an a priori optimal design built for meeting the initial requirements will probably be far from optimal after a few larger changes in the traffic matrix. However, it may be difficult to decide whether a variability is persistent or only a disappearing, short-time disruption. Constant monitoring of network traffic can help to decide if a change in the traffic matrix is significant.

Persistent variability as a disruptive, unpredictable, and non-quantifiable uncertainty is classified as an external uncertainty, and is usually not considered in network design models. However, it makes sense to regard the ability of a design to cope with traffic variability as an evaluation criterion. This conforms to the notion of *robustness* and *flexibility* which will be introduced in Section 4.5.7. Incremental and evolutionary network design may be triggered by the changes caused by persistent variability. Persistent variability and traffic growth coincide to some degree.

4.4.2.3 Time The key uncertainties of evolutionary network design are mainly caused by time dependent changes in the environment.

The growth of traffic over time is a classical example of a long-term, dynamic uncertainty. It is most critical in the context of network design because a single underestimated application can have a disastrous impact on the network traffic growth in the long run (like the introduction of the World Wide Web in 1994). One reason for this sensitivity is that, in contrast to voice services, a single traffic source can consume a considerable amount of the installed bandwidth[34]. Obviously, growth of network traffic is one of the **key uncertainty** in *ENDP*s.

The uncertainty in traffic growth is resolved by **forecasting**, i.e. a projection of developments from the past into the future. Subjective judgments have to be made by people who know the subject very well (experts). Then, together with forecasting history, they put forward forecasts as their considered opinions. There are numerical methods that can be used for forecasting but they are only aids and do not replace the subjective judgment. These numerical methods include:

- judgment techniques[35] (jury of experts, user expectations, or pilot trials) [Ros76]
- heuristics (Kruitoff, or Weighted Least Squares) [ITU92a, ITU93]
- curve-fitting (correlation, or extrapolation) [BD89]
- modeling methods (e.g. econometric & logistic models) [Wöh90, ITU92b]
- time series analysis (e.g. Box & Jenkins ARIMA-models) [Ham94]
- state space models (e.g. Kalman filtering) [PW82, Kno89]
- neural networks [HSV+99]

Note that this is not a crisp classification (e.g. logistic models use curve-fitting for calibration).

Sophisticated forecasting techniques only make sense in situations with less uncertainty, i.e. low growth rates. The more uncertain the settings are, the more the designer has to rely on the opinion of experts. An additional complicating influence which has to be kept in mind is the **traffic stimulus phenomenon** [ITU83], which states that improvement of networks stimulates its usage over-proportionally (i.e. if spare capacity is available it is likely to be used up).

Usually, forecasting is subdivided into different time-scales with different degrees of uncertainty. In the following, three different time scales will be used:

Short-term forecasts: span from a few weeks up to six months and allow prediction with good accuracy.

Medium-term forecasts: span from six months up to one year and allow forecasts with acceptable accuracy.

[34] High quality video applications like HDTV can easily require hundreds of Mbps!

[35] These techniques also include non-numerical parts and are denoted as semi-numerical or semi-subjective.

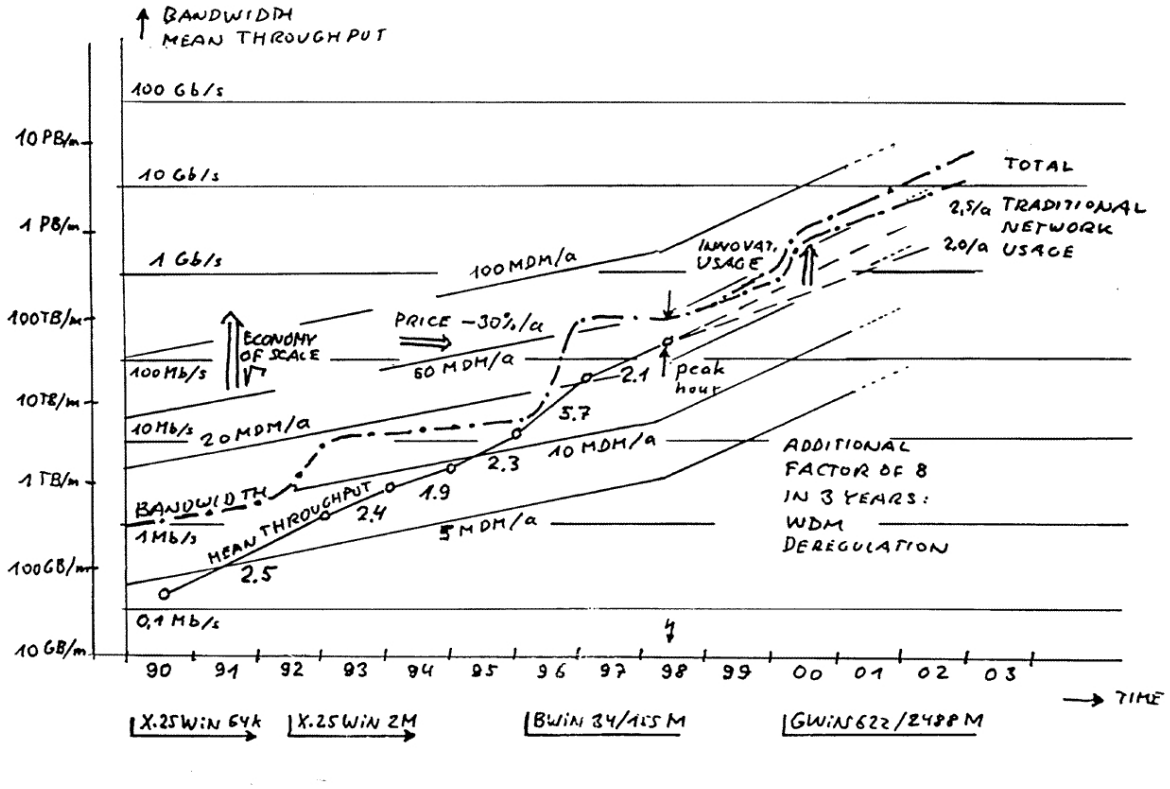

Figure 15: Expert Planning Scenario from [Jes99]

Long-term forecasts: cover more than one year and are important for strategic goals, but do not allow an accurate prediction.

It has to be mentioned that this time scale is subjective. In general, the following statement holds: The more certain the design environment is, the longer accurate forecasts can be made, influencing the length of time-scales "short", "medium" and "long". For example, in the context of planning in voice communication, long-term forecasts can span up to decades. But as already mentioned, the pace of technological development and traffic growth is much slower in voice communication. Traffic development therefore has a *predictable short-term* part and becomes more and more *unpredictable* with longer time periods.

Whenever possible, the forecasts should predict lower and upper bounds, most likely value, confidence limits, or even fully qualified probability distributions. If the opinions of experts are inquired they should be converted into probability distributions. These probability distributions are called *subjective* or *Bayesian* (i.e. the probability is the degree of a person's belief that an event will occur). Semi-subjective methods include the results of field trials, expert discussions, and many more. The assessment of the bounding of traffic forecasts is as important as the forecast itself.

Figure 15 shows the results of a discussion of experts concerning the development of traffic growth in the German Wissenschaftsnetz [Jes99] in June 1998. The traffic forecast is accomplished by the extrapolation of an exponential function (i.e. curve-fitting) assuming an expected growth rate between 2.0 and 2.5 per year, whereas the upper and lower bound is predicted by growth rates of 2.0 and 2.5, respectively. In the figure, the upper and lower bounds form a funnel with the expected traffic growth in its middle. The transformation of the prediction into a probability

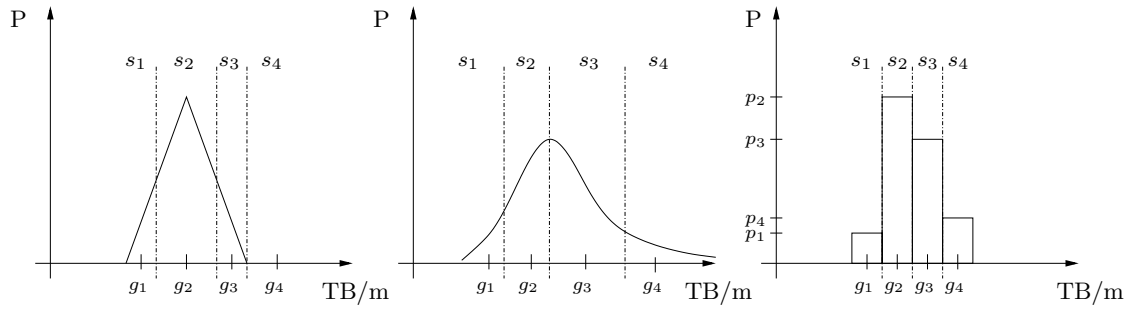

Figure 16: Triangle, Bell & Discrete Probability Distribution of Traffic Forecasts

distribution could result in one of the distributions shown in Figure 16.

Summarizing, uncertainty in traffic growth is resolved by forecasting, resulting in a probability distribution for each observation time in the observation period. Traffic growth becomes an internal factor of the *END* model that initiates an upgrade whenever the network is overloaded. If traffic growth is forecasted for a network as a whole, the traffic has to be transformed into a traffic matrix (e.g. by the weighted least squares method). In the past, forecasts of network traffic were notorious for their lack of accuracy. This observation supports the need for the concept of *robustness* and *flexibility* that will be introduced in the following section.

Additional time-dependent factors in evolutionary network planning are inflation, the cost of capital, and facility costs. Both, cost of capital and inflation are subject to smooth time-driven changes and can be predicted with great certainty (especially compared to traffic growth), but usually are no key uncertainties, so they are resolved by assuming a deterministic (expected) value.

Costs of facilities drastically influence the evolutionary network design $\mathcal{D}^T$ and can be one of the key uncertainties. Therefore, they are subject to a detailed prediction. For example, the availability of Wavelength Division Multiplexing equipment is expected to lower the prices for high bandwidth connection services considerably. The chart in Figure 15 shows a cost prediction which assumes a cost reduction of factor 8 in three years due to this development. On the contrary, costs for T1 [↑SDH] facilities were growing unexpectedly in the Unites States during 1998 due to high demand. In some cases, the designer can protect himself against sudden changes in tariffs by long-term contracts, but this also diminishes the possibility of taking profit of cost reductions on short notice.

4.4.2.4 Network Technology Network technology was changing very rapidly in the past and this trend is expected to continue in the future. Evolutionary network design takes place in heterogenous equipment environments, where more than one generation of equipment is used. For example, due to the enormous investments necessary in nation-wide or world-wide telecommunication infrastructure, Deutsche Telekom's reorganization from analog to digital technology lasted more than ten years. Network technology also has a time-dependent long-term uncertainty.

When deciding for one technology, it may be difficult to predict whether this technology is on the market in a few years at a reasonable price. The assessment of the ATM technology is a good example of this conflict. Some experts predict that ATM will play a role only in very specialized environments and therefore will stay very expensive, whereas others see it as the all-embracing technology that will control the market in a few years and will become very low-priced soon. The prices for communication equipment needed to materialize a corporate network can contribute a substantial amount of network costs. Depending on the deployed technology, costs may decrease in the future with high probability but the development stays uncertain.

Summarizing, network technology is a non-quantifiable, internal uncertainty which has to be resolved by scenario analysis (introduced in Section 4.5.2.2) and a prediction of the development of the market.

Network facilities themselves contribute to the model uncertainty if abstractions regarding their abilities are used, i.e. by the use of generic facility models.

4.4.2.5 Network protocols An additional model uncertainty in network design is based on the assumption of static routing if a dynamic routing protocol is used in the real setting. However, flow patterns of good adaptive routing policies are shown to be close to optimal fixed policies.

Recent developments like Integrated Services [↑] or Differentiated Services [↑] introduce new paradigms in IP networks. The exact influence of these paradigms on network design that allow Traffic Management [↑] to a much higher degree is still subject to future research.

4.4.2.6 Objective Criterion As elaborated in Section 2.7.8, usually, there is no single objective criterion in network design, but a multi-criteria objective weighing costs versus performance. The demand for the "best" network justifying a "certain" amount of money makes clear that no undisputed idea exists how to weigh the criteria against each other. The terms "best" and "certain" are classical fuzzy terms and also depend on the subjective assessment of a designer.

The uncertainty in the objective criterion is introduced by fuzzy terms and therefore cannot be formalized by probability theory. It is a designer uncertainty that has to be resolved by the planner, e.g. by the use of multi-criteria objectives as described in Section 2.7.8.2. One way to relieve this unsatisfactory situation is to use different objectives to produce different scenarios, but this aggravates the decision making process.

4.4.2.7 Algorithms The main uncertainty introduced by algorithms is their gap between the solution produced and the optimal solutions of the mathematical model. As Table 2 in Appendix A indicates, all algorithms that are computationally tractable do not guarantee optimal solutions. Even if lower bounds exist, they have to be reasonable small to be of any use.

Another kind of uncertainty which is entailed with meta-heuristic search algorithms lies in the calibration of the heuristics, e.g. the selection of the problem representation or termination criteria.

4.4.2.8 Tariffs Tariffs, analogous to facilities, are typical candidates for abstraction by generic cost models (some cost models are discussed in Appendix C.4) and therefore belong to model uncertainty. However, costs are notorious for their odd behavior (see Section 2.7.5) and a thorough investigation whether cost models can be applied is necessary.

Another uncertainty lies in the customary bulk discounts every vendor offers to customers. In some situations, accurate tariffs do not exist. Bulk discounts which are subject to negotiations are non-quantifiable and therefore evade formalization.

4.4.2.9 Resources As already mentioned in Section 2.7.9, limited resources can contribute to the uncertainty of an environment, e.g. the lack of measurements due to limited time, man power or budget.

4.4.2.10 Level of Detail Obviously, the level of detail belongs to the model uncertainty. The resolution usually consists of the decision for a certain level of detail by the designer.

4.4.2.11 Other Other uncertainties arising in network design are for instance:

- regulatory influences
- disasters
- change in organizational constraints
- vendor goes out of business or drops product line
- a new competitor enters the market
- a new technology becomes available
- non-renewal or breach of contracts
- a new, unexpected bandwidth-consuming application comes online
- customer budget or business activities are reduced

These uncertainties have to be treated as unpredictable, external, non-quantifiable, and disruptive since they remain inaccessible to prediction or formalization in general. Disruptive changes usually call for reaction at the time they arise and therefore are regarded as external trigger elements.

factor	type of uncertainty	resolution proposal
traffic characterization	predictable, inherently irreducible, internal, dynamic	stochastic traffic models
periodic variability	predictable, inherent irreducible, internal	busy traffic period, (dynamic routing)
persistent variability	unpredictable, external, disruptive, non-quantifiable	external trigger event, robustness & flexibility
traffic growth	predictable, smooth, long-term, internal, dynamic	forecast, internal trigger event, robustness & flexibility
costs & tariff development	predictable, smooth, long-term, internal	forecast, internal trigger event, robustness & flexibility
network technology	non-quantifiable, internal	scenario analysis
network facilities	model	abstraction by generic cost models
routing	model uncertainty	static routing
objective criterion	fuzzy	designer
algorithm	model	designer
modeling costs & tariff	resolvable, quantifiable, model	abstraction by generic cost models
level of detail	model	designer
other	unpredictable, external, disruptive, non-quantifiable	external trigger events

Table 1: Uncertainty Classification and Resolution

4.4.2.12 Summary and Resolution Proposal Table 1 gives a summary of the classification of uncertainties. Since it does not make sense to spend much effort on the reduction of uncertainties that influence the network design only slightly while the key uncertainties are high, the designer can concentrate on those. The candidates for key uncertainties are printed in bold face.

Usually, in *SNDP* and *INDP* the key uncertainty consists of model uncertainty such as traffic characterization, costs & tariff models, and the objective criterion. In *ENDP* forecasting of traffic growth and the development of costs & tariffs play the major role. The characteristics of network design algorithms contribute to the uncertainties of all types of *NDP*s.

The resolution of uncertainties depends critically on the environment in which the planning takes place. The following general guidelines for dealing with uncertainties are proposed:

Resolution by the model: Traffic characterization can be resolved by stochastic traffic models if there is accurate information about traffic between end-nodes available, however, it becomes a key uncertainty that cannot be resolved by stochastic traffic models alone if no such information is available. Other candidates for uncertainties that can be resolved by the modeling process are network facilities, tariffs & costs, and routing. Periodic variability is shadowed by the concept of "busy traffic period" and therefore resolved by the model, too.

Resolution by the designer: It is the task of the designer to choose the objective criterion, the level of detail, and the solver. Also, network technology, forecasting traffic growth, and cost development are uncertainties the designer is responsible for resolving.

Resolution by trigger events: Two different kinds of trigger events are distinguished. **Internal trigger events** are subject to uncertainty resolution and therefore (like internal uncertainty factors) enter the planning process. Traffic growth and cost development are candidates for internal trigger events. **External trigger events** may also cause a reconfiguration of the network at the time they occur, but due to their nature cannot be considered in the planning process. Disruptive, non-quantifiable uncertainties in general fall into this category.

Persistent variability on a small scale can be viewed as a special case of traffic growth, on a larger scale it has to be resolved by an external trigger event.

Resolution by flexibility and robustness: The notion of flexibility & robustness can be motivated by the following remark:

Remark 23
In an environment with many potentially unpredictable and disruptive uncertainties, a network design that can be adapted easily and with small costs may be more favorable than an optimal network design that is costly to change.

The associated technique "flexibility engineering" will be introduced in Section 4.5.7.

4.5 Fundamental Concepts in Optimization and Planning with Regard to Uncertainty

4.5.1 Motivation

All concepts regarded in the following sections rely on a probabilistic representation of uncertainty. The attribute "uncertain" to an input or output parameter is always entailed with

formalization by a probability distribution function (PDF). Therefore, the term **deterministic** denotes a situation without uncertainty. The introduction of random elements such as PDFs departs from the deterministic view of the world. Any technique that assigns probabilities to input parameters of a model and calculates the likelihood of resulting outcomes is called a **probabilistic analysis**.

The purpose of the following sections is to discuss and assess concepts which allow to include uncertainty in optimization and decision making while keeping an eye on the core problem. The author focuses on the following concepts[36]:

- Extension of deterministic optimization techniques
- Stochastic Programming
- Stochastic decision analysis
- Flexibility and Robustness

4.5.2 Important Terms and Concepts: System Analysis

By common definition, system analysis is the systematic investigation of a real or planned system to determine the characteristics of the system and how they relate to each other. System analysis consists of the following core techniques: uncertainty analysis, scenario analysis, sensitivity analysis, and simulation (see Figure 11). The main concepts will be reviewed briefly in the next sections. Some of the definitions are adapted from [Isu99].

4.5.2.1 Uncertainty Analysis

Definition of Concept 44 (Uncertainty Analysis)
Uncertainty analysis *shall provide insights into the degree of confidence in the factors in a given environment, scenario, or model. Uncertainty analysis will not reduce any existing uncertainties, its task is to aid the designer in the identification of the* ***key uncertainties*** *in a model.*

Uncertainty analysis usually comprises the following steps:

Uncertainty formalization: estimation and appropriate formalization of uncertainties by input parameters or by a selection of different scenarios.

Uncertainty propagation: finding a way to efficiently propagate the uncertainties into the underlying model, e.g. by sensitivity analysis (see Section 4.5.2.3).

Model uncertainty influence: characterization of uncertainties caused by the model structures and model formulations.

Ranking of uncertainties: the ranking of input uncertainties, allowing to focus on the important and highly sensitive factors (**key uncertainties**).

Uncertainty factors are distinguished by their role in the planning process: Internal factors are subject to sensitivity analysis (introduced in Section 4.5.2.3), that is, changing one value at a time; External factors are varied in the context of scenario analysis (introduced in Section 4.5.2.2).

The discussion in Section 4.3.3 is part of an uncertainty analysis of network design problems.

[36] Other possible techniques are for example stochastic dynamic programming or game theory ([↑]).

4.5.2.2 Scenario Analysis

Definition of Concept 45 (Scenario Analysis)
Scenario analysis *is a term used for the generation and evaluation of different settings by the resolution of (external and internal) uncertainty factors of a design problem. It is a less formal and less structured method because it makes use of the subjective judgment of the designer intensively.*

Scenario analysis generates views of the future, typically including the most likely, expected, and extreme (worst or best) settings.

Remark 24
The difference between scenarios generated by scenario analysis and a scenario $\mathcal{S}$ introduced in Definition 31 lies in the fact that in a scenario $\mathcal{S}$ all *external uncertainties have to be resolved.*

While scenario analysis covers a range of possible futures, it may not take into account fluctuations within or deviations from each scenario. Sensitivity analysis is needed to explore the uncertainties in individual parameter values and variations of scenarios. The problem of scenario analysis lies in the fact that it depends critically on the abilities of the designer to create plausible scenarios. Automation is only possible if the corresponding uncertainty factors are subject to useful formalization.

4.5.2.3 Sensitivity Analysis After the uncertainties in the input parameters to the model are determined, the challenge consists in **propagating** the uncertainty factors through the chosen model, i.e. getting an idea of how the uncertainty in the input parameters influences the output parameters of the model. Sensitivity analysis is the most common method for performing uncertainty propagation.

Definition of Concept 46 (Sensitivity Analysis, Key Uncertainty)
The aim of sensitivity analysis *is to estimate the rate of change in the output of a model with respect to changes in input parameters, e.g. correlation. Such knowledge is important for*

- *evaluating the applicability of the model (i.e.* comprehensiveness *of the model)*
- *identifying parameters which have the strongest influence on the solution, i.e. for which it is important to have the most accurate values (key uncertainties)*
- *understanding the behavior of the system being modeled (i.e.* comprehensibility *of the model)*

Weak sensitivity analysis *determines how much the output parameters vary with variations in the input values but gives no indication of their relative likelihood (e.g. achieved by interval arithmetics, see Appendix B.5), whereas* ***strong sensitivity analysis*** *describes the likelihood of an output variable to take on a specific value.*

Based on the choice of a sensitivity metric (e.g. variance, or confidence intervals) and the degree of uncertainty in the factors, sensitivity analysis techniques can be broadly classified into the following categories:

Differential analysis: requires analytical models and involves either the differentiation of equations in the mathematical program and subsequent solution of a set of auxiliary

sensitivity equations, or the reformulation of the original model using stochastic algebraic/differential equations. Due to the fact that network design problems usually do not involve analytical techniques (see Appendix A.4 or Section 6.1.1), these methods are less important in network design.

Sampling based methods: involve running the original model for a set of input parameters (sample points) or with straightforward changes in model structure (e.g. in model resolution) and estimating the sensitivity/uncertainty using the model outputs at those points. Well-known examples of sampling based methods are e.g. *Monte Carlo Sampling* [↑] or *Latin Hypercube Sampling* [↑].

Conventional methods of uncertainty propagation using sampling based methods typically require several model runs that sample various input values. The number of runs can sometimes be very large, resulting in intractable computational demands. Therefore, uncertainty analysis performed by Monte Carlo Sampling or Latin Hypercube Sampling may not be efficient or even feasible for complex models. However, sometimes it is possible, depending on the structure of the underlying problem and the type of desired results, to reduce the number of model runs drastically (e.g. by Stratified or Importance Sampling).

Model based sensitivity analysis: uses the fact that some mathematical models allow direct propagation of uncertainty information through the model, i.e. information about sensitivity is a part of the model output parameters.

Sensitivity testing: involves studying the output of a mathematical model for a set of changes in model formulation, and for selected variation of input parameters. Sensitivity testing does not exactly conform with the definition of sensitivity analysis, but makes sense in cases where sensitivity analysis is not applicable. The network sensitivity analysis proposed in Definition 16 is an example of sensitivity testing.

Another problem that is entailed with sensitivity analysis based on probabilistic sampling or testing parameters is the lack of information about dependency between them. Usually, input parameters which are sampled independently from each other may lead to distorted results. However, the computation slows down exponentially as more parameters are varied simultaneously, which is more graphically denoted as "the *curse of dimensionality* in coping with multi-variate distributions".

4.5.3 Extension of Deterministic Solving Techniques I: Post-optimal Sensitivity Analysis

Deterministic mathematical programs as introduced in Appendix A.5 as well as the other algorithms introduced in Appendix A do not have any capabilities to cope with uncertainty directly. However, the possibility of coping with uncertainties would contribute to their comprehensiveness.

Scenario analysis together with **post-optimal sensitivity analysis** or **post-optimal sensitivity testing** is an intensely used technique that allows to propagate uncertainty in deterministic solving techniques at least to some degree.

It comprises the following two steps:

1. Scenario generation: The idea is to generate a set of scenarios $\llcorner\{\mathcal{S}_1, \ldots, \mathcal{S}_s\}\lrcorner$ by the use of scenario analysis resolving all (external and internal) uncertainty factors and thereby covering all important possible settings that allow a deterministic solver to produce a solution $\mathcal{A}_i^+$ for each (deterministic) scenario.

2. Post-optimal sensitivity analysis: Each of the solutions $\llcorner x \in \mathcal{A}_i^+ \lrcorner$ is submitted to (post-optimal) sensitivity analysis for evaluation of how the output parameters behave with respect to changes of input parameters. An appropriate sensitivity measure must be determined.

Two different kinds of sensitivity analysis exist:

Global sensitivity analysis: Sampling based methods like Monte Carlo sampling are used to produce different settings of the input parameters, allowing evaluation of the behavior of each solution over a wide range of possibilities.

Local sensitivity analysis: Sampling based methods are used to generate a set of input parameters that are in the local neighborhood[37] of the initial input parameters to see how the solution behaves with respect to small changes in the input parameters.

Each modified setting generated by sensitivity analysis has to be solved by the deterministic solving technique. It should be obvious that the designer has to concentrate on few uncertainties.

The use of post-optimal sensitivity analysis lies in the assessment of the key uncertainty, i.e. which of the uncertainties has the greatest (negative) influence on the solution.

However, the black-box model that is assumed in sensitivity analysis techniques does not work well in the context of network design due to the non-analytical (discrete) structure of the output. A small change in an input parameter can lead to a completely different design (which not necessarily has a worse objective), let alone the use of random-driven solving techniques that produce no deterministic solutions at all.

A somewhat different method for sensitivity analysis is to evaluate a solution with respect to possible settings that may differ from the assumptions made in the scenario from which the solution is derived. This approach is based on the observation that the solution of a mathematical model usually leads to a real-word implementation, e.g. a deployment of a network, which has to prove its usability in an (uncertain) environment. It may be important for a designer to know how "well" the solution performs over a range of possible environments that could be generated e.g. by Monte Carlo sampling. This approach is called **(post-optimal) sensitivity testing**.

The main problem of this extension of deterministic techniques lies in the fact that the solution produced cannot take sensitivity demands into account a priori. The optimization process is targeted only at one optimal situation. Scenario analysis relieves this problem only to some degree by producing a (subjective) set of scenarios that have different characteristics with respect to sensitivity. Post-optimal sensitivity analysis therefore does not allow a sufficient propagation of uncertainty.

Post-optimal sensitivity analysis for network design problems has to be replaced by post-optimal sensitivity testing to be meaningful at all. Post-optimal sensitivity testing allows the performance evaluation of different scenarios under different real world settings. To produce solutions that yield good results with respect to sensitivity testing, sensitivity has to be part of the model, e.g. by demanding an over-provisioning of facilities.

Remark 25

The course of action proposed here corresponds to that proposed by the design frame $\mathcal{F}(\mathcal{E})$ in Section 3.2. The key uncertainties and sensitivity requirements are part of the scenario generation process. The results of sensitivity testing are taken into account by the decision

[37] For example, sample each internal factor within $\pm x\%$ deviation of its initial value.

making process, choosing a design which has a good objective and good sensitivity testing results (i.e. a multi-criteria decision).

4.5.4 Extension of Deterministic Solving Techniques II: Robust Optimization

In [MVZ91], Mulvey et al. introduced a framework for robust optimization of large-scale systems which are subject to uncertain input data that allows to include the sensitivity of an optimal solution. An application and extension of this robust optimization technique for electrical power systems can be found in [MZ94]. To give an idea of the underlying concepts, the results of both reports will be used to formulate a robust optimization for network design problems with uncertain demand scenarios.

As shown in Section 4.4.2.3 (Figure 16), the planner faces a set of possible demand alternatives, denoted as **states** $\llcorner\{s_i : i \in \{1 \dots S\}\}\lrcorner$, each of them occurring with a certain probability p_i (a more detailed discussion how this can be accomplished can be found in Section 4.5.6). The goal is to design a communication network which is close to optimal for each demand alternative and has minimal slack capacity.

Robust optimization introduces the following two notions of robustness:

Definition of Concept 47 (Model & Solution Robustness)

*A solution x is denoted as **solution robust** if it is close to optimal for each state s_i:*

$$\forall i : \frac{o(x) - o(x_i^*)}{o(x_i^*)} < D \tag{13}$$

D reflects the relative deviation from the optimum solution, x_i^* is the optimum solution given state s_i.

*A solution is denoted as **model robust**, if it is close to feasible for each state s_i. This allows to accept a solution x even if $\llcorner x \notin \mathcal{A}_i \lrcorner$ for some states i. The "closeness" is usually measured by a feasibility penalty function ρ (see below).*

*In summary, a **robust solution** should be almost optimal for any state and close to feasible for any realization of a state.*

Obviously, both goals are contradictory since making a solution more model robust usually leads to a higher slack capacity and therefore reduces the solution robustness. Robust optimization aims at a tradeoff between model and solution robustness.

Central to the tradeoff is a **feasibility penalty function**

$$\rho(y_1, \dots, y_S),\ \rho : \mathbb{R}^S \to \mathbb{R}, \tag{14}$$

which is used to penalize the violations of a subset of the constraints $\mathcal{Z}$. In the context of network design, y_i could be a measure for the absolute traffic demand that cannot be transported in each demand scenario s_i and the function ρ could be the sum over all y_i.

Remark 26

Technically, this approach is similar to transforming some hard constraints $\llcorner h \in \mathcal{Z} \lrcorner$ into soft constraints, which may be violated in return for a less attractive objective function value and a multi-criteria optimization problem (see Section 2.7.8.2).

It differs by the way the penalties are calculated. Robust optimization allows to take into account arbitrary attributes (e.g. the variance of the solutions).

If the objective function o is assumed to be independent from the state under consideration, the objective function of the robust optimization process becomes:

$$o(x) + w \cdot \rho(y_1, \ldots, y_S) \tag{15}$$

Analogous to the Archimedian approach, w corresponds to the relative importance of the two goals. A smaller w aims at solution robustness, a larger w at model robustness, respectively. Again it is the task of the designer to choose the best weighing for his purpose.

Remark 27

In the case that the objective function depends on the state, i.e. there exists a different $o_i(x_i)$ for each state $\llcorner\{s_i : i \in \{1 \ldots S\}\}\lrcorner$, the multi-criteria objective becomes:

$$\sigma(o_1(x_i), \ldots, o_S(x_S)) + w \cdot \rho(y_1, \ldots, y_S), \qquad \text{with } \sigma : \mathbb{R}^S \to \mathbb{R} \tag{16}$$

4.5.5 Stochastic Programming

Definition of Concept 48 (Stochastic Programming)

Stochastic Programming describes the situation where some of the coefficients in the mathematical program are random variables whose characteristics (i.e. probability distributions) are known[38]*. The objective of stochastic programming is to consider the random effects explicitly in the solution of the model.*

Solutions to stochastic programs base on optimizing expected values, mean-squared errors, or other stochastic attributes.

For example, the *LP* ([↑LP]) defined in equation (17) becomes **stochastic** if some coefficients of the set $\llcorner(A, b, c)\lrcorner$ are random variables. Depending on the stochastic type of minimization (e.g. each constraint in $\llcorner Ax \geq b\lrcorner$ has to be fulfilled with at least $x\%$ probability, which is denoted as **chance-constraint** programming), the stochastic *LP* can be transformed into a deterministic one that is subject to deterministic solvers.

$$\begin{gathered}\text{minimize } o : \mathbb{R}^n \to \mathbb{R} \quad x \to c^\top x \text{ subject to } Ax \geq b \\ c \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ A \in \mathbb{R}^{(m \times n)},\ \llcorner n, m \in \mathbb{N}\lrcorner\end{gathered} \tag{17}$$

Due to the rapidly growing complexity of the problem size, however, this approach becomes prohibitive when dealing with problems of a certain size. Stochastic programming is a very complex topic. The most practical approach again is to consider a probabilistic range of alternatives and solve the deterministic cost minimization for each alternative scenario (i.e. scenario analysis), having all the drawbacks of deterministic solving techniques. An example of stochastic programming in network design can be found in [DMT97].

4.5.6 Stochastic Decision Analysis by Decision Trees

Section 3.5 addressed multi-stage decision problems under complete knowledge by the use of deterministic decision trees. In case of incomplete knowledge, it is necessary to incorporate at

[38] In the case of independent random variables, this means that the PDF for every single random variable is known.

least the key uncertainties (e.g. traffic growth) in the decision making process. This leads to the concept of **probabilistic decision trees**. In the following, the underlying concepts are discussed, extending the example of Section 3.5.2.

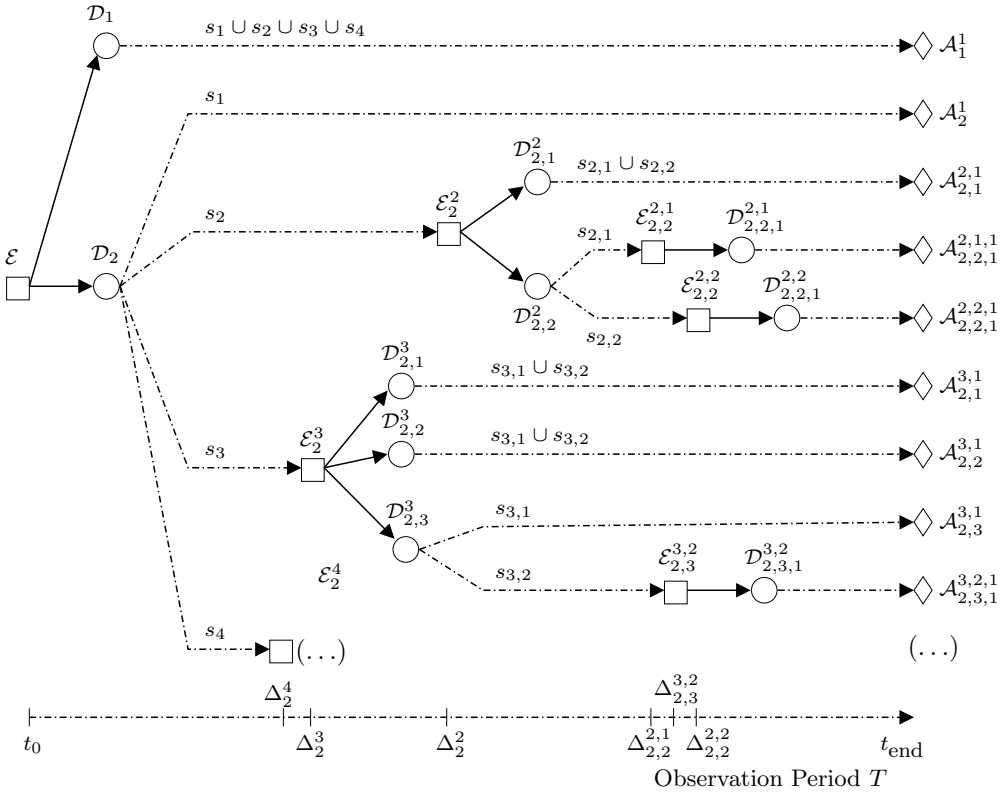

Figure 17: *END* Decision Tree (Probabilistic Case)

Representation of uncertain knowledge in probabilistic decision trees is accomplished by chance nodes that succeed each decision, where each possible outcome, i.e. state $\llcorner s_i$, $i \in \{1 \ldots, S\}_\lrcorner$, is expressed by a different chance branch. To make probabilistic decision trees computationally tractable, an efficient representation of the states is recommended. Efficient representation in this context means: a small number of chance branches (denoted as *fans*), i.e. a concentration to an adequate number of different, expressive states. In each of the states, the uncertainties are resolved by the use of a (deterministic) value (e.g. the expected $E(x \mid s_i)$ value under the condition that the state occurs). For example, the discrete probability distribution of traffic growth in Figure 16 can be expressed by the states:

State s_1: low traffic growth rate g_1 with probability p_1

State s_2: medium traffic growth rate g_2 with probability p_2

State s_3: high traffic growth rate g_3 with probability p_3

State s_4: extreme high traffic growth rate g_4 with probability p_4

and $p_1 + p_2 + p_3 + p_4 = 1$

The modeling of the number and range of the disjoint states is the task of the designer. If a continuous PDF is given, such as the left and middle PDFs in Figure 16, a disjoint interval on the x-axis can be assigned to each state s_i thus defining its range and probability. Like in the deterministic case, a (deterministic) value g_i representing the state s_i has to be determined.

Figure 17 shows a probabilistic decision tree that will serve as an example to point out the basic concepts. Its task is (as it was in the deterministic case) to aid the designer in choosing between the two designs $\mathcal{D}_1$ and $\mathcal{D}_2$ at time t_0. The design $\mathcal{D}_1$ represents a worst case design which can cope with all possible traffic developments over the whole observation period (illustrated in the figure by the union of all possible states over the chance branch). The design $\mathcal{D}_2$ may reach its limits within the observation period (i.e. has to be upgraded) in three of the four possible states, depending on the real traffic growth (i.e. which of the states will come true). From the chance node which follows the decision for the design $\mathcal{D}_2$ therefore stem four chance branches:

- s_1: under the assumption of traffic growth g_1 the design is feasible over the whole observation period
- s_2: under the assumption of traffic growth g_2 the design is feasible until at Δ_2^2 an upgrade is necessary
- s_3: under the assumption of traffic growth g_3 the design is feasible until at Δ_2^3 an upgrade is necessary
- s_4: analogous to s_2 and s_3 but omitted in the figure

The length of an individual chance branch visualizes the amount of time until the next trigger event occurs (the action branches are assumed not to consume observation time). The decision subtree that follows each chance branch is called **conditional subtree**. Each of the decisions that follow a conditional subtree can be made under the knowledge that all states that lie on their path from the root have occurred. The notation which was introduced in Remark 18 allows to identify the sequence of decisions and chances immediately by regarding the indices.

For example, after deploying $\mathcal{D}_2$ the state s_2 occurs with probability p_2. The considerations that follow the action node labeled with $\mathcal{E}_2^2$ can be made under the knowledge that until the observation time Δ_2^2, the traffic has grown with the factor g_2.

Two different approaches exist from there:

- continue the tree with a completely new forecast starting at the point Δ_2^2.
- use the conditional probability distribution (denoted as $P(x \mid s_2)$) that results from the knowledge of state[39] s_2.

Each of the end-nodes $\mathcal{A}_x^y$ is reached with a probability that can be calculated by the product of the probabilities entailed with the states along the (unique) complete path[40] ending in $\mathcal{A}_x^y$.

Returning to the root node, i.e. the choice which of the designs $\mathcal{D}_1$ or $\mathcal{D}_2$ to deploy at time t_0, the following strategies (see Definition 37) could be attractive:

Expected value/utility: Choose the alternative which leads to the best expected objective, i.e. the sum of the objectives weighted by their probability (here in the case of maximization):

$$E(\mathcal{D}_b^a) = \sum_{i \in \{1,\ldots,n\}} P(s_i) \cdot \begin{cases} o(\mathcal{A}) & \text{,if the state } i \text{ ends in the end-node } \mathcal{A} \\ \max_{j \in \{1,\ldots,m\}} \left\{ E(\mathcal{D}_{b,j}^{a,i}) \right\} & \text{,otherwise} \end{cases} \tag{18}$$

[39] In case of a discrete distribution this may result in branches of equal likelihood.

[40] which corresponds to an independence assumption between the states entailed with the chance nodes along the path

MiniMax: minimize maximum objective function value

MaxiMin: maximize minimum objective function value

Flexibility measures: will be introduced in Section 4.5.7.3

However, the size of decision trees may increase very rapidly (with exponential worst case) depending on the level of detail in the modeling of chance nodes and decision nodes and on the number of stages. The designer has to restrict himself to few alternatives and designs, and to one or two independent key uncertainties, otherwise the decision tree becomes computationally intractable.

4.5.7 New Paradigms: Flexibility and Robustness

4.5.7.1 Introduction and Motivation Traditional approaches deal with the uncertainty of meeting demand by forecasting, setting reserve margins based on past variability of demand, and optimizing deterministically against this to produce an "acceptable" network design. This method of dealing with uncertainty is oriented towards *robustness*, i.e. it aims at acceptability of the resulting design with respect to the uncertainties and scenarios considered. Another, even more rigid but commonly used paradigm to safeguard feasibility is to assume a worst case scenario (also denoted as "**extreme value engineering**"). The resulting *worst case design* will be robust against changes in parameters.

However, in environments where the key uncertainties allow a wide range of possibilities, e.g. due to uncertain growth and/or a high degree of variability in traffic patterns, a worst case scenario leads to a massive *over-design* that results in an extremely wasteful if not prohibitive expenditure.

An additional problem arises if reliable traffic data is missing (e.g. due to the lack of measurements) and network traffic itself is extremely hard to quantify (see Remark 23). Optimizing network designs with only a vague notion of the traffic matrix is a quite unreasonable task for a designer:

> "Unpredictability of future traffic patterns and technology changes make it close to impossible and even worthless to look for global optima for static network design problems." ([DH98])

Here the notion of *flexibility engineering* fits in. "Flexibility" is a paradigm when dealing with highly uncertain or even unpredictable environments. The intuitive idea is to choose a design which allows to adjust to changing requirements or disruptive events on short notice. This freedom usually has to be paid for.

The necessity to realize flexibility in network design is emphasized by many authors, e.g. [GP87, Doo94, TF97, DH98], however, appropriate formalisms and practical implementations are missing so far. In [Ku95], the application of flexibility in the context of electricity capacity planning is discussed. The formalism presented here is motivated by this work.

The robustness and flexibility paradigms interact. It will be made clear in the next sections that robustness and flexibility are no contradiction, but complement each other.

4.5.7.2 Introduction of the Core Concepts Rather than modeling uncertainty more comprehensively or accurately, the ability to react, such as having different options available, may be a better, i.e. more cost efficient concept to cope with an uncertain environment. Therefore, flexibility is a demand as a response to highly uncertain and unpredictable environments.

Definition of Concept 49 (Flexibility)
Flexibility is a generic term that describes the ability to change or react when necessary.

Attributes that are potential symptoms (depending on the concrete settings) of flexibility are: adaptability, scalability, lack of commitment, and reactability. In the case of network design, candidates for introducing more flexibility in a network design are:

- the choice of a network technology that can effectively react to traffic variations (e.g. ATM with its PVCs/SVCs)
- the commitment to a certain network technology and the entailed ability to switch to other technologies (e.g. copper vs. fiber)
- the scalability of the network technology
- the choice of software technology, e.g. routing protocols that can cope with traffic variability
- a network configuration $\mathcal{C}$ that can be changed easily
- (specific) slack capacity in the design that can cope with traffic variability
- the possibility to lease overflow facilities, e.g. dial-up lines, to cope with (rare) traffic peaks
- short lead times[41] for activating additional facilities, e.g. by contracts that allow inexpensive increase of bandwidth on short notice

For illustration, two aspects are discussed in greater detail:

- The decision to buy or lease network facilities is not only an accounting issue but also a strategic one, affecting the way a corporation can deal with future uncertainty [Ku95]. Strategically speaking, buying ties the designer to this specific technology for the life of the facility whereas leasing enables him to switch to new technologies when necessary. Ownership and leasing differ by the degree of commitment or confinement to a specific technology. By leasing, the corporation can terminate its commitment at any time and limit its technological commitment. In practice, a corporation may choose to own some of its facilities (e.g. fixed line capacity) and lease additional facilities when necessary (e.g. use dial-up lines or buy capacity on the spot market). This arrangement can be seen as robustness to handle expected demand and flexibility to handle the unexpected. The lead time is translated into how soon the corporation can lease or return the extra facilities required. The shorter the lead time, the more flexible and attractive is the option of leasing.
- The choice between copper and fiber cables when equipping a new building is a further example of adding flexibility to a design, albeit at higher costs. It makes a difference in costs of introducing fiber (together with the necessary optical equipment) instead of copper of orders of magnitude. However, if the situation arises that a successor network technology (which naturally works at higher speed) needs fiber as transport medium, the initial costs are soon amortized. The choice of the more flexible option is commonly denoted as **enabler**, since it gives the designer more flexibility, if the situation should arise (i.e. at a certain trigger event).

[41] The **lead time** t_{lead} is defined as the time needed to put a change into action.

The need for flexibility directly depends on the stability of the environment. In a stable environment, where there is low uncertainty the designer can concentrate on optimizing his design and there is no need for flexibility. In a rapidly changing environment with many unforeseen (external) uncertainties, the ability to react may be a much more desirable design goal than optimality. In such a scenario flexibility is of "great value". In this sense flexibility is "a hedge against uncertainty". The following design principle is derived from [MB90]:

Definition of Concept 50 (Flexibility Design Principle [MB90])
"Under great uncertainty, flexibility is preferred to optimality as a decision criterion. Flexibility is also preferred over optimality when the decision maker does not have full confidence in the model."

Flexibility is also a potential to change, reflecting future costs that have not occurred yet. Due to the provision of flexibility, that is, the capability to be flexible, present costs may incur. The ability to accommodate future changes is reflected in the opportunity costs for admitting under-utilization. The amount of flexibility built in a design depends on the costs and gains of flexibility, hence implying that flexibility is not a free good.

In contrast to flexibility, a robust network is designed to avoid changes as much as possible, i.e. the design itself deals with changes without intervention of the designer. However, a network design which provides a solution to all possibilities will, in all likelihood, be a very wasteful expenditure. This results in the following definition:

Definition of Concept 51 (Robustness)
Robustness, sometimes also called "passive flexibility", means the absence of a need to change or react.

As mentioned in the introduction, one way to achieve robustness consists of assuming a **worst case scenario** which generally leads to a massive over-provisioning of facilities. Hence, robustness communicates the notions of predictability and stability. Like flexibility, robustness is no free good. Robust designs incorporate all possible future changes reducing the ability to optimize the design effectively.

4.5.7.3 Modeling Flexibility & Robustness There is no agreed upon concept or formalism for incorporating flexibility. Flexibility is context sensitive, it depends heavily on the nature of possible decisions which usually are problem-dependent. Another problem is that some concepts discussed in the previous section (such as bindingness to a contract) evade proper formalization. Therefore the value of flexibility is difficult, if not impossible, to ascertain. Robustness, on the other hand, is easier to formalize but due to limited resources is quite unrealistic to achieve in highly uncertain environments. The problem lies in how to measure flexibility and robustness and to find a tradeoff between both paradigms. To assess this, there is a need for a method of measuring flexibility and robustness and trading off their costs and benefits. Examples for possible measures are:

Simulation methods: Similar to sensitivity testing, simulation based on Monte Carlo Sampling can be used for evaluation of flexibility and robustness in particular situations.

Number of open choices: Given an initial state in a decision tree, flexibility can be measured by the number of available choices. This measure has the advantage of depending on minimal information about the uncertainty under observation.

Number of chance branches: Given an initial state in a decision tree, robustness can be measured by the size of chance branches. A robust design should need only few decisions.

Expected value: If the uncertainty can be quantified, the expected value (e.g. in terms of discounted costs) of different choices is a good measure.

Relative flexibility benefit [HHB94]: This benefit measures the difference between considering uncertainty as opposed to treating it as 100% probable.

Remark 28

Simulation and expected value seem to be preferable measures. However, in most of the situations where especially flexibility is desirable, detailed information (e.g. in form of a PDF) is not available.

An example of how flexibility and robustness can interact is shown in Figure 18. Given a distribution of the overall expected traffic, the idea is to find a tradeoff between flexibility that handles the tail of the traffic distribution, and employing a (still cost efficient) robust design which deals with the most probable situations. Since robustness and flexibility are both no free goods, a cutting point having minimum overall costs has to be sought.

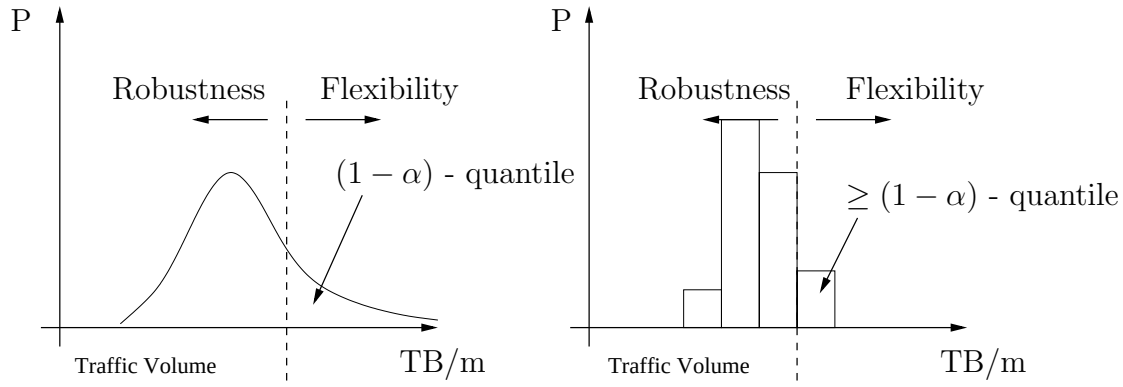

Figure 18: Continuous and Discrete Flexibility vs. Robustness Separation

A further example of a weighing between robustness and flexibility was already proposed in Section 2.7.1.5 by cutting the highest values of the distribution of traffic variations and let rerouting, respectively congestion control, take over the handling of overload.

4.5.7.4 Discussion The words "flexibility" and "robustness" often appear in the same articles and are even used inter-changeably as if they would have the same meaning. Flexibility is the ability to react or change, and robustness is the lack of a need to react or change. Flexibility and robustness are not opposite or the same, but merely two sides of a coin, corresponding to two ways of responding to uncertainty. In spite of its popularity, flexibility has not received the same formal recognition as optimality. The reasons for this fact are the problems of finding and formalizing appropriate measures. This has to be subject of further research.

However, especially in highly uncertain environments the network designer should keep both paradigms in mind when deciding in favor of a certain network design. Simple (even subjective) assessments of the flexibility of a design could be one of the criteria in a multi-criteria decision process. The designer could e.g. vote for the most flexible solution out of the s best scenarios in a design frame $\mathcal{F}$.

There are three potential intervention points in the planning process where flexibility and robustness can be taken into account:

1. The scenario generation process binds all free variables, thus offering the possibility to incorporate flexibility and robustness in the model. For instance, selecting hardware, software, and tariffs that support flexibility or to over-dimension traffic matrices to cope with higher traffic variations.
2. The mathematical model of a *NDP* can cope with uncertainty and robustness by incorporating these concepts into the objective or constraints. The optimization algorithms applied to the model may offer the possibility to prefer flexible and robust designs in the optimization process.
3. The decision making process can choose a design that represents the best balance between costs and flexibility & robustness.

4.5.8 Review and Assessment

Finally, the author wants to give a short review and assessment of each strategy:

Post-optimal Sensitivity Analysis: Uncertainties are incorporated in the model through scenario analysis. The generated (deterministic) scenarios are solved by mathematical programming, and the resulting solutions are rated by sensitivity analysis and sensitivity testing. Uncertainty is treated in the same way as variability.

Uncertainties cannot be sufficiently incorporated in the optimization process, therefore the resulting solutions cannot guarantee any optimal characteristics in dealing with the uncertainties in the environment. Another inherent uncertainty lies in the preferences and capabilities of the designer who influences the solution space by the quality of the scenarios that are permitted to enter the optimization process. All these efforts do not sufficiently address the resolution of uncertainties, they merely try to restrict and measure their influence.

- ⊕ relies on well-known (and efficient) deterministic solving techniques
- ⊕ can be easily added to existing design algorithms
- ⊖ due to the a posteriori uncertainty propagation, constraints on uncertainty cannot be incorporated into the model
- ⊖ makes use of (inefficient) sampling-based methods and therefore becomes computationally intractable for growing problem sizes

Robust Optimization: Uncertainties can be included in the model by the use of different scenarios and a penalty term in the objective function. The resulting multi-criteria optimization problem is able to consider uncertainties directly. Even if no hard bounds with respect to uncertainty can be safeguarded by this technique, it is a good compromise between efficiency and uncertainty incorporation. A potential weakness lies in the preferences and capabilities of the designer who influences the solution space by the quality of the selected scenarios that are regarded in the optimization process.

- ⊕ relies on well-known and efficient deterministic solving techniques
- ⊕ uncertainties can be considered a priori in the model
- ⊖ weighing between objectives is task of the designer and cannot be automated
- ⊖ incorporation of uncertainty relies on a few coarse scenarios which have to be chosen by the designer

Stochastic Programming: The approach of incorporating uncertainty parameters directly into the model seems to be an elegant solution technique. However, with no restricting demand, e.g. on the PDFs of the input uncertainties, the propagation of the uncertainties through the model has to fall back on sampling based methods, where a deterministic optimization run is necessary for each sample. The more detailed the output has to be, the more model runs are necessary. Therefore, stochastic techniques result in an expanded and more structured uncertainty analysis that becomes computationally intractable with growing problem size.

- ⊕ good, a priori incorporation of uncertainty
- ⊖ usually has to fall back on sampling based, deterministic optimization techniques and therefore becomes computationally intractable for growing problem sizes

Probabilistic Decision Analysis: Probabilistic decision trees address the input uncertainties by subdividing the problem into stages and branching at the chance nodes. As a technique taken from the field of decision analysis, the focus is rather on a sequence of decisions than on the optimization itself. Therefore, the level of detail necessary for a thorough optimization is missing. The strength of probabilistic decision analysis lies in the fact that it incorporates uncertainties to a much higher degree than the previous approaches, thus allowing a designer to contribute his personal preferences and risk attitudes in the design. Besides, the structure of decision trees has a much higher resemblance to multi-staged problems such as *ENDP*s than any other technique.

- ⊕ explicit consideration of evolutionary design aspects
- ⊕ uncertainty can be incorporated in almost any level of detail; can cover many aspects of uncertainty
- ⊕⊖ dependence on the optimization algorithm at any stage, no a priori assessment may be possible
- ⊖ becomes computationally intractable very rapidly due to the multi-stages
- ⊖ requires independence assumptions
- ⊖ complete optimization is necessary in every stage
- ⊖ if computational tractability has to be maintained it is unable to capture many details

Flexibility & Robustness: Both concepts as introduced here are less formal and no mathematical technique in the strict sense, but rather an interesting design philosophy. Especially if the designer misses vital data, flexibility engineering can offer a way out of the dilemma.

Uncertainties are not considered directly, but are "resolved online" by providing a flexible but robust design which is able to react to changes on short notice. The goal of the optimization shifts from optimization in terms of costs towards the new goal "flexibility". The main problem entailed with this approach is that flexibility as a design goal often evades a proper formalization.

- ⊕ allows reasonable network design, even in highly uncertain environments
- ⊕ can cover many aspects of uncertainty
- ⊖ relies on problem-specific, hard to formalize design goals and therefore optimization in strict mathematical sense is hard to achieve
- ⊖ hard to automate, relies on subjective assessments of the designer

This short review clearly shows that there is no known all-embracing method to cope with uncertainties, some approaches cover some aspects but none of them covers all. Models that have to deal with uncertainties are generally less tractable than deterministic models. To reduce the complexity, the key uncertainties must be reduced to few factors and less important uncertainties should be treated as deterministic values, thus increasing efficiency with only minimal loss of accuracy.

5 An Object-Oriented Software Framework for Network Design Problems

5.1 Motivation

The relation between computer science and network planning is manyfold. Voice telecommunication network planning was one of the first applications for which the computational power (supported by advances in mathematical optimization) was and still is used. The savings which were achieved by the use of new (non-hierarchical) routing techniques led to new types of network planning algorithms. These algorithms required large computational power which exceeded the possibilities of manual calculations by far. Moreover, the explosion of communication service demands (the Internet) causes a vital interest of computer science in telecommunication network design.

However, due to its heritage, software engineering aspects have played only a minor role in network design. Practical engineers have traditionally been fond of imperative programming languages, such as FORTRAN and C. The trend in computer communication is towards object-oriented software engineering with its ability to cope with complexity even for large problem sizes and to reuse software. Some projects have already shown the power of object-oriented approaches in communications, for example in the field of network-management (OSI network management) or protocol design (e.g. [Böc97]). But the network design task itself lacks tools supporting the object-oriented programming paradigm. It seems that especially in telecommunication the application of object-oriented software design is promising. Entities which occur in network design problems can be transformed into object-oriented data-types in a very natural way. For example, Jackson at al. showed in [JGJ97] that the reuse of telecommunication software by AT&T was between 40% and 92%.

In the following sections, the basic ideas of a software tool called *FooNet*[42] is presented which is the result of submitting the telecommunication network planning process to an object-oriented analysis&design (OOA&OOD). *FooNet* is an application framework that relieves the designer from "reinventing" the parts of the software design that are common to all network design problems. Object-orientation can achieve this without limiting the generality of the design process itself by making restrictions that the designer cannot overrule. *FooNet* provides a system of base classes from which the application specific classes have to be derived. All problem-independent parts are invisible to the user such that he can concentrate on the problem-specific algorithms and data-structures. The task of "inventing" an appropriate network design algorithm and/or representation cannot be completely relieved, since this is problem-depending. However, all other activities in network design (e.g. persisting, displaying, editing, forecasting, loading, performance evaluation, sensitivity testing) are excluded from the design process.

The characteristics of *FooNet* in contrast to other planning tools are its consequent object-oriented design and the support of reuse techniques on different levels of abstraction. *FooNet* comes with a (still growing) library of algorithms used in network planning. Additionally, it makes use of the capabilities of XML[43] as a data exchange format. XML allows to overcome the lack of agreed upon standards for data exchange between applications.

In the following, an overview of the underlying ideas and basic abstractions will be given. A more detailed reference of *FooNet* can be found in [Fis99], the software and programming manuals are available at [Fis00].

[42] Framework for the object-oriented Network Design

[43] With the success of Internet technologies, the W3C consortium has introduced a technology called Extensible Markup Language (**XML**) - that allows the exchange of almost arbitrary information between different applications.

5.2 Introduction to Object-Oriented Software Design Principles

5.2.1 Overview

This section gives a brief overview of recent advances in object-oriented software design. It is assumed that the reader has some experience with object-orientation, otherwise, the author recommends [Boo91, Cop92, Sto97] for a detailed and application-oriented introduction.

Some important terms and concepts in object-orientation are introduced as follows:

- **Object-Oriented Analysis (OOA)**: Object-oriented analysis is a method of analysis that examines the requirements on the software from the perspective of classes and objects found in the problem under consideration.
- **Object-Oriented Design (OOD)**: Object-oriented design is a method of design encompassing the process of object-oriented decomposition and notation (here in the language of UML) in order to depict logical and physical as well as static and dynamic properties of the problem under design. The most important principles in object-oriented design are abstraction, encapsulation, modularization and hierarchy.
- **Object-oriented Programming (OOP)**: Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class.
- **Inheritance**: Inheritance is a hierarchical relation among classes, in which one class shares the structure or behavior defined in one (single inheritance) or more (multiple inheritance) other classes. Inheritance defines a "is-a" hierarchy among classes in which a subclass inherits from one or more generalized super-classes. A subclass typically specializes its superclass by augmenting or redefining the existing structure or behavior.
- **Polymorphism**: Polymorphism is a concept in type theory wherein a name may denote instances of many different classes (usually related to some common superclass in C++). **Dynamic Binding** is a consequence of polymorphism, which implies that sending the same message to different objects could stimulate different behavior. Technically, this is realized by the use of **virtual functions**.
- **Interface**: An abstract base class which provides a set of virtual functions is called an interface.
- **Persistence**: Persistence is the property of an object through which it can transcend time (i.e. the object continues to exist after its creator ceases to exist) and/or space (i.e. the object's location moves from the address space in which it was created). Persistence in *FooNet* works with character streams and relies on two complementing methods: **serialize** (write object in a stream) and **de-serialize** (read object from a stream).
- **Class Library**: A class library is a collection of reusable classes that rely on object-oriented design paradigms such as hierarchy and polymorphism. Usually, class libraries provide support in solving problems belonging to a specific problem category.

5.2.2 Unified Modeling

The object-oriented software design paradigm does not dictate a methodology of how the classes and the relationships between them are derived from the requirements. This part has to be performed by the designer[44] supported by a detailed object-oriented analysis.

[44] and therefore is often considered an art

However, there are several process models (for an overview see e.g. [NS99]) which guide the designer through this task. The process model that is used here is the **Unified Process** proposed in [Jac99], which is a use-case-driven and incremental approach and consists mainly of the five stages illustrated in Figure 19.

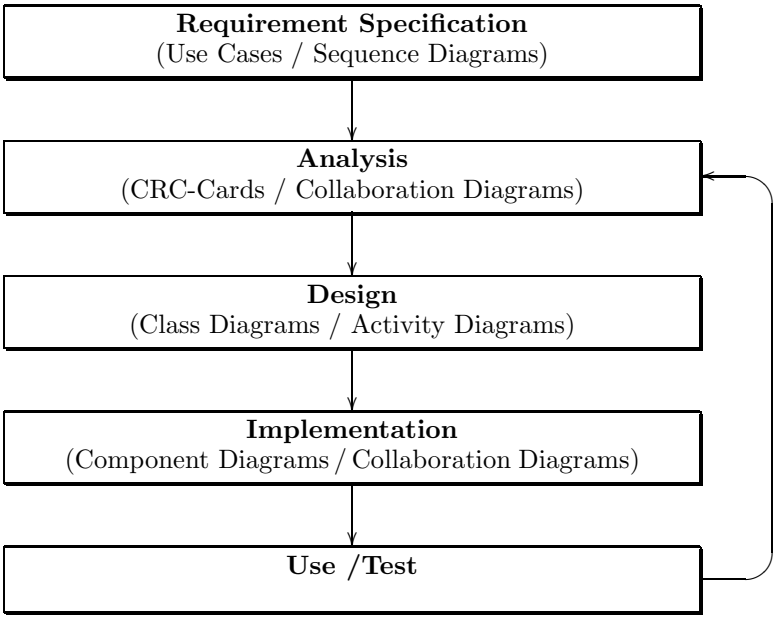

Figure 19: Object-Oriented Software Design Process

Connected with the Unified Process is **UML**, the Unified Modeling Language, that supports the process model by providing a set of easy to read but expressive diagrams. UML is a symbolic language for specifying, constructing and documenting (the semantics of) software systems. It is standardized by the Object Management Group (OMG) in [OMG97] and has become the "Esperanto" of object-oriented design. The terms in parentheses beneath each stage of the diagram in Figure 19 denote examples of which part of the UML notation may be appropriate when working in this stage. However, UML does not dictate a particular process, it is rather a "blueprint" for software design. A short introduction to UML that is sufficient to understand the diagrams in this thesis is presented in Appendix D.

5.2.3 Software Reuse

5.2.3.1 Motivation The implementation of complex software systems remains resource expensive and error prone. Already the design of a medium sized computer program is a nontrivial task. Much of the costs stem from the rediscovery and reinvention of core concepts and components.

The concept of software reuse originates from the observation that a software designer should concentrate on the peculiarities of his task instead of solving problems that have already been solved many times before. Software reuse is commonly defined as "*the systematic development of reusable components and the systematic reuse of these components as building blocks to create new software systems*".

A reusable component may be code, denoted as **code reuse**, but the bigger benefits of reuse come from a broader and higher-level view of what can be reused: software specifications, abstractions, design patterns and frameworks. This is commonly denoted as **design reuse**.

FooNet provides an application framework (see Section 5.2.3.3) for network design problems and as such provides **design reuse**, but builds itself on top of two other reuse techniques:

- **Idioms**: Idioms allow **code reuse** by providing higher level datatypes, such as lists, queues, and graphs (see e.g. [Cop92]). The C++ language provides a number of idioms, which are standardized in the Standard Template Library (STL, [ISO97]). Additionally, the idioms provided by the Graph Template Library (GTL, [Him00]) are used. Sometimes this kind of code reuse is denoted as **generic programming**.
- **Design Patterns**: Design Patterns are a (very successful) **design reuse** technique. A short description is given in Section 5.2.3.2.

From this perspective, *FooNet* has a three level reuse hierarchy: Classes, objects and idioms on the lowest level of abstraction, design patterns on a medium level and frameworks on the highest level.

5.2.3.2 Design Patterns A design pattern [GHJV95] describes a (often reoccurring) problem, the core of a simple and elegant solution together with the context in which the solution works, and its costs and benefits. Design patterns serve as the micro-architectural elements of frameworks, but due to their abstractness, they cannot be expressed as classes or lines of code.

The following list gives a short overview of all design patterns[45] that have been used in *FooNet* together with examples of where the design patterns occur:

- **Letter-Envelope** (also known as **Handle-Body** or **Bridge**)
  A *letter-envelope*-pattern decouples an abstraction, denoted as envelope, from its implementation, denoted as letter. The advantage of this design pattern lies in the fact that the letter object can vary independently from the envelope which allows a greater flexibility of usage.

  A commonly used example of the letter-envelope design pattern is a reference-counted class. The node-envelope of *FooNet* administrates a pointer and a reference-counter to the node-letter. The actual node-letter object is duplicated only if necessary. This significantly reduces resources.
- **Factory**
  A *factory* defines an interface for creating an object, but lets subclasses decide which class to instantiate. Usually a factory provides a method `produce()` which creates an object of a known base class. Factories prevent the inclusion of user-specific code in the class design.

  FooNet provides a *factory* pattern for network facilities. To produce a certain kind of a network facility (for example an IP-router), the user sends a message to that factory with a set of requirements (for example having at least a throughput of 100 Megabit per second) and the factory returns the least expensive network facility found (for example a Cisco IP Router having 1000 Megabit per second throughput) which meets the requirements. The user can add more manufacturers dynamically by deriving a new sub-factory. The selection process and the vendor-specific data is shadowed by the design pattern.
- **Strategy**
  A *strategy* defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategies allow to formulate a skeleton of an application which is independent of a concrete realization.

[45] For a more detailed description and sample implementations please refer to [GHJV95].

The (static) routing interface in *FooNet* is a typical example of a *strategy*. For example, testing the network for overload needs the routing functionality, but whether routing is performed by an OSPF, PNNI or some other kind of routing scheme is irrelevant.

- **Visitor**
  A *visitor* represents an operation to be performed on the "elements of an object structure" (see example below). A visitor allows to define a new operation without changing the classes of the elements on which the visitor design pattern operates.

  The topology of a network layer forms an object structure that consists of topological (node and edge) elements. *FooNet* provides a *visitor* pattern for topological elements. For example, if a user wants to know the average load on the edges of that layer, he could derive this functionality from the visitor interface and simply call the visit-all-edges method.

- **Composite**
  A *composite* is used to represent "part-whole" hierarchies of objects, when differences between compositions and individual objects can be ignored.

  The logical hierarchy of a network is a typical example of a *composite* pattern. The two types of nodes (end-nodes and tier-nodes) form a tree structure, but e.g. when calculating the costs of a network, there is no difference between tier-nodes and end-nodes.

- **Warper** (also known as **Decorator**)
  A *warper* dynamically attaches additional responsibilities and functionalities to an object. It provides a flexible alternative to sub-classing.

  Warpers are frequently used for attaching visualizing capabilities to an object. So does the GraphWin-warper of a layer object in *FooNet*. The GraphWin-warper can be transparently used wherever a layer object is expected, but observes any message sent to that layer-object and updates its visualization if necessary.

- **Builder**
  A *builder* separates the construction of a complex object from its representation such that the same construction process can create different representations.

  The XML-Factory class calls a builder design-pattern which is provided by an external XML-parser. This parser takes an XML document and builds a parse tree from which the internal representation is derived.

- **Prototype**
  A *prototype* specifies the kind of object to create by using a prototypical instance and creates a new object by copying this prototype.

  This design pattern is used wherever a pointer to a base class has to be cloned. Technically, every class supporting the *prototype* pattern provides a method `clone()` which guarantees that a perfect copy is made.

- **Command** and **Observer**
  A *command* encapsulates a request as an object, thereby parameterizing the clients of the *command* pattern with different requests. An *observer* defines a one-to-many dependency between objects such that when one object changes state, all its dependents are notified and updated automatically.

  Both design patterns are typically used in graphical user interfaces. A menu-bar entry could be realized as a command. An observer can guarantee that different views of a single object are consistent. Since both patterns are not used in the base framework of *FooNet*, they are not discussed in greater detail.

- **Virtual Constructor**
  A *virtual constructor* allows to build an object of known abstract type but unknown concrete type, which is important when de-serializing objects.

  A builder that sequentially reads an object out of a stream usually knows the base type of the next expected object, but not its concrete type. For example, the XML-Factory knows that the next object has to be of the type "facility", but it does not know which concrete type this facility has, i.e. its subclass. The *virtual constructor* pattern is a possibility to solve this problem.

5.2.3.3 Frameworks Object-oriented application frameworks are a promising technology for reusing proven software designs (**design reuse**) and implementations (**code reuse**) in order to reduce the costs and improve the quality of software. A framework is defined as follows [Joh97, FSJ99]:

Definition of Concept 52 (Framework)
A framework is a reusable design of all or parts of a system that is represented by a set of abstract classes and the way their instances interact. Frameworks aim at solving a family of similar problems.

The purpose of a framework is to provide the skeleton of an application that can be customized by an application developer.

Frameworks differ from class libraries by their additional reuse of high-level design, since frameworks do not only define classes but also a **model for interaction** between them. A framework is therefore a "semi-complete" application (by the use of **inversion of control**[46]) that can be specialized to produce custom applications. Frameworks enhance modularity by encapsulating volatile implementation details behind stable interfaces. Extensibility is supported by providing explicit hook methods that allow applications to extend the interface.

There exist two different kinds of frameworks:

- **White-box** frameworks rely heavily on object-oriented language features like inheritance and dynamic binding in order to enhance extensibility. To use white-box frameworks, intimate knowledge of their internal structure is needed. *FooNet* is designed to be a white-box framework.
- **Black-box** frameworks support extensibility by defining interfaces for components that can be plugged into the framework via object composition. The functionality of black box frameworks is based on design patterns, such as the strategy and command pattern. Black-box frameworks are less flexible than white-box ones, but usually easier to use.

Several properties are commonly demanded for frameworks:

- Completeness: The framework should provide all necessary functionality.
- Efficiency: The framework should provide efficient implementations of the relevant, time-critical parts.
- Flexibility (reusability): The framework should be applicable in more than one context.

[46] Sometimes also denoted as the Hollywood principle:"don't call us, we call you".

- Ease of use: The user of the framework should only be responsible for the part of the implementation that is software specific.
- Extensibility: the framework should have the ability to grow with future requirements.
- Portability: The framework should not be restricted to a specific hard- or software.

5.3 A Framework for Object-Oriented Network Design

5.3.1 Overview

FooNet is a white-box application framework that is designed to significantly reduce the development effort of network design applications. At the moment, its focus is on telecommunication network design, but it could be easily expanded to cope with other network design problems (such as road networks or gas pipelines) as well.

The design of *FooNet* consists of two parts:

- A core framework, which is the result of an object-oriented analysis & design process. It represents the abstractions, interfaces and interaction models of network design problems.
- An extended framework that builds on top of the core framework and enriches it by providing additional functionality and applications. It is expected to grow very fast in the future, providing a library of algorithms helpful for telecommunication network design.

FooNet provides a system of base classes from which the application specific subclasses can be derived. All problem independent parts are invisible to the user, such that he can concentrate on the problem-specific algorithms and data-structures. As a white-box framework, *FooNet* relies heavily on inheritance from base-classes and overriding pre-defined hook-methods, i.e. the user derives his specializations from a set of appropriately designed (interface) classes. Virtual functions provide default implementations that are often useful, but can be overridden, if required. Sometimes such virtual functions do nothing at all, but they allow the user to add some functionality. The task of "inventing" an appropriate network algorithm and/or representation cannot be completely taken off the designer, since this is problem-dependent. However, all other activities in network design (e.g. persistence, displaying, editing, forecasting, loading, performance evaluation, sensitivity testing) are managed by the framework.

The next sections introduce the main abstractions of *FooNet* in greater detail. References to the classes or methods in the implementation are in `teletype` font. References to design patterns are in *italics* font.

5.3.2 The Core Design Classes

The UML static class diagram of the core framework is shown in Figure 20. The purpose of these class diagrams is to clarify the model of interaction between the classes.

In the following sections, the main abstractions are introduced, but the details of the implementation are left to the program documentation that is included in the *FooNet* distribution [Fis00].

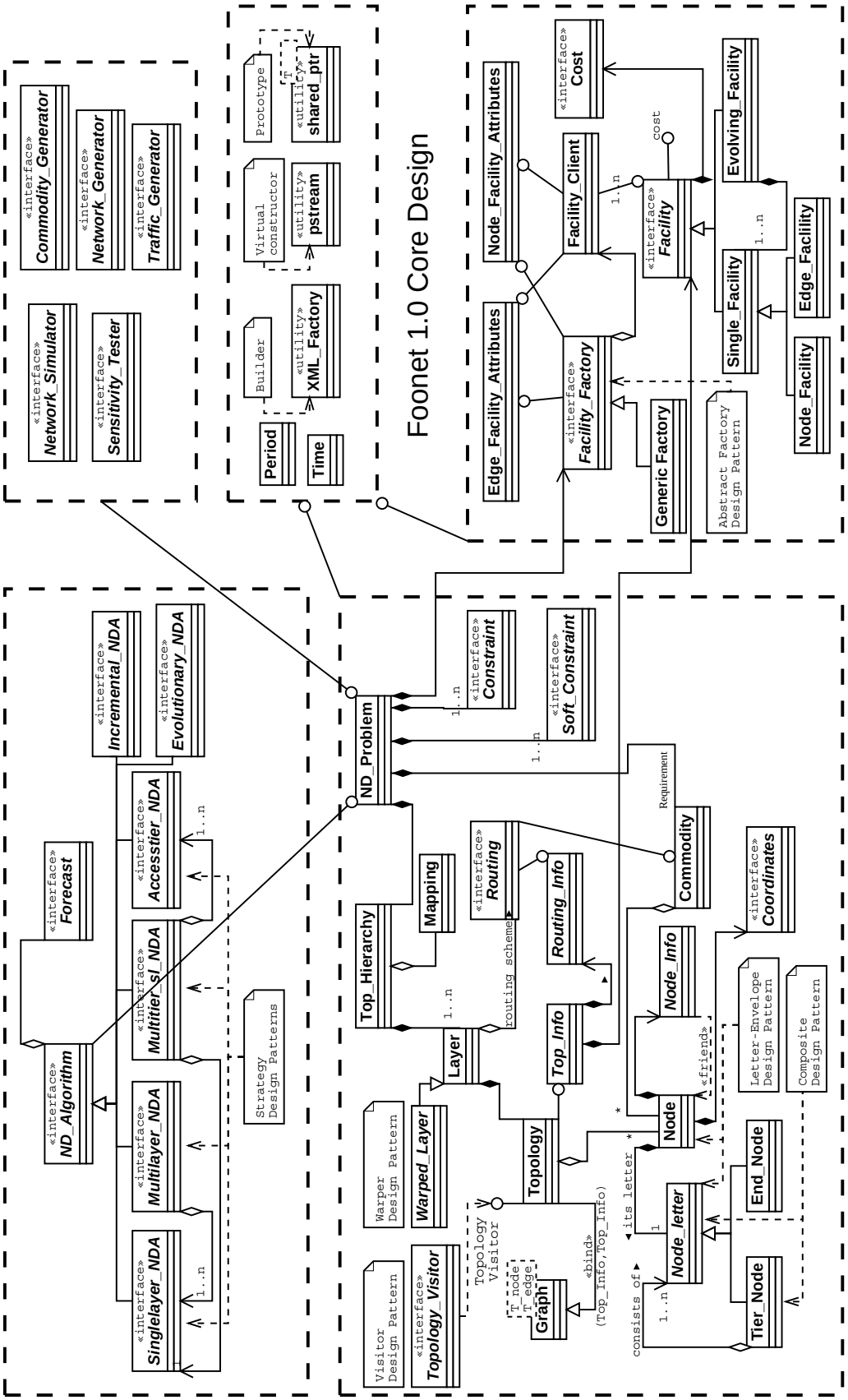

Figure 20: Static UML Core Class Diagram

5.3.2.1 Network Nodes The class `C_node` is an abstraction of a network node. A node is a location (given as `C_coordinate`) of an arbitrary source or destination of a traffic requirement. A node may represent a single workstation or even a whole corporation. Nodes are elements of the network topology (see Section 5.3.2.6) and are embedded in the logical hierarchy of the network by their tier-level (see Definition 5). The nodes on the lowest hierarchical level are denoted as **end-nodes**, all others are called **component- nodes**. Definition 5 proposes a two-tier hierarchy in which the component-nodes on the higher layer are denoted as backbone-nodes.

Nodes are implemented by `C_node_letter` objects using the *composite* design pattern. `C_node` is an envelope according to the *letter-envelope* design pattern that hides a (reference counted) letter object `C_node_letter`.

Each node contains a `C_node_info` object that allows the user to store additional information by deriving his own information class from it.

5.3.2.2 Commodities The class `C_commodity` represents the communication demands or requirements in terms of bits per second between a set of network nodes, i.e. the **traffic matrix**. A single commodity is identified by a pair of network nodes, the source and the destination. The class `C_commodity` provides methods for the computation of the traffic demands on each tier level.

5.3.2.3 Facilities & Costs The class `C_facility` is an interface that represents all (edge and node) facilities of a network (corresponding to Definition 6), whereby each facility $f \in F$ belongs to a single network layer (see Section 5.3.2.5). Each facility has a reliability and a capacity.

The most important attribute common to all facilities is the cost interface that allows to calculate the expenses necessary during their lifetimes. These costs are stored in a `C_cost` object and calculated according to the discounted cash flow formula (see equation (48) and (49) in Appendix C) using the setup-, reoccurring- and termination (respectively upgrade) costs. The cost interface is intended to cope with usage-dependent costs in the next software release.

The specializations of `C_facility` include single facilities, `C_single_facility`, on the one hand and evolutionary facilities, `C_evol_facility`, on the other hand. Single facilities divide into edge, `C_edge_facility`, and node, `C_node_facility`, facilities. Evolutionary facilities represent a series of succeeding facilities $\llcorner f \in F \cup \oslash \lrcorner$ over the observation period[47].

5.3.2.4 Facility Factory & Facility Client The class `C_facilityclient` represents all available facilities of a single network layer. Internally, it stores a list of `C_facilityfactory` classes. Instances of `C_facilityfactory` subclasses may represent a single vendor, product-line, public carrier service etc. To obtain a list of appropriate facilities the user must specify the characteristics by (node or edge) attributes, which are based on `C_ef_attribute` for edge-facilities and `C_nf_attribute` for node-facilities, respectively. `C_facilityclient` is a realization of the *factory* design pattern.

`C_facilityclient` enables the designer to handle different facility factories, but it burdens him with the task of choosing a facility out of a list of appropriate ones, since one facility may have lower setup costs while the other has lower reoccurring costs.

Sometimes vendors offer special "upgrade fees" when switching from one facility to another. This can be modeled by overriding the upgrade-costs method (`C_facilityfactory::upgrade()`). If no appropriate facility is found, an empty object is returned.

[47] For example, the facility f_1 is known to be replaced by facility f_2 at planning time Δ.

An instantiation of a facility factory is given by the class `C_generic_factory` which implements the economy of scale model introduced in Appendix C.4, Equation (54). Additionally, a list of discrete values for the available capacities can be specified. When producing a facility, the cheapest facility that meets the given attributes is returned.

5.3.2.5 Network Layer and Topological Hierarchy The class `C_layer` is a collection of all information associated with a single network layer, e.g. SDH, ATM or IP (see Figure 1). A layer object contains information about the available facilities (in a `C_facilityclient` object), the used routing scheme (in a `C_routing` object) and its topology (in a `C_topology` object).

The network layers are organized according to the **topological hierarchy** of the network (`C_topological_hierarchy`) (see Figure 1). The mapping between peer layers can be supported by a mapping object (`C_mapping`)[48]. It provides routing functionality between peer layers and allows e.g. to query which facilities of a lower layer (e.g. an SDH-path) are used by a facility on a higher layer (e.g. an ATM-VP) and vice versa.

Additional functionality can be attached to a layer by the class `C_warped_layer` which is an instantiation of the *warper* design pattern.

5.3.2.6 Topology The class `C_topology` represents the topology of a single network layer, i.e. the layout of its nodes and edges (see Definition 6). `C_topology` is implemented by a parameterized `C_graph` object. Every topological element, i.e. all nodes and edges, contains a `C_topology_info` object. Each topological node-element is associated with a `C_node` object and each topological edge element is associated with two `C_node` objects, the source and destination node.

`C_topology_info` is a base class for (node and edge) information associated with every topological element. It contains a facility (`C_facility`) object, a routing information (`C_routing_info`) object, and a variable to store the current load in terms of bits per second. It is designed to be overridden if additional information is required.

`C_topology_visitor` represents an abstract interface of a *visitor* design pattern for topological elements. For example a `C_topology_visitor` object is used to calculate the costs of the topology by visiting all topological elements and summing up the associated facility costs.

5.3.2.7 Routing The class `C_routing` is an interface for a static routing (respectively loading) scheme (see Definition 17). The main functionality is in the `C_routing::load()` method, which loads a commodity on the topology according to the routing scheme by setting the `load` variable in each topological element.

To support all possible routing schemes, each topological element stores a `C_routing_info` object, which should be overridden to support the needs of the routing scheme. In its basic implementation `C_routing_info` contains no information.

5.3.2.8 Network Design Problems, Forecast & Constraints The class `C_ndp` represents both a network design problem and its solution. It serves as the parameter to the network design algorithms (`C_nda`).

Remark 29
A `C_ndp` object together with a `C_nda` forms a NDP *that conforms to Definition 32.*

[48] For a detailed introduction to this subject see [Auc99].

A `C_ndp` object stores:

- a topological hierarchy (`C_topological_hierarchy`) object that is subject to the planning
- a commodity (`C_commodity`) object representing the load that the highest network layer has to carry at the time of planning
- the planning period (`C_period`)
- a forecasting algorithm (`C_forecast`)
- a set of (hard) constraints (`C_constraint`)
- a set of soft constraints (`C_soft_constraint`)

The method `C_ndp::objective()` calculates the objective function of the current state of the network design (which is given by the topological hierarchy object). The functionality provided by the base class calculates the objective function by summing up the costs of all used facilities over the planning period plus the (weighted) penalties from all soft constraints.

The class `C_forecast` is an interface class for forecasting algorithms that calculates an expected commodity at any time in the future given a commodity at the present time.

The class `C_constraint` is an interface for hard constraints in $\mathcal{Z}$ according to Definition 25. A constraint takes a network design problem (in the method `C_constraint::is_fulfilled()`) and decides whether the constraint is fulfilled or not.

The class `C_soft_constraint` is an interface for soft constraints in $\mathcal{Z}$ according to Definition 26. A soft constraint takes a network design problem (in `C_soft_constraint::penalty()`) and returns a penalty (i.e. a non-negative number) if the associated condition is not fulfilled, otherwise it returns zero. Usually the penalty grows with the "distance" to the condition that should be fulfilled. The penalties of soft constraints contribute to the objective function of the object.

5.3.2.9 Network Design Algorithms `C_nda` is an interface for network design algorithms. There is a wide range of possibilities of how to design communication networks that depend heavily on the given facilities, protocols, network layers and many more.

Therefore, the class hierarchy derived from `C_nda` implements well known strategies for network design (realized by the *strategy* design pattern) without placing restrictions on new design ideas.

The following strategies are supported:

- `C_singlelayer_nda` is an interface for a single layer design algorithm.
- `C_accesstier_nda` is an interface for designing access networks within a layer.
- `C_multitier_sl_nda` is an interface for a single layer design algorithm consisting of at least one access design algorithm (`C_accesstier_nda`) and a backbone design algorithm (`C_singlelayer_nda`).
- `C_multilayer_nda` is an interface for a design algorithm that plans more than one layer at a time. The default implementation propagates the highest layer commodity from the top to the bottom layer and solves each layer by a single layer algorithm.
- `C_incremental_nda` is an interface for a incremental network design algorithm.
- `C_evolutionary_nda` is an interface for an evolutionary network design algorithm.

5.3.3 The Extended Design Classes

The second part of the *FooNet* framework extends the core design by a library of useful algorithms and applications. An overview of the available algorithms is given in Figure 21. A more detailed description and the references to the implemented algorithms can be found in [Fis99, Fis00].

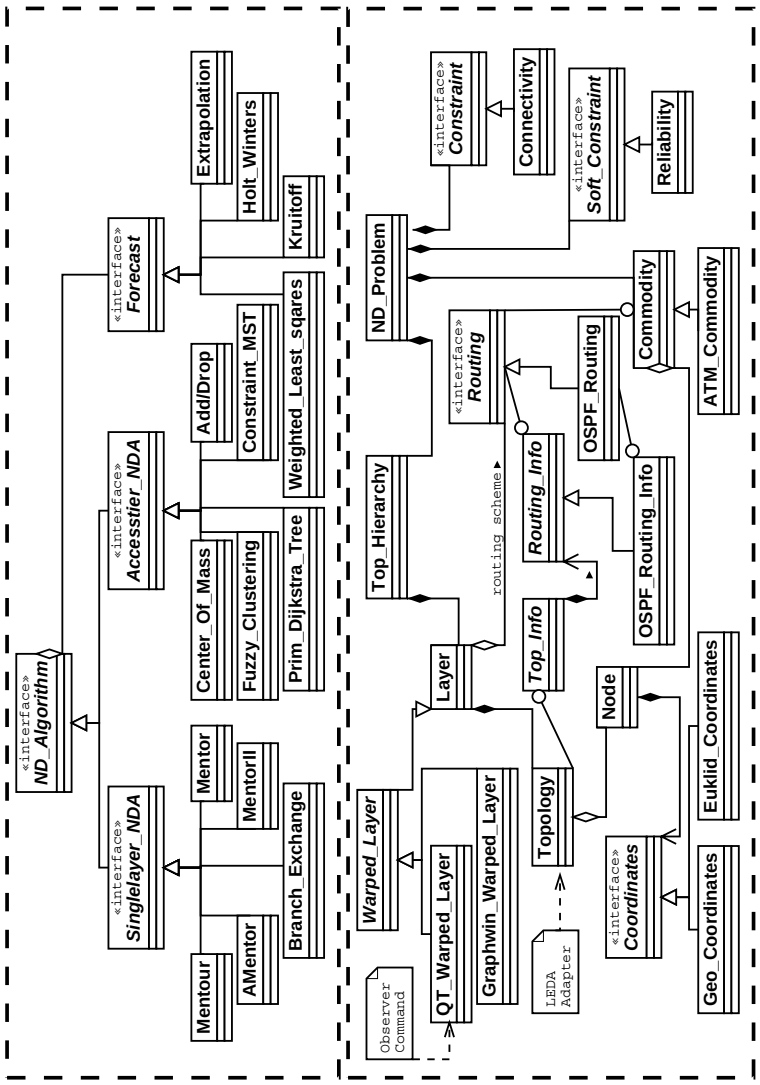

Figure 21: Static UML Extended Class Diagram

5.3.4 Network Design using *FooNet*

The design principles that are realized by *FooNet* are more than a simple implementation of a set of base classes for network design problems. The following non-trivial functionality is entailed with *FooNet*:

- The design relies on a set of well known and well understood design patterns. The internal implementations profit from this fact and so does the software designer who uses *FooNet*. Examples of design patterns that may be used by the software designer are:
  - the *visitor* design pattern for topological elements

  - the *factory* design pattern for facility creation
  - the *warper* design pattern for adding functionality to network layers
  - the *strategy* design pattern for the creation of new network design algorithms
  - the *prototype* design pattern for transparently cloning objects
  - the *virtual constructor* and *builder* design pattern for persistence

- All important functionality is implemented in virtual base classes, i.e. interfaces. The interfaces provided by *FooNet* are:

  - `C_node_info` - extends a network node by additional information
  - `C_topology_info` - extends a topological element by additional information
  - `C_coordinate` - allows the use of arbitrary coordinate systems
  - `C_routing` - allows the software designer to realize arbitrary routing schemes which are used whenever routing functionality is required
  - `C_routing_info` - extends each topological information by the necessary routing information
  - `C_facility`, `C_node_facility`, `C_edge_facility` & `C_evol_facility` - interfaces for families of facilities
  - `C_facilityclient` - interface for selecting facilities having certain attributes
  - `C_constraint` & `C_soft_constraint`- interface for arbitrary (soft) constraints
  - `C_nda` - interface for arbitrary network design algorithms (including the interfaces of the derived classes)

  The software designer can use the default functionality given by each interface, but no restrictions are imposed on him. He can always overrule them by deriving his own class that realizes the desired functionality. For example, the objective function of a network design problem and the selection & upgrade of facilities are useful candidates that can and should be overridden by the software designer.

- The design of the persistence concept guarantees that each piece of information is stored only once. This is similar to the "normal forms" of relational database systems.

- The cost structure realized by *FooNet* bases on a well known and widely accepted theory of finance, the discounted costs. *FooNet* is the first tool that explicitly takes this notion into account. The support for evolving facilities `C_evol_facility` is a direct consequence.

- The extended design provides a repository of well-known algorithms used in network design. These algorithms may be helpful for solving a concrete design problem.

- *FooNet* implements Data Exchange using XML. The Extensible Markup Language (XML) [XML98] is a proposed recommendation from the WWW-Consortium (`http://www.w3c.org`) for a file format to support the distribution of electronic documents. XML is a subset of the SGML (Standard Generalized Markup Language) and data is processed in human readable form. The outstanding feature of XML is the fact that unlike other formats it contains also information about how to process the data, i.e. the data describes its own format. Like HTML, XML is a markup language, which relies on the concept of rule-specifying tags and the use of a tag-processing application that knows how to deal with the tags. In contrast to HTML, XML is a meta markup language which allows to define application-specific markup-tags. A software module, called XML processor, is used to read an XML document and to provide access to its content and structure.

- Last but not least, *FooNet* is designed to offer the highest possible degree of code reuse. However, only the future will reveal how good this design is in practical use.

5.4 Outlook

The author is aware of the fact that the present version of *FooNet* is not nearly covering all facets of network design problems. It lacks a lot of features that are useful or even necessary for some problems.

The following work is intended to be done in the near future:

- Include support for usage dependent costs.
- Finish the interfaces and provide default implementations for the network performance analysis part.
- Include extensions for planning mobile networks.
- Increase the number of network design strategies derived from `C_nda`.
- Increase the number of algorithms and utilities in the extended framework.
- The XML part of *FooNet* is based exclusively on proposed standards of the W3C. However the DTDs of XML lack a support of object-oriented datatypes. The W3C is currently working on a "Schema for object-oriented XML" [SOX99] that will solve this problem. A soon as this specification is available as a proposed standard, it will be supported by *FooNet*.

6 Meta-Heuristic Search Algorithms and Application of Concepts to a Real-World Problem

6.1 Meta-Heuristic Search Algorithms

Appendix A gives an overview and an assessment of different types of network design algorithms. The most promising type with respect to computational tractability and optimality are meta-heuristic search algorithms. In the last few years, in particular the field of operations research identified meta-heuristic search algorithms as a serious alternative to classical optimization algorithms. The following sections will motivate why meta-heuristic search algorithms show their qualities especially in network design problems.

6.1.1 Fundamental Properties of Network Design Problems with Respect to Optimization

The problem under consideration shows a number of essential traits that can be found in the field of dimensioning of large-scale systems in general. The thesis of M. Greiner [Gre96] discusses existing mathematical optimization algorithms with respect to this type of problems. Large-scale systems have the following properties:

- P_1: The objective function is computationally expensive to calculate. As a consequence, most of the computational power spent in the optimization algorithm is due to the calculation of the objective function.
- P_2: The problems belong to the field of Combinatorial Optimization, therefore the objective function is not differentiable nor continuous.
- P_3: It is sufficient to know the solution with moderate precision.

As an immediate consequence of $P_1 - P_3$, an optimization algorithm should have the following properties:

- A_1: The solution should be obtained by as few calculations of the objective function as possible, since the number of these calculations determines its computational efficiency.
- A_2: Every iteration step of the optimization algorithm should generate a feasible solution $\llcorner x \in \mathcal{S}_{\mathcal{V}} \lrcorner$. This enables the designer to stop the optimization algorithm after any amount of computation time yielding an (at least optimized) solution.
- A_3: To speed up the optimization algorithm, the use of a substitution model that allows an efficient computation of the objective function (bought by a reduced precision) is of advantage. The "precise" objective function value can be calculated on special occasions, for example when a "new best" solution is found.

6.1.2 Some Notes on the Structure of Network Design Problems

Definition of Concept 53 (State Space Graph, Landscape)
Each solution x of a mathematical model (see Definition 3) can be regarded as a vertex in a (directed) ***state space graph*** $\llcorner \mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}}) \lrcorner$ *and is associated with its objective function value $o(x)$. Considering the values as altitudes, this induces a* ***landscape****, with "valleys" containing local minima and "mountains" containing the local maxima.*

Remark 30

The feasible solution space $S_{\mathcal{V}}$ is a subset of the vertices $V_{\mathcal{G}}$ of the state space graph.

Since the state space graph contains non-feasible solutions, it is common practice to make vertices $\llcorner x \notin S_{\mathcal{V}} \lrcorner$ unattractive to the optimization process by using soft constraints which penalize non-feasibility.

Remark 31

The more valleys and large plateaus an optimization problem has, the more difficult it is to guide the search towards an optimal solution.

One important, largely unaddressed field in optimization is the understanding of the structure of the state space graph. In [SERW98] it is shown[49] that multi-criteria[50] optimization problems (more graphically denoted as frustrated optimization problems) are characterized by a tremendous number of nearly evenly good solutions that have to be found in a very rugged landscape which is induced by the objective function. The author is aware of the fact that a strict mathematical proof of this statement is missing and even may not exist in general, but at least there exists strong evidence for this statement in "real-world" problems. The literature supports this evidence by the fact that a lot of works about heuristic network design algorithms which produce[51] near-optimal solutions (e.g. [XCG95, XCG98, SERW98, Bec98]) are published.

Remark 32

A further complication is induced by the weighing between different criteria in the objective function. Due to the "fuzziness" entailed with that weighing, the search for a global optimum is most difficult if not impossible since a small shift in the weight could move the optimum in a completely different region of the state space graph.

As a consequence of these observations, the author proposes that practical network design problems can be treated as "good natured" in the sense that either there are a sufficiently large number of near optimal solutions in the state space graph, or the state space graph provides a landscape which allows to find sufficiently good solutions by "intelligent" search heuristics in a reasonable time.

6.1.3 Classification and Assessment of Heuristic Search Algorithms

There is exhaustive literature about heuristic search methods. The author can recommend e.g. [Ree93, Rar95, RSORS96] for a thorough introduction. Basically, there are two different kinds of heuristic search algorithms:

1. **Constructive search algorithms:** Constructive search algorithms try to find the best possible solution by constructing a solution step by step relying on problem-specific knowledge. Usually, the runtime of the algorithm is known in advance and depends on the number of nodes, number of links etc. In general, the construction rules guarantee a feasible solution.

[49] In the context of optimizing road networks, which are structurally very similar to telecommunication networks.

[50] Whereby the overall objective function is calculated by the Archimedian approach, i.e. the weighted sum of all criteria.

[51] At least for sufficiently small problem sizes, where optimal solutions are known!

2. **Improving Search Heuristics** or **Meta-Heuristics**: The search begins at an initial vertex v_0 in the state space graph. Guided by a set of rules[52], each iteration i constructs a set of new vertices N depending on the previously found solutions $\llcorner v_0, \ldots, v_{i-1} \lrcorner$. Then, the algorithm either makes a transition to a vertex v_i out of N or stops if some stopping-criterion is met (e.g. none of the vertices $\llcorner v \in N \lrcorner$ has a better objective function value). There are a huge number of variations of this general scheme. However, meta-heuristics always show the following characteristics:

   - The search aims at improving the found solution at each iteration, but exceptionally may accept a non-improving solution.
   - The potential successors of the current solutions are constructed by a set of problem-independent rules.

Remark 33

In general, improving search heuristics honor the property A_2, i.e. the search can be stopped any time yielding an at least optimized solution.

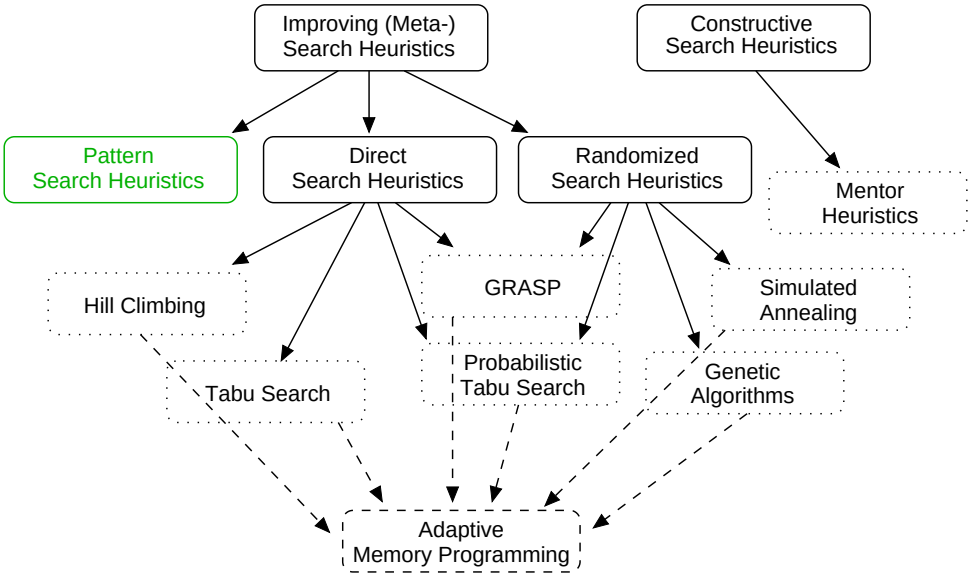

Figure 22: Classification of Search Heuristics

Definition of Concept 54 ($\mathsf{A}(i)-$optimization Algorithm)

An optimization algorithm is called $\mathsf{A}(i)-$optimization algorithm [Gre96] if it relies explicitly on derivates of the objective function of the i-th order.

Due to the property P_2, the algorithms used in network design can be restricted to the $\mathsf{A}(0)$-type. $\mathsf{A}(0)$-optimization algorithms only use the solutions calculated so far to guide the search process in the next iteration. Figure 22 gives an (incomplete) overview of how search heuristics can be classified. The various improving search methods differ by the rules how the transitions are generated and how the transitions are selected among the possible transitions. *Direct search heuristics* and *randomized search heuristics* rely on the notion of a *neighborhood*:

[52] Due to the fact that the rules often mimic processes from physics or nature and are not specific to a single problem they are denoted as **meta-heuristics**.

Definition of Concept 55 (Neighborhood, Transition, Candidate Set)
A mapping $\llcorner\mathcal{N} : V_\mathcal{G} \to \wp(V_\mathcal{G})\lrcorner$ *which associates each vertex* $\llcorner v \in V_\mathcal{G}\lrcorner$ *with a subset of* $V_\mathcal{G}$*, is denoted as a* ***neighborhood*** *if:*

i) $\forall v \in V_\mathcal{G} : v \notin \mathcal{N}(v)$
ii) $\forall v \in V_\mathcal{G} : |\mathcal{N}(v)| \geq 1$

The set $\mathcal{N}(v)$ *is called neighborhood of* v*. The move from* v *to one of its neighbors is called a single* ***transition****. The edges of the state space graph* $\mathcal{G}$ *with the neighborhood* $\mathcal{N}$ *are defined by the transitions, i.e.* $\llcorner\forall v \in V_\mathcal{G}, \forall v' \in \mathcal{N}(v) : (v, v') \in E_\mathcal{G}\lrcorner$*.*

The state space graph $\mathcal{G}$ *for a given problem and neighborhood is called* ***well-defined*** *if it fulfills the following condition:*

"From every $\llcorner v \in V_\mathcal{G}\lrcorner$ *there exists a path to an optimal solution in* $\mathcal{G}$*."*

If the size of the neighborhood is large, due to property P_1 *it makes sense to evaluate only a subset of the neighborhood which is denoted as* ***candidate set*** $\mathcal{N}_C(v)$*.*

Definition of Concept 56 (Representation, Encoding, Decoding)
The concept of the neighborhood is heavily influenced by the internal ***representation*** $\mathcal{R}$ *used by the improving search heuristic. The space of all representations is denoted as* $S_\mathcal{R}$*. Furthermore, a representation is entailed with two transformations:*

- *a transformation* $E_\mathcal{R}$*, denoted as* ***encoding****, which assigns a representation* $\llcorner y \in S_\mathcal{R}\lrcorner$ *to every solution* $\llcorner x \in S_\mathcal{V}\lrcorner$
- *a transformation* $D_\mathcal{R}$*, denoted as* ***decoding****, which assigns a (not necessarily feasible) solution* x *to every representation* $\llcorner y \in S_\mathcal{R}\lrcorner$

Often the neighborhood in the state space graph is given indirectly by the operations defined on the representation used.

Remark 34
The following text is less formal with respect to distinguishing between a solution and its representation. Often the terms "solution" and "representation" can be used interchangeably. Note however, that in general $E_\mathcal{R}$ *is not the inverse function of* $D_\mathcal{R}$ *and vice versa.*

Often an encoding is not even necessary because there is no need to transform a solution into a representation.

The next sections introduce some important examples of heuristic search algorithms. For notational convenience, a minimization process is assumed.

6.1.3.1 Hill Climbing Hill Climbing (see Algorithm 1) is the easiest improving search heuristic. It evaluates all neighbors of the current solution v and makes a transition to the "best" neighbor. The iteration stops if no new best neighbor can be found in the neighborhood, i.e. a local optimum is reached.

The most restrictive drawback of the Hill Climbing heuristic is that it misses information about the global state, i.e. it critically depends on the initial vertex and cannot escape a local minimum (with respect to the neighborhood). A first improvement is to run Hill Climbing from many initial

Algorithm 1 Basic Hill Climbing

1: choose initial vertex $\llcorner v \in V_{\mathcal{G}} \lrcorner$
2: $v' := v$
3: **while** $\llcorner o(v') \leq o(v) \lrcorner$ and v' not visited before **do**
4: $\quad v := v'$
5: $\quad v' := n \in \mathcal{N}(v) \ : \ o(n) \leq o(n') \ \forall n' \in \mathcal{N}(v)$
6: **end while**
7: **return** v

vertices which is denoted as **multi-start** Hill Climbing. A more sophisticated variant of multi-start Hill Climbing which uses elements of randomness are *Greedy Randomized Adaptive Search Procedures* (GRASP). An overview can be found e.g. in [Res98].

6.1.3.2 Simulated Annealing Simulated Annealing (see Algorithm 2) is a very successful variant of improving search heuristics. It uses randomness to avoid getting stuck in a local minimum. At each step a transition is chosen randomly. If the transition improves the solution, it is immediately accepted. If the transition does not improve, it is accepted with a certain rate[53] related to the difference in the objective function values between the current and the new solution and scaled by the temperature C. Otherwise, a new move is chosen and the process is repeated.

Algorithm 2 Basic Simulated Annealing

1: choose initial vertex $\llcorner v \in V_{\mathcal{G}} \lrcorner$
2: **while** stopping criteria are not met **do**
3: $\quad$ choose $\llcorner v' \in \mathcal{N}(v) \lrcorner$ randomly
4: $\quad$ **if** $o(v') < o(v)$ **then**
5: $\quad\quad v := v'$
6: $\quad$ **else if** $\exp((o(v) - o(v'))/C) >$ `rand()` **then**
7: $\quad\quad v := v'$
8: $\quad$ **end if**
9: $\quad$ modify C according to the *cooling down scheme*
10: **end while**
11: **return** v

Practical implementations of Simulated Annealing start with a high temperature C and reduce it towards zero with each iteration according to a predefined **cooling down scheme**. The basic idea is to allow the search to wander randomly in the state space graph on its early stages by accepting worse solutions and focus only if it has discovered a particularly promising region in the landscape.

An appealing property of Simulated Annealing is its theoretical convergence property. If the neighborhood is well-defined, and the temperature is allowed to approach zero according to certain cooling down schemes, simple Markov chain analysis shows that Simulated Annealing guarantees an almost sure[54] convergence to the global optimum. However, the real number of steps may be hopelessly vast. In general, the cooling down scheme is one of the critical parameters of Simulated Annealing and has to be chosen with great care. A very successful modification to Simulated Annealing is *Adaptive Simulated Annealing* which allows to re-heat the temperature

[53] The function `rand()` returns a uniformly distributed random number between 0 and 1.
[54] i.e. with probability 1 independent of the starting point

if necessary. Intensely used stopping criteria include e.g. the number of iterations, a sufficiently low temperature C and the number of non-improving iterations.

6.1.3.3 Genetic Algorithms The term "Genetic Algorithm" is derived from the similarity with the processes that take place in reproduction of biological life. The genetic algorithm manipulates a set of solutions, i.e. a *population* out of individuals. Each individual is represented by a genetic code, the *chromosomes*, that can be manipulated through the genetic operators *mutation* and *cross-over* (or *mating*). The genetic operators define the neighborhood of the search heuristic from which only a candidate set is realized in each *generation* (i.e. iteration of the search). Ideally, all individuals produced by the genetic operators correspond to feasible solutions. Each individual is associated with a *fitness value* that reflects how good it is, compared to other solutions in the population. As in nature, a *selection* process reduces the population in each iteration towards a population consisting of only the fittest individuals. The higher the fitness value of an individual is, the higher are its chances of survival and reproduction. It can be shown that genetic material which has promising characteristics is more frequent in the next generation (schemata theorem).

Algorithm 3 Basic Genetic Algorithm

1: choose initial population $\llcorner Pop \subseteq V_{\mathcal{G}} \lrcorner$
2: **while** stopping criteria are not met **do**
3: choose a subset of the population $\llcorner Pop' \subseteq Pop \lrcorner$
4: generate a new population Pop'' by reproduction of the individuals in Pop' (**crossover**)
5: change a part of the populations Pop and Pop'' by randomly changing their genetic code (**mutation**)
6: select the new population Pop out of the initial population Pop and their descendants Pop'' (**selection**)
7: **end while**
8: **return** best individual out of Pop

Similarly to Simulated Annealing, the stopping criteria may consist of the number of iterations, the number of non-improving iterations, and statistical properties of the population.

6.1.3.4 Tabu Search Tabu Search is a direct search heuristic that has the ability to overcome the limitation of local optimality which is encountered e.g. by Hill Climbing. The term "tabu" has no elements of supernatural origin, instead, Tabu Search guides the search through difficult regions of the state space graph by imposing restrictions on the available transitions, i.e. "tabu transitions". This is accomplished by a systematic use of a memory structure which stores information about the search. The purpose of this memory is threefold: First, it avoids the cycling of the search within a local minimum; Second, it diversifies the search by driving it towards non-explored regions of the state space graph; Third, it can help to intensify the search in a particularly promising region. Connected with the three goals there are three different types of memory:

Short-term memory $\mathcal{M}_s$: The short term memory operates by imposing restrictions on the neighborhood. After an elementary transition the restriction assuring that the transition cannot be reversed is imposed. How long a given restriction is kept up depends on a parameter called the **tabu tenure** which identifies the number of iterations a particular tabu restriction remains in force. The tenure can either be fixed or variable, but a tenure that

varies within a small range about a central value often proves more robust. An important TS component is the use of **aspiration criteria** to allow a restriction to be overridden if the outcome of a move is sufficiently desirable. A commonly used criterion is to override the restriction if the candidate would lead to a new best solution. The short term memory helps to escape local minima and makes the search avoid recently searched areas.

Medium-term memory $\mathcal{M}_m$: This memory collects information on good local minima found so far. The memory is used to **intensify** the search around good solutions in order to find even better solutions in the close vicinity. A reasonable strategy is to return to a solution in $\mathcal{M}_m$ after a number of non-improving iterations.

Long-term memory $\mathcal{M}_l$: The complementary to the short term memory is the long-term memory. The purpose of this memory is to drive the direction of the search into unexplored areas of the solution space, i.e. **diversify** the search. This memory is often **frequency based**, that is, based on counts of solution or transition attributes. This frequency is normalized and scaled to create a **penalty term** which is added to the objective function thus making already visited areas of the state space graph less attractive.

Algorithm 4 Basic Tabu Search

1: choose initial vertex $\llcorner v \in V_{\mathcal{G}} \lrcorner$
2: initialize tabu memories $\mathcal{M}_s$, $\mathcal{M}_m$, and $\mathcal{M}_l$
3: **while** stopping criteria are not met **do**
4: evaluate neighborhood $\mathcal{N}(v)$
5: assign the best non-tabu solution $\llcorner v' \in \mathcal{N}(v) \lrcorner$ to v
6: store the information about v in $\mathcal{M}_s$, $\mathcal{M}_m$, and $\mathcal{M}_l$
7: **end while**
8: **return** best solution found

Tabu Search is an extremely general heuristic, it can be formulated to include Simulated Annealing and Hill Climbing as a special case. In the literature, very good results have been obtained, which were in most of the cases superior to alternative approaches. Also, observation of applications have shown that the execution time of Tabu Search outperforms the best local search heuristics currently available. Still, no quality guarantees have been given so far, and the strength of the method is demonstrated on a purely experimental basis.

6.1.3.5 Adaptive Memory Programming From a theoretical point of view, convergence theorems for Simulated Annealing and Genetic Algorithms exist. In practice, these theorems are not effective since they would require more computation time than complete enumeration. However, the following four characteristics can be observed in all of the introduced algorithms (each to a varying degree):

- the algorithms memorize already visited solutions in some way
- the algorithms produce new solutions during the search process by transitions that depend on the memorized solutions
- the algorithms influence the search by random elements
- the algorithms apply a local search by greedy improvements

In [TGGP98], meta-heuristic search algorithms are reviewed with respect to their memory structure, and a unified representation, which is denoted as **Adaptive Memory Programming**

(AMP), is proposed. Algorithm 5 shows a unified framework from which each of the algorithms can be specialized.

Algorithm 5 Generic Adaptive Memory Programming

1: initialize memory structure
2: **while** stopping criteria are not met **do**
3: generate a provisory set of vertices V using the data in the memory
4: improve V with local search; let V' be the improved set of solutions
5: update the memory and V using the knowledge represented by V'
6: **end while**
7: **return** best solution v stored in the memory

However, AMP is not an all embracing solution for improving search algorithms. In a concrete setting, the designer still has to structure and transform the problem in an appropriate way for which the one or other meta-heuristic works better. The main advantage of AMP lies in the new, different view on meta-heuristics which enables the designer to develop better, tailor-made algorithms for a given problem.

6.1.3.6 Summary The successful application of heuristic search algorithms relies critically on devising an appropriate method of representing the problem and a concept of neighborhood which allows a clever method for maneuvering in the search space. Experiments show that especially in the context of network design problems, algorithms that base on Tabu Search are more economical (in terms of number of evaluations of the objective function) than Genetic Algorithms and Simulated Annealing. This fits in the picture of AMP because Tabu Search uses the memory more intensely than any other technique. Therefore, Tabu Search is more goal-directed and may reduce the number of iterations necessary to find sufficiently good solutions. Another advantage of Tabu Search lies in the fact that an effective modeling leads to a robust search that is not too sensitive to parameter settings.

6.2 Exemplary Application of Concepts

6.2.1 Overview

The concepts which were introduced in the previous chapters will be put into practice by the example of a real-world setting: The backbone-network design task of the Wissenschaftsnetz (WiN). The design environment and the design decisions of the WiN will be introduced in Section 6.2.2.

Section 6.2.3 demonstrates how the static network design problem can be solved using a Tabu Search meta-heuristic. It discusses the necessary representation and neighborhood structure, as well as the calibration of the heuristic.

Section 6.2.4 tackles the evolutionary network design problem under complete knowledge trying to further improve the initial (static) network design that was found in the previous section. Thereby the search takes into account (certain) knowledge about the future traffic development.

Section 6.2.5 and Section 6.2.6 extend the network design problem by the introduction of uncertainties about the traffic development. The corresponding network design problems under incomplete knowledge will be solved by using the concept of robust optimization that was introduced in Section 4.5.4.

6.2.2 The Settings of the Wissenschaftsnetz (WiN-Problem)

6.2.2.1 Design Environment The design environment $\mathcal{E}$ which is regarded as an example is in accordance with the reality found in the WiN in July 1999 (except for some mainly unimportant end-nodes). As already mentioned, the WiN satisfies the demands of the German research community and is maintained by the Deutsches Forschungsnetz-Verein (DFN-Verein). The traffic matrix under consideration consists of 335 end-nodes whose locations and relative weights (i.e. the sum of in- and out-going traffic) are shown in Figure 23. The network transports almost exclusively IP-traffic and uses the OSPF-routing protocol. For simplicity reasons, other constraints (such as survivability or hop-limits) are left out. These could be included, e.g. by soft constraints, albeit at higher computation time of the objective function. In summary, the set of constraints $\llcorner\mathcal{Z} = \{\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3\}\lrcorner$ which is burdened on the design is given as:

$\mathcal{Z}_1$: the OSPF routing protocol specification (see [RFC98b])

$\mathcal{Z}_2$: the demand that the initial design has to be able to transport the requirement matrix

$\mathcal{Z}_3$: the set of available facilities

As a further simplification of the example, the task of the designer only consists of finding a *backbone network design* $\llcorner\mathcal{D}_0 = (\mathcal{T}_0, \mathcal{C}_0)\lrcorner$ for the given set of backbone-nodes V_b. The access design will be omitted in the calculations. Figure 24[55] shows the underlying access-design as well as the backbone-node locations[56]. The static cumulated traffic matrix R between the backbone-nodes can be found in Appendix E.1.1.

A network design algorithm has to assign values to the following design variables $\llcorner\mathcal{V} := \mathcal{D}_0\lrcorner$:

- the topology of the backbone network $\mathcal{T}_0$
- the facilities $\llcorner F_{\mathcal{D}_0} \subseteq F_0\lrcorner$ deployed at each topological element (part of $\mathcal{C}_0$)
- the OSPF-routing parameters for each edge in $\mathcal{T}_0$ (part of $\mathcal{C}_0$)

6.2.2.2 Traffic Forecasts The development of the future traffic requirements is identified as the key uncertainty in the *ENDP*. One particularity of the WiN is the fact that the traffic requirements are quite asymmetric. More than half of the traffic transported by the network originates from international sources (specifically from the United States).

Figure 25 shows the overall traffic development together with the development of national and international traffic. A thorough discussion of the present traffic situation and the future development of the WiN can be found in [Gog99] and [Jes99].

As it can be seen easily, not only the overall growth rate of the traffic is of vital interest to the designer, but also the ratio of the international to the national traffic growth. For the exemplary discussion, two different developments of traffic, each of them occurring with equal probability, are assumed:

Development/State s_1: predicts a traffic growth rate of 2 per year for the national traffic, and a traffic growth rate of 3 per year for the international traffic.

[55] The thickness of lines in all following figures corresponds the the capacity of the deployed facility.

[56] The access-network design was done by an algorithm that took the node-weights as well as "center of mass" calculations into consideration. It is included in the *FooNet* distribution.

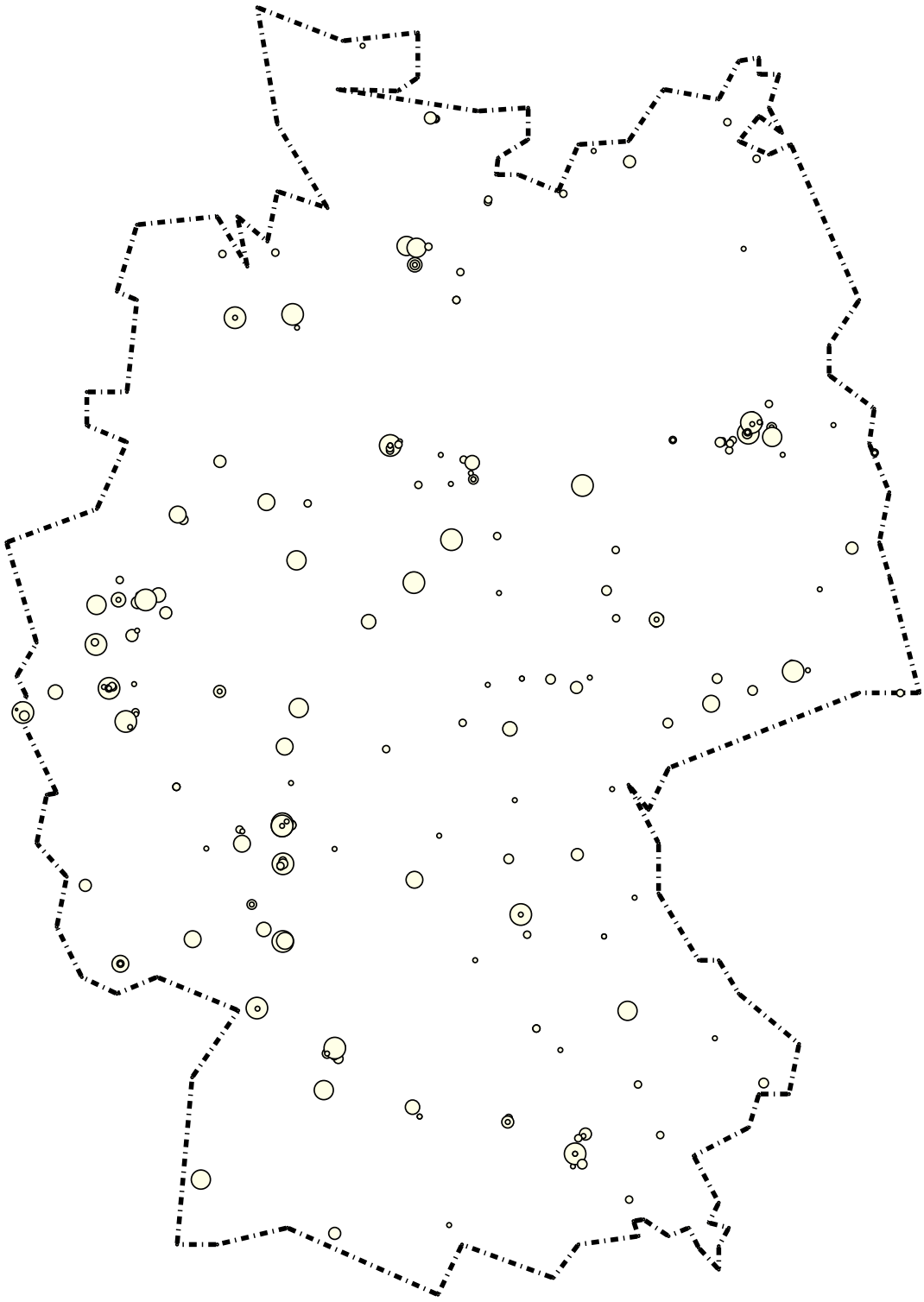

Figure 23: WiN-Problem: 335 End-Nodes Scaled by their Relative Weight

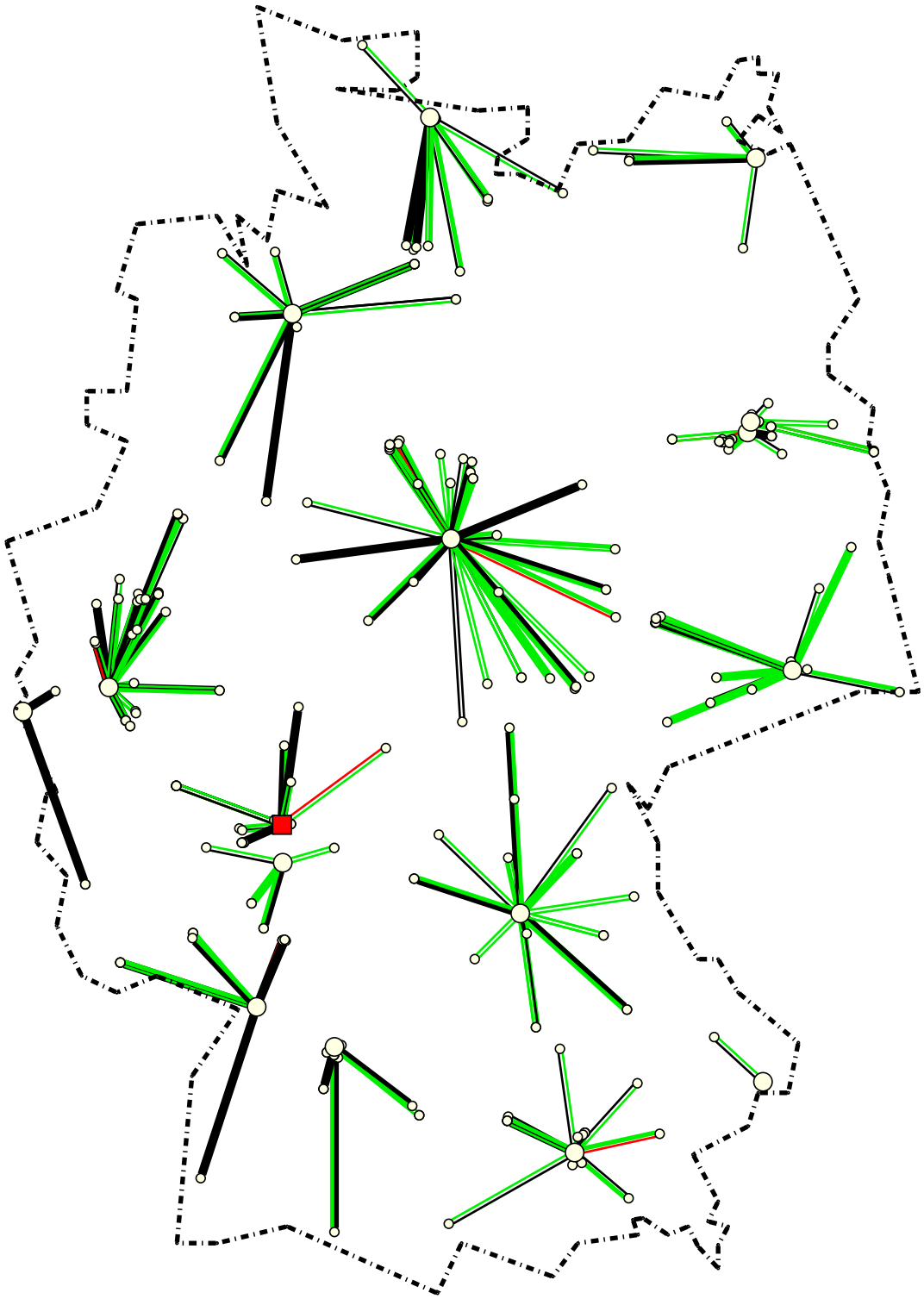

Figure 24: WiN-Problem: Access Network and Backbone-Node Locations

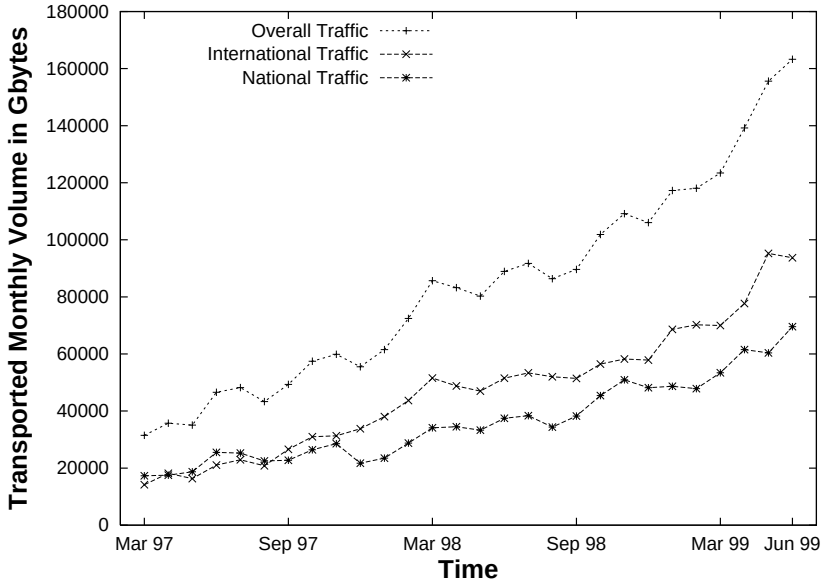

Figure 25: Traffic Development in the WiN ([Gog99])

Development/State s_2: predicts a traffic growth rate of 3 per year for the national traffic, and a growth rate of 2 per year for the international traffic.

The corresponding traffic matrices projected in one year's time from the initial matrix R are denoted as R_1 and R_2.

6.2.2.3 Cost Model Obviously, the cost model (which is part of $\mathcal{E}$) is one of the most important influence factors of what an optimized network topology $\mathcal{T}$ looks like. In the example, a concave power-law cost model according to Equation (19) (an overview can be found in Appendix C.4) is assumed. The cost model is specified by the following settings:

$$\widetilde{C}(f, n_1, n_2, t) = s_f + \mathrm{PVR}(f^\alpha \cdot l(n_1, n_2), 12, t) \tag{19}$$

where:

- At time t_0 of the initial design $\mathcal{D}_0$, the set of facilities F_0 includes three different facility-types for each possible edge

  $$\llcorner FT = (f_1 := 34\,\mathrm{Mbps}, f_2 := 155\,\mathrm{Mbps}, f_3 := 622\,\mathrm{Mbps})\lrcorner$$

  (part of $\mathcal{Z}$).
- At planning time Δ_1, which is in 12 months time from t_0, the set of available facilities F_1 extends F_0 by a new facility-type f_4 having 2.044 Mbps capacity.
- The setup costs $\llcorner s_f, f \in FT \lrcorner$ are fixed for each facility-type ($10 \cdot 10^3$ for f_1, $25 \cdot 10^3$ for f_2, $50 \cdot 10^3$ for f_3, $100 \cdot 10^3$ for f_4) and independent from the edge at which the facility is deployed. s_f is chosen to have the order of magnitude of the average reoccurring costs caused by a facility in a two months period.
- The reoccurring costs are payable once a month in advance.
- There are no termination costs.

- The economy of scale parameter is set to $\lfloor \alpha := 0.7 \rfloor$.
- The interest rate is set to $\lfloor r := 0.05 \rfloor$.
- $\widetilde{C}(f, n_1, n_2, t)$ returns the discounted costs of deploying a new facility type $\lfloor f \in FT \rfloor$ between the backbone nodes n_1 and n_2, $\lfloor n_1, n_2 \in V_b \rfloor$, and operating it for the duration of t months.
- $l(n_1, n_2)$ corresponds to the distance in terms of kilometers between the backbone-nodes n_1 and n_2.
- The upgrade costs from an already deployed facility-type f to a facility-type f' having a different capacity are set to 50% of the setup costs s'_f of the new facility. Therefore, it is economically attractive to upgrade an already deployed facility.
- There is only one node facility-type available in F_0 that has fixed costs and can cover all throughputs. As a consequence, the node costs can be omitted in the following calculations.
- The costs stay constant over the observation period.
- The line facilities have to be deployed symmetrically, i.e. the facility-type deployed at an edge has to be identical with the facility-type of the anti-parallel edge.

There are two reasons for choosing exactly this cost model:

- It resembles "real" cost models that are used in practice to a high degree.
- The order of magnitude of the setup-costs and the difference between setup and upgrade costs allows a cost efficient evolutionary network design.

An example of a cost matrix can be found in Appendix E.1.3. However, it has to be mentioned that all of the presented Tabu Search heuristics work with arbitrary cost models.

6.2.2.4 Objective The primary objective $\lfloor o_T : \mathcal{S}_{\mathcal{D}^T} \rightarrow \mathbb{R}^+ \rfloor$ is to find the minimum discounted cost network $\mathcal{D}^T$ over the observation period T. The overall discounted network costs are calculated according to the cost model of the previous section (a more general formula can be found in Appendix C.3).

6.2.3 WiN-Problem: Static Network Design

6.2.3.1 Introduction The task of the *SNDP* is to find a least cost network that is able to transport the traffic requirements R without capacity shortage. Furthermore, R is assumed to be static over the observation period $\lfloor T_0 = [t_0, \Delta_1] \rfloor$, whereby the observation period has a duration of 12 months, i.e.

$$\Delta_1 := t_0 + 12 \text{ months}. \tag{20}$$

6.2.3.2 Network Representation $\mathcal{R}_1$: No Capacity Shortage The representation $\mathcal{R}_1$ is heavily influenced by the characteristics of the OSPF routing protocol. OSPF is a link-state routing protocol which transports the traffic between two nodes on the shortest path according to an edge metric consisting of user-defined "lengths" (denoted as **OSPF-lengths**). The metric allows to define the OSPF-length for each edge independently from all other edges, from its direction, and from its physical length. One particularity of OSPF is that the traffic from any source node to a given destination node n_d follows the shortest path tree which is rooted at n_d. For convenience, the OSPF-lengths are bounded by the interval $[1, \text{OSPF}_{\max}]$.

Definition of Concept 57 (Supply Graph)
The supply graph $\llcorner G_S = (V_S, E_S) \lrcorner$ of a network contains all nodes of the network in V_S and a set of edges in $\llcorner E_S \subseteq V_S \times V_S \lrcorner$ that are allowed in the topology $\mathcal{T}$ of a design.

Most often the supply graph is fully meshed. However, in the context of heuristic search algorithms, complexity considerations may demand a sparser supply graph where edges that have a low probability to be part of an optimal design are removed.

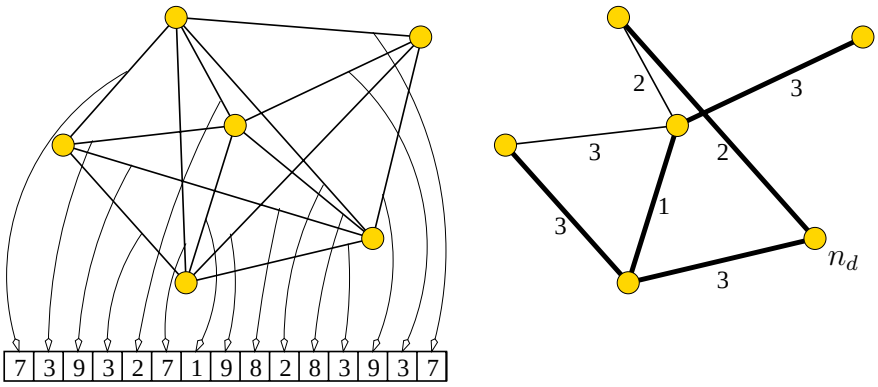

Figure 26: OSPF Network Representation (Undirected Graph)

Figure 26 (left) shows an example for a supply graph G_S with an OSPF-length attached to each edge. Figure 26 (right) shows the graph which only consists of those edges that are in fact used by the shortest path trees. Furthermore, the shortest path tree routed in n_d is illustrated (thick lines). For simplicity, both graphs are undirected.

The basic idea of the representation $\mathcal{R}_1$ was introduced in [Rie98] and was originally designed for the use with Genetic Algorithms. The representation assigns an arbitrary pseudo OSPF-length to each edge $\llcorner e \in E_S \lrcorner$. The traffic which flows over an edge can be calculated by an OSPF network loader (see Definition 17). Knowing the load on each edge, the line facilities are determined by choosing the minimal cost facility $\llcorner f \in F \lrcorner$ that is able to transport the load. The benefit of the approach is the fact that it covers the topology of the network as well as the configuration simultaneously, i.e. it is a joint representation of all design variables $\mathcal{V}$.

An internal representation v_r of a solution v simply consists of a vector

$$v_r \in [1, \mathrm{OSPF}_{\max}]^{|E_S|} =: S_{\mathcal{R}_1} \tag{21}$$

having an element $v_r[e]$ for each edge $\llcorner e \in E_S \lrcorner$ of the supply graph that contains its OSPF-length. The decoding $D_{\mathcal{R}_1}$ from the representation space to the solution space is done simply by the determination of the facility-type for each edge as described above and take over the OSPF-lengths in $\mathcal{C}_0$.

In the given setting, a problem can arise if an edge has a higher load than any of the available facilities can cover. These representations are made unattractive by adding a penalty term to the objective function (see Remark 30). Except this, every representation is a feasible solution, i.e. all other constrains $\llcorner z \in \mathcal{Z} \lrcorner$ imposed on the solution are automatically fulfilled.

The neighborhood $\mathcal{N}(v_r)$ of a representation $\llcorner v_r \in S_{\mathcal{R}_1} \lrcorner$ is created by modifying the value of an element $v_r[e]$ for a number of edges $\llcorner e \in E_S \lrcorner$. It is immediately obvious that we have to restrict this neighborhood to an appropriate candidate set since there may be a unlimited number of possible OSPF-lengths.

During the course of this work, many possible candidate sets were evaluated. A first successful candidate set $\mathcal{N}_C(v_r)$ was created by modifying each element of v_r (one at a time) by a uniformly distributed random number[57] m:

$$m \in \llcorner[-\mathrm{MOD}_{\max}, 0) \cup (0, \mathrm{MOD}_{\max}]\lrcorner, \quad \text{with } \llcorner 0 < \mathrm{MOD}_{\max} \leq \mathrm{OSPF}_{\max}\lrcorner \tag{22}$$

The candidate set $\mathcal{N}_C(v_r)$ consists of $|E_S|$ representations. $\mathcal{N}_C$ is well-defined in the sense that an optimal solution can be reached from a current one with non-zero probability.

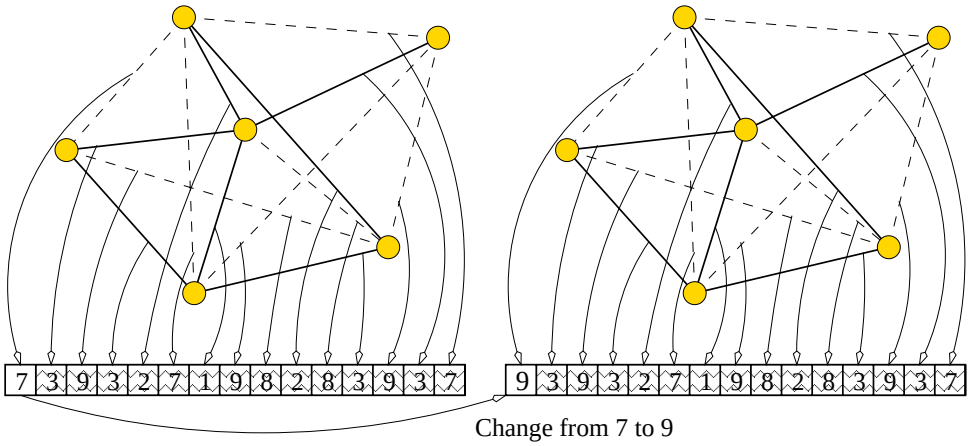

Figure 27: Modification of an "Inactive" Edge which Results in an Identical Solution

The major problem entailed with this candidate set is that it may contain a large number of identical solutions v having different representations v_r. The fundamental reason for this is that a modification of an OSPF-length $v_r[e]$ does not necessarily result in a different shortest path tree (illustrated in Figure 27).

The convergence of the search was extremely slow, since the neighborhood $\mathcal{N}_C(v_r)$ induces large plateaus in the landscape (see Remark 31). The problem of deciding whether a modification leads to a different design is not a trivial one. Of course, the decoding of the representation gives an answer but is computationally expensive.

The notion of **activity** proved to be an efficient solution to this problem:

1. During the calculation of the objective function $o(D_{\mathcal{R}_1}(v_r))$ mark all edges which are realized in the design $D_{\mathcal{R}_1}(v_r)$ as **active**, otherwise **inactive**. In Figure 27, all inactive edges are shown as dashed lines, whereas all active edges are shown as solid lines.
2. In the construction phase of $\mathcal{N}_C(v_r)$, restrict the allowed modification values, m, of inactive edges to negative values, i.e. $\llcorner m \in [-\mathrm{MOD}_{\max}, 0)\lrcorner$, thus increasing the chance of the line to become active.

Remark 35

In representation $\mathcal{R}_1$, the property i of Definition 55 is fulfilled for the representations v_r but not for the decoded solutions $D_{\mathcal{R}_1}(v_r)$. The effect of the activity concept is that representations in $\llcorner v' \in \mathcal{N}_C(v_r)\lrcorner$ for which $\llcorner D_{\mathcal{R}_1}(v') = D_{\mathcal{R}_1}(v_r)\lrcorner$ holds occur less frequently.

Another degree of freedom entailed with the neighborhood is the domain of admitted OSPF-lengths. Experiments have shown that OSPF-lengths of real numbers $\mathbb{R}$ lead to better results than OSPF-lengths of natural numbers $\mathbb{N}$.

[57] Of course, other distributions are possible too, but a uniform distribution proved to be a good choice.

Finally, the (time) complexity of one iteration step has to be determined. For notational convenience, let $\llcorner m := |E_S|\lrcorner$ and $\llcorner n := |V_S|\lrcorner$. To calculate the objective function value of a representation, a decoding is necessary. This involves to solve an "all pairs shortest path problem" which is implemented by solving a "single source shortest path problem" for each node $\llcorner n \in V_S \lrcorner$ using the well-known Dijkstra algorithm. The Dijkstra algorithm has the complexity $\llcorner O(m + n \cdot \log n)\lrcorner$. Having a candidate set with m neighbors as suggested above, the overall complexity of one iteration step can be identified as:

$$O(m \cdot (n \cdot (m + n \cdot \log n))) = O(m^2 \cdot n + m \cdot n^2 \cdot \log n) \tag{23}$$

6.2.3.3 Calibration of the Tabu Search heuristic The calibration of the parameters presented is the result of a number of tests. The calibration of the Tabu Search heuristic suggested in [XCG98] served as a first orientation and could be confirmed to be a good overall choice for our problem, too. Some of the conducted tests can be found in [Mel00].

Despite the fact that Tabu Search is known to be a heuristic whose ability to converge does not critically depend on its parameter settings, the convergence speed may decrease drastically with poor parameter settings. The following parameter settings (whose actual values are specified in parentheses) proved fruitful:

Short Term Memory $\mathcal{M}_s$: The *tabu tenure* used is of variable length and varies within a range (± 1) around the central value (7). Tests with a shorter or longer tabu tenure did not prove advantageous. After every transition, the edge e whose OSPF-length was modified is set "tabu", i.e. for the time the tabu restriction holds, transitions that would modify $v_r[e]$ are left out of account.

Medium Term Memory $\mathcal{M}_m$: During the **first phase** of the search the heuristic stores the best local minima found in $\mathcal{M}_m$ ($\llcorner |\mathcal{M}_m| = 6 \lrcorner$). A solution is regarded as a local minimum if no better solution is found for a fixed number of iterations (150). After the end-conditions of the first phase are met, the heuristic starts a **second phase** that relaunches the search beginning with the worst local minimum $\llcorner v_r \in \mathcal{M}_m \lrcorner$ for a fixed number of iterations (50). The representation v_r is removed from $\mathcal{M}_m$ and a possibly found better solution than v_r during the search is again inserted in $\mathcal{M}_m$. The heuristic ends if $\mathcal{M}_m$ is empty. The goal of the medium term memory is to *intensify* the search in the region of good solutions already found. This is in accordance with the observation that in the vicinity of good solutions there is a high chance to find even better solutions.

Long Term Memory $\mathcal{M}_l$: The long term memory is *frequency based* and activated in the first phase of the search only. With each iteration the heuristic keeps track of how often the OSPF-length of an edge $\llcorner e \in E_S \lrcorner$ was modified by the executed transitions. The long term memory is activated after a number (500) of iterations. After that, for each representation

$$\{v'_r \in \mathcal{N}_C(v_r) \,:\, v'_r \text{ differs from } v_r \text{ by the OSPF-length of } e\} \tag{24}$$

a penalty term is calculated that scales linearly with the frequency of e in the long term memory and can go as high as $x\%$ (7%) of the objective function value. The order in which the transitions are evaluated in the transition selection phase (see below) is influenced by the long term memory.

Transition Selection: After all neighbors in the candidate list $\mathcal{N}_C(v)$ have been evaluated, the heuristic checks whether a "new best" solution has been found. A "new best" solution

is accepted immediately, regardless of its tabu status (*aspiration criterion*). Otherwise, all non-tabu representations are inserted in a priority queue according to the sum of their objective function value and the penalty term induced by the Long Term Memory.

The selection process removes the best representation from the priority queue and accepts it with a predefined probability (0.3), otherwise, the transition selection is repeated (however, the last candidate in the priority queue is always accepted). In the literature, a Tabu Search heuristic having this transition selection scheme is denoted as *Probabilistic Tabu Search*.

End Condition: The first phase of the heuristic ends after a predefined number (3000) of iterations. The second phase ends after $\mathcal{M}_m$ is empty.

Initialization & Multi-start: In the experiments, the OSPF-lengths of the starting representation are initialized with random numbers which are uniformly distributed over the interval $\llcorner[1, \mathrm{OSPF}_{\max}]\lrcorner$. Each search is repeated for a number (3) of times with different initial representations (multi-start runs).

The execution time of the Tabu Search heuristic was measured on a LINUX© operation system using an Intel Celeron© CPU running at 400MHz.

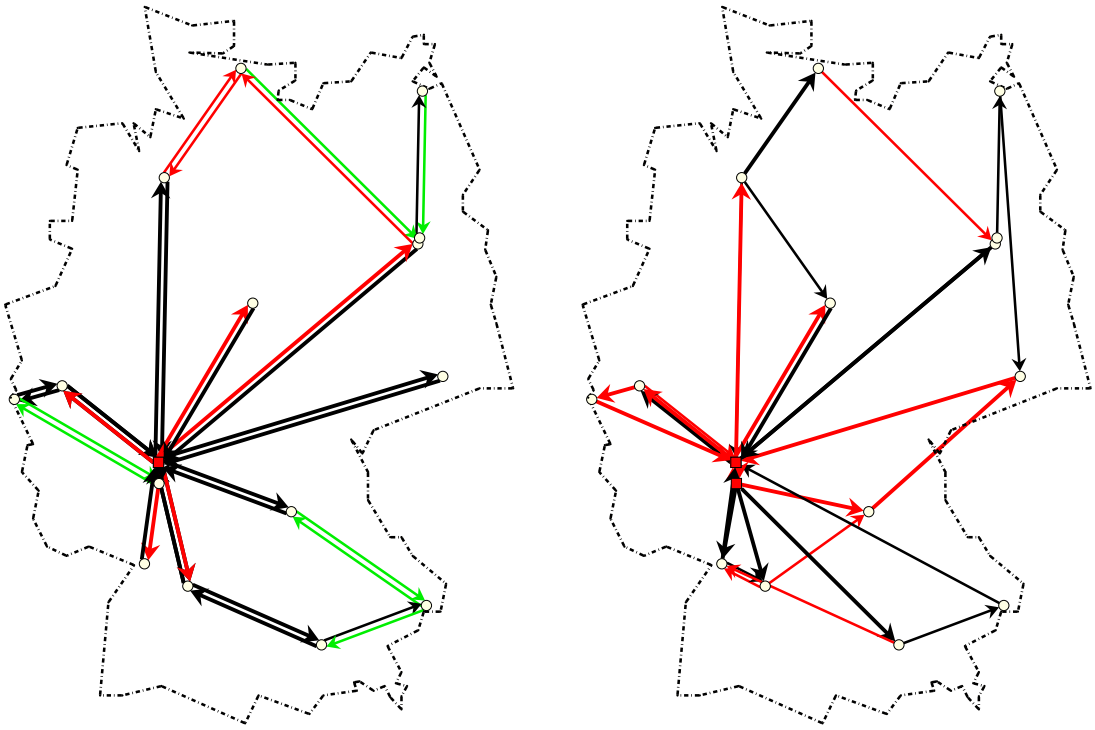

Figure 28: WiN-*SND*: Design with Symmetric (left) and Asymmetric (right) Facilities

6.2.3.4 Tabu Search Result The Tabu Search heuristic finds different network designs for each of the multi-start runs. The (ordered) objective function values of the solutions are:

1: $3.27 \cdot 10^6$
2: $3.30 \cdot 10^6$
3: $3.38 \cdot 10^6$

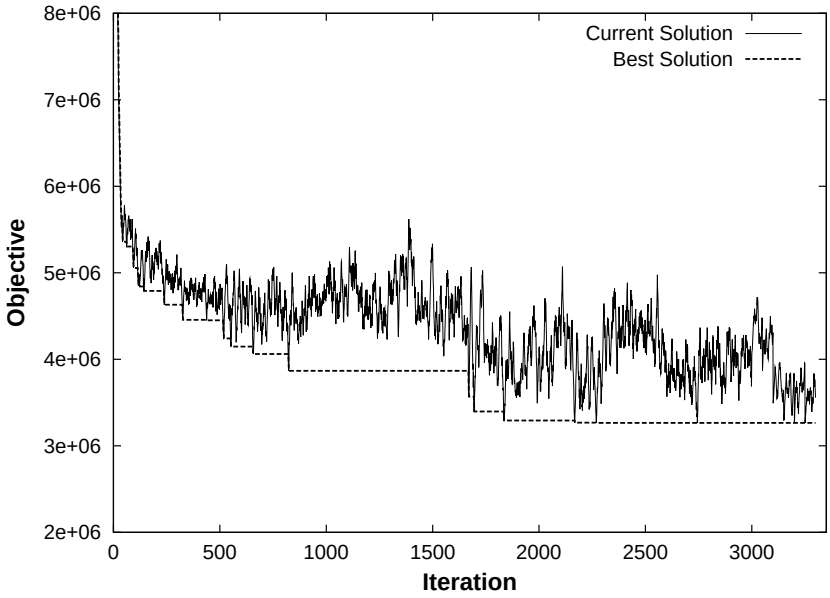

Figure 29: WiN-*SND*: Convergence of the Tabu Search Heuristic for the Best Symmetric Design

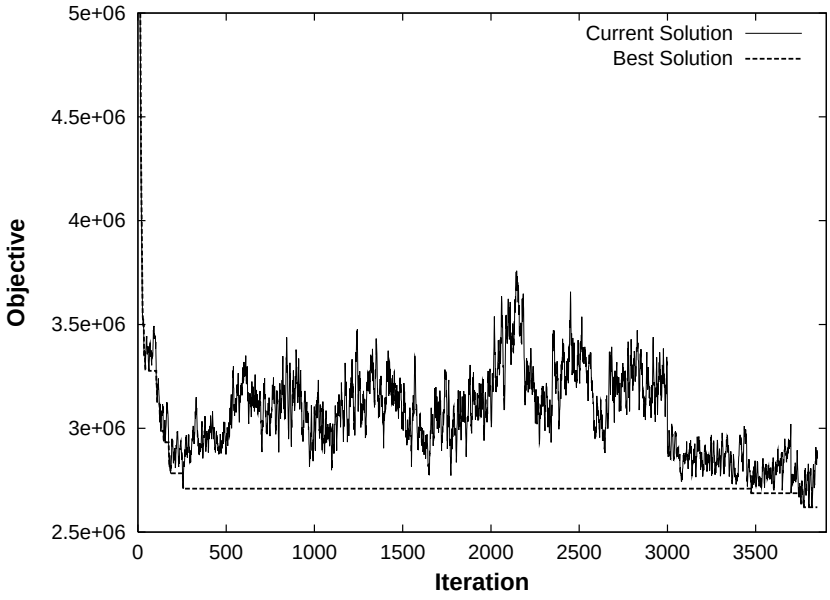

Figure 30: WiN-*SND*: Convergence of the Tabu Search Heuristic for the Best Asymmetric Design

The best two solutions differ by less than one percent. The design of the best solution is shown in Figure[58] 28, the other solutions can be found in Figure 41 (Appendix E.2).

The diagram in Figure 29 illustrates that the overall best solution was found in the first phase after about 2200 iterations, and the second phase (starting with iteration 3000) did not improve it any further. The execution time was about one hour per run.

6.2.3.5 Tabu Search Result (Asymmetric Facilities) In contrast to the suggested cost model, the assumption of symmetric facilities is dropped for once. Any available facility can be

[58]The thickness of the edges in the figures corresponds with the capacity of the deployed facility.

deployed at an edge independent of the deployed facility (if any at all) on the anti-parallel edge.

The solutions which were found by the Tabu Search heuristic have the objective function values:

$$\begin{aligned} 1: &\quad 2.62 \cdot 10^6 \\ 2: &\quad 2.67 \cdot 10^6 \\ 3: &\quad 2.70 \cdot 10^6 \end{aligned}$$

The best solution outperforms the second best by about two percent. The best network design is shown in Figure 28, the other designs can be found in Figure 42 (Appendix E.2).

The diagram in Figure 30 indicates that during the first phase of the search a very good local minimum was found after ca. 300 iterations. This local minimum could not be improved during the remaining iterations of the first phase of the search. However, the *intensification* strategy of the second phase achieved a further improvement. The execution time was about one hour per run.

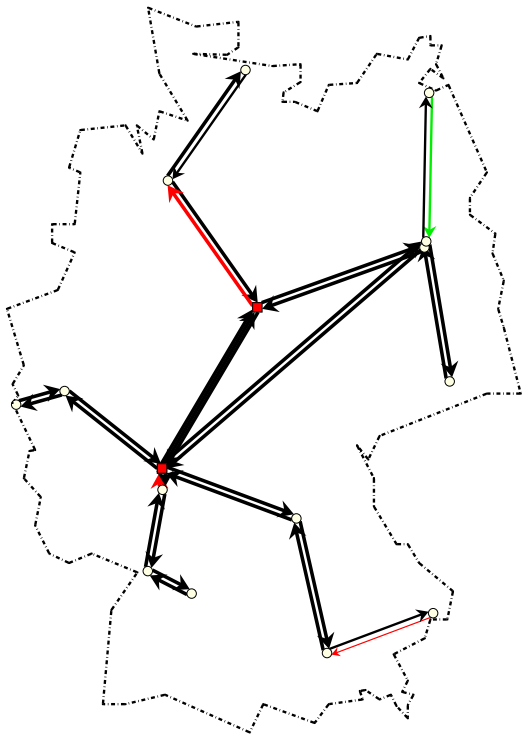

Figure 31: WiN-*SND*: Mentor-II Network Design

6.2.3.6 MENTOR-II design The MENTOR-II heuristic ([Cah98]) is one of the most intensely used constructive search heuristics. It is also part of commercial planning tools (e.g. [Sie98]). It was implemented and tuned for the design of the WiN in [Bes99]. MENTOR-II works with symmetric line capacities.

The best network design produced by MENTOR-II is shown in Figure 31. The costs of this design are

$$3.82 \cdot 10^6$$

which is more than 10% worse than the result of the Tabu Search heuristic. The execution time was about five minutes.

6.2.3.7 Discussion The Tabu Search heuristic with the representation $\mathcal{R}_1$ proved to be a promising solving technique for the static network design problem under consideration. The execution times are not prohibitive and the solutions found are at least credible, especially if compared to the MENTOR-II heuristic that is known to produce sufficiently good solutions.

Furthermore, it is shown that due to the asymmetry of traffic matrix R, a network design which allows the use of asymmetric facilities yields cost savings of more than 20% compared to the symmetric case.

6.2.4 Evolutionary Network Design under Certainty

6.2.4.1 Introduction In the previous section, the design $\mathcal{D}_0$ was calculated by taking into account the requirements R for the observation period T_0 only. Now, the designer also knows that during the observation period

$$T_1 = [\Delta_1, \Delta_2],\ \Delta_2 := \Delta_1 + 12 \text{ months} \tag{25}$$

the development s_2 comes true. As with R, the traffic matrix R_2 is assumed to be static during the observation period T_1. The objective of the *ENDP* is to find a minimum discounted cost network design during the observation period $\llcorner T = [0, \Delta_2] \lrcorner$.

6.2.4.2 Tabu Search Heuristic To solve the *ENDP*, the representation $\mathcal{R}_1$ and the calibration of the Tabu Search heuristic introduced in Section 6.2.3.3 can be taken over without any modifications. The Tabu Search heuristic simply solves two *SNDP*s simultaneously:

- $SNDP_0$: find a design $\mathcal{D}_0$ that is able to transport R during the observation period T_0.
- $SNDP_1$: find a design $\mathcal{D}_1$ that is able to transport R_2 during the observation period T_1.

The designs $\mathcal{D}_0$ and $\mathcal{D}_1$ are linked by a joint objective function. The overall costs of the *END* are calculated by the addition of the costs of the design $\mathcal{D}_0$ during T_0 and the discounted costs of the design $\mathcal{D}_1$ during T_1, taking into account possible upgrade savings. In every iteration step, all transitions in the candidate sets of both *SNDP*s are inserted in a single priority queue, and the next transition (which affects either $\mathcal{D}_0$ or $\mathcal{D}_1$) is selected according to the "transition selection" of Section 6.2.3.3.

The Tabu Search heuristic considers the tradeoff between economy of scale savings of large expansion sizes versus the costs of deploying capacity before it is needed and the costs entailed with the upgrading of existing facilities.

6.2.4.3 Tabu Search Results The results obtained by the best run of the Tabu Search heuristic are shown in the following table:

	END	*IND*
Costs of the initial design $\mathcal{D}_0$	$3.36 \cdot 10^6$	$3.27 \cdot 10^6$
Discounted costs of upgrade design $\mathcal{D}_1$	$5.04 \cdot 10^6$	$4.93 \cdot 10^6$
Overall costs $\mathcal{D}_0$ & $\mathcal{D}_1$	$7.50 \cdot 10^6$	$7.90 \cdot 10^6$
Upgrade savings:	$-0.90 \cdot 10^6$	$-0.30 \cdot 10^6$

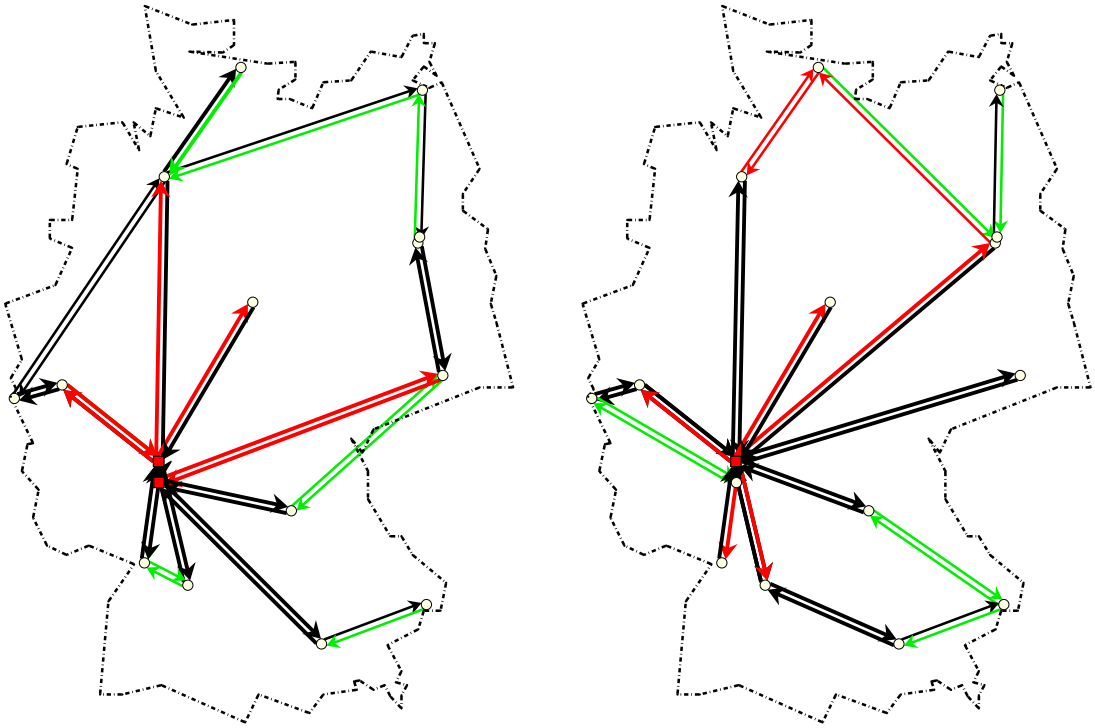

Figure 32: WiN-*END*: Design $\mathcal{D}_0$ with (left) and without (right) Evolutionary Optimization

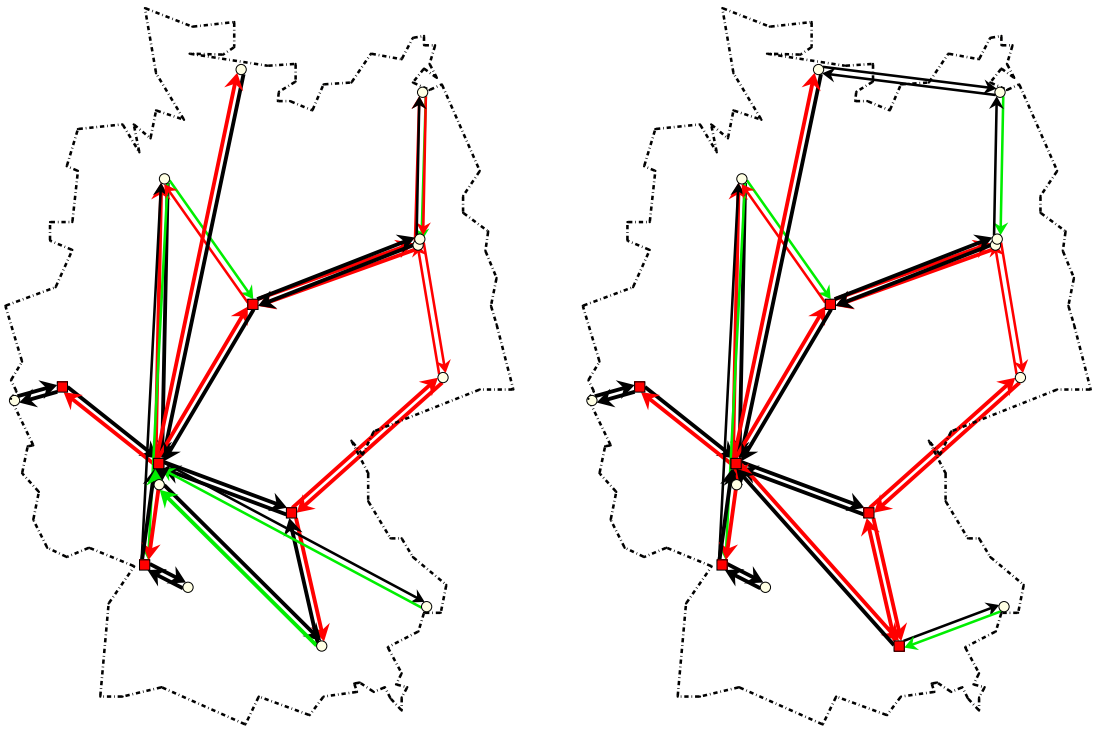

Figure 33: WiN-*END*: Design $\mathcal{D}_1$ with (left) and without (right) Evolutionary Optimization

The last column of the table lists the results for separately (greedily) optimizing and sequentially realizing the network designs $\mathcal{D}_0$ and $\mathcal{D}_1$. This approach corresponds to the **Incremental Network Design** discussed in Section 2.5. The second column of the table contains the result of the joint (evolutionary) optimization of both network designs using the Tabu Search heuristic as described above. With the joint optimization, both designs $\mathcal{D}_0$ and $\mathcal{D}_1$ are slightly more expensive than the optimized separate designs. However, the joint objective yields an improvement of more than 5% compared to the objective that does not take evolutionary aspects into account.

Figure 32 shows the initial design $\mathcal{D}_0$ of the evolutionary network design in comparison with the optimized (static) design. Figure 33 shows the corresponding designs for $\mathcal{D}_1$. The execution time of the Tabu Search heuristic was about two hours per run.

6.2.4.4 Discussion The evolutionary network design improves the overall costs by about 5% which is considerable. The order of magnitude of which cost improvements are possible extremely depends on the cost model, especially on the order of magnitude of setup-, upgrade-, and termination costs. The positive cost effect also increases for shorter upgrade periods.

6.2.5 Network Design under Uncertainty

6.2.5.1 Introduction In the network design under uncertainty, the observation period T_1 with its possible states s_1 and s_2 is regarded only. It is assumed that both states s_1 and s_2 come true with an equal probability.

To illustrate the difference between both states, the optimized static network designs for s_1 ($\mathcal{D}_1^1$) and s_2 ($\mathcal{D}_1^2$) are illustrated in Figure 34. The "worst case design" ($\mathcal{D}_1^3$) that can cope with both states (by assuming an overall traffic growth of 3 for the national and the international traffic[59]) is shown in Figure 36. The costs of the designs are:

$$\begin{aligned} \mathcal{D}_1^1\text{:} &\quad 6.15 \cdot 10^6 \\ \mathcal{D}_1^2\text{:} &\quad 5.17 \cdot 10^6 \\ \mathcal{D}_1^3\text{:} &\quad 6.52 \cdot 10^6 \end{aligned}$$

The task of the designer is to find a network that performs "sufficiently" well with both states. Obviously, the designer could vote for $\mathcal{D}_1^3$. Since both states do not deviate from each other extremely, the worst case design $\mathcal{D}_1^3$ is about 6% more expensive then $\mathcal{D}_1^1$ and about 26% more expensive than $\mathcal{D}_1^2$.

To find a cost efficient design which is able to cope with both states to a high degree, the technique of robust optimization which was introduced in Section 4.5.4 will be used. According to Definition 47, a **robust solution** should be almost optimal for any state and have almost no excess capacity for any realization. The entailed multi-criteria objective has to weigh the discounted costs against the non-feasibility (here in terms of "non-transported" traffic) of a design. The necessary feasibility penalty function is calculated according to Equation 29 which will be introduced in the following section.

[59] Another, more difficult possibility of a "worst case design" would consist in finding a minimum cost network design that uses different routing parameters for the two states and fulfills $\mathcal{Z}_2$. We will see that the result of the robust optimization found here is very close to this.

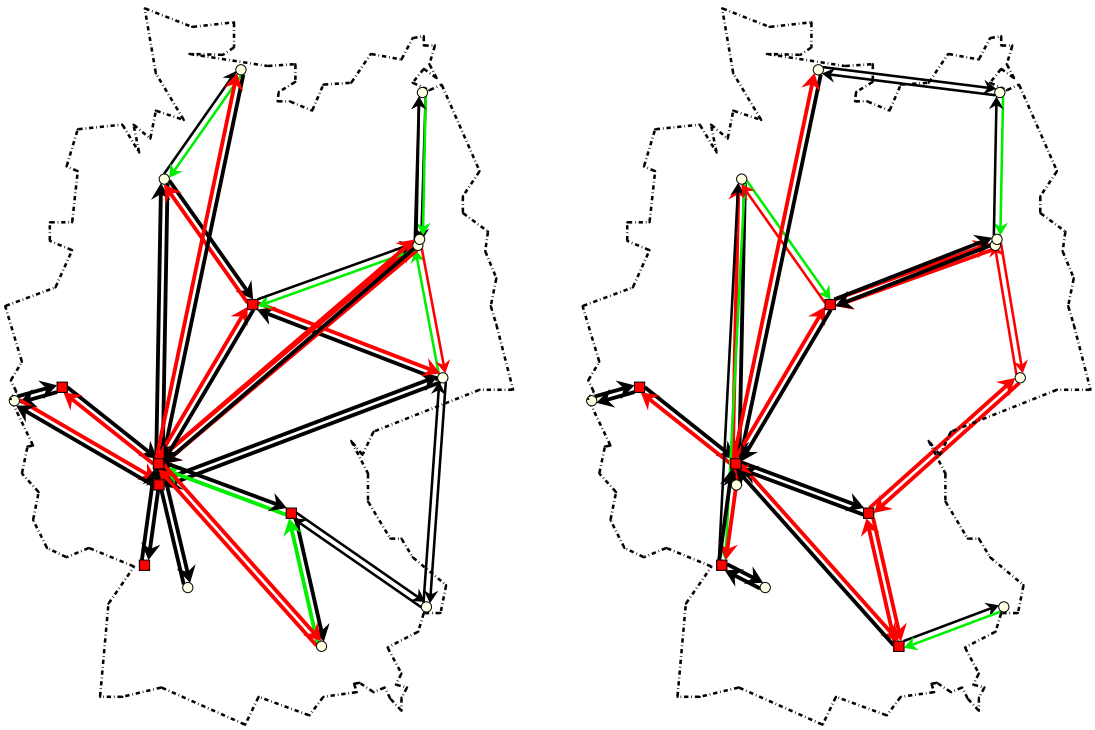

Figure 34: WiN-Robust Network Design: Optimized Static Network Design $\mathcal{D}_1^1$ for R_1 (left) and $\mathcal{D}_1^2$ for R_2 (right)

6.2.5.2 OSPF Network Representation $\mathcal{R}_2$: Capacity Shortage The network representation $\mathcal{R}_1$ is only applicable in situations where no capacity shortage is allowed. However, in the context of a robust optimization it is desirable to accept solutions which are "almost feasible", i.e. violate the constraint $\mathcal{Z}_2$.

Representation $\mathcal{R}_2$ contains for each edge $\llcorner e \in E_S \lrcorner$ both a facility $\llcorner f \in F \cup \{\oslash\} \lrcorner$, as well as an OSPF-length $\llcorner l_s \in [1, \mathrm{OSPF}_{\max}] \lrcorner$ for every possible state s. For notational convenience, a "facility-number" is assigned to each facility-type:

Facility-Number	Facility-Type
'0'	$\oslash$
'1'	34 Mbps
'2'	155 Mbps
'3'	622 Mbps
'4'	2.044 Mbps

An example for a representation is shown in Figure 35 (the thickness of edges corresponds to the capacity of the deployed facility). The representation space for n different facility types and n_s different states can be defined as:

$$S_{\mathcal{R}_2} := \{0, 1, \ldots, n\}^{|E_S|} \times [1, \mathrm{OSPF}_{\max}]^{|E_S| \cdot n_s} \tag{26}$$

Similar to the representation $\mathcal{R}_1$, a candidate set $\hat{\mathcal{N}}_C(v_r)$ is created by modifying each element of $\llcorner \hat{v}_r \in \mathcal{S}_{\mathcal{R}_2} \lrcorner$ (one at a time). The modification[60] consists in increasing (+1) or decreasing (−1)

[60] The modification is only permitted if it produces a valid representation out of $\mathcal{R}_2$!

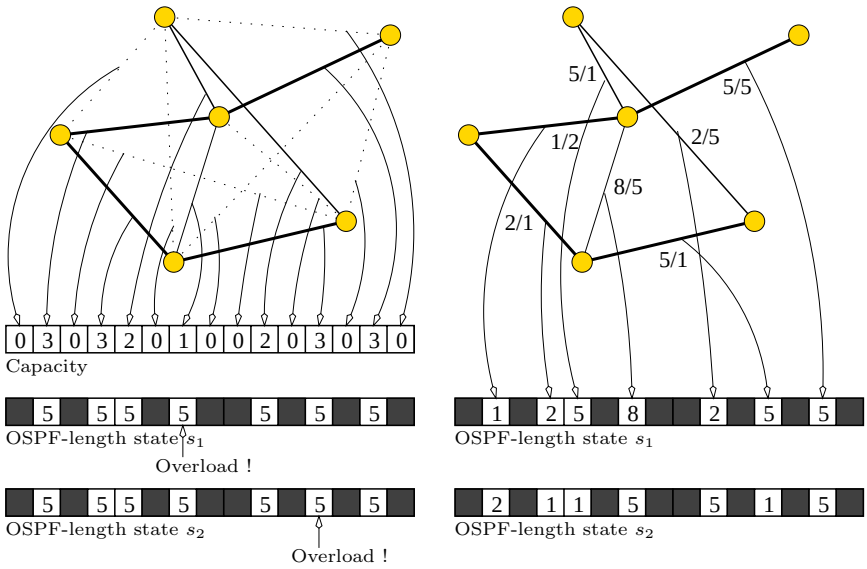

Figure 35: Capacity Network Representation (Undirected Graph)

the number of the facility type (each modification occurring with equal probability). For each representation $\llcorner \hat{v}'_r \in \hat{\mathcal{N}}_C(\hat{v}_r) \lrcorner$, the OSPF-lengths are determined as follows:

1. Let $\llcorner E' \subseteq E_S \lrcorner$ be the set of edges having facility-numbers other than '0'.
2. Let $\llcorner S := \{s_1, \ldots, s_n\} \lrcorner$ be the set of all states.
3. Initialize all OSPF-lengths of each state $\llcorner s \in S \lrcorner$ with the value[61] $\llcorner l_s = \mathrm{OSPF}_{\max}/2 \lrcorner$. This assignment induces a minimum-hop routing (which usually is sub-optimal). Note that for all edges $\llcorner e \in E_S \setminus E' \lrcorner$ the OSPF-lengths are insignificant.
4. Run a Tabu Search heuristic for each state $\llcorner s \in S \lrcorner$ to find the "best" assignment of the OSPF-lengths to the edges in E'. The term "best" refers to the minimization of the following objective function:

$$\hat{\rho}_{\gamma,s}(\hat{v}_r) := \sum_{e \in E'} [\mathrm{Ov}_s(e)]^\gamma \tag{27}$$

The function

$$\mathrm{Ov}_s : E \to \mathbb{R}_0^+ \tag{28}$$

returns the overflow in terms of Mbps that occurs on an edge e after the network is loaded with the requirement matrix of state s according to the OSPF loader.

The total **infeasibility** of a representation $\hat{v}_r$ is defined as:

$$\hat{\rho}_\gamma(\hat{v}_r) := \sum_{s \in S} p_s \cdot \rho_{\gamma,s}(\hat{v}_r) \tag{29}$$

with p_s being the probability of state s to come true

The greater the exponent $\llcorner \gamma \geq 1 \lrcorner$, the less attractive a high overflow on a single edge e becomes, i.e. the tendency is towards accepting a greater number of edges having smaller overflows.

[61] Experiments of an initialization with random numbers proved to be less successful.

The neighborhood of the Tabu Search heuristic is equivalent to the neighborhood of representation $\mathcal{R}_1$. Obviously, the neighborhood can be restricted to the edges $\llcorner e \in E' \lrcorner$. Experiments show that even a small number of iterations (between 4 and 7) result in a near-optimal assignment of OSPF-lengths, in particular if the graph $\llcorner G = (V_S, E') \lrcorner$ is sparse.

The decoding of a representation $D_{\mathcal{R}_2}(\hat{v}_r)$ can easily be achieved by simply assigning the values stored in the representation to the topology $\mathcal{T}_1$. The configuration $\mathcal{C}_1$ has to be chosen depending on which of the states comes true[62].

The complexity of one iteration step depends on the following factors:

- the number of nodes $\llcorner n := |V_S| \lrcorner$ and edges $\llcorner m := |E_S| \lrcorner$ in the supply graph
- the number m_v of realized (i.e. having a facility-number other than '0') facilities in $\hat{v}_r$
- the number n_{ts} of Tabu Search iterations carried out in the OSPF-length optimization process for a single state s
- the number $n_s = |S|$ of different states under consideration

According to Equation (23), a single iteration of the Tabu Search heuristic entailed with $\mathcal{R}_1$ has the complexity:

$$O(m_v^2 \cdot n + m_v \cdot n^2 \cdot \log n) \tag{30}$$

Each of the m neighbors has to execute n_{ts} of those iterations for each state. The overall complexity of one iteration step of the Tabu Search heuristic entailed with the representation $\mathcal{R}_2$ is therefore:

$$O(m \cdot n_s \cdot n_{ts} \cdot m_v^2 \cdot n + m \cdot n_s \cdot n_{ts} \cdot m_v \cdot n^2 \cdot \log n) \tag{31}$$

6.2.5.3 Tabu Search Heuristic Due to the higher complexity of one iteration step of the Tabu Search heuristic, the number of iterations were reduced (200), and the other Tabu Search parameters discussed in Section 6.2.3.3 were scaled accordingly. The Tabu Search heuristic was initialized with the encoding[63] of the worst case design $\mathcal{D}_1^3$.

The (multi-criteria) objective function is given by:

$$\hat{o}(\hat{v}_r) = o_{T_1}(D_{\mathcal{R}_2}(\hat{v}_r)) + \omega \cdot \rho_\gamma(\hat{v}_r) \tag{32}$$

The scaling exponent of the feasibility penalty function was set to $\llcorner \gamma := 1.4 \lrcorner$. The weighing factor between costs and feasibility was set to $\llcorner \omega := 8000 \lrcorner$.

The following optimized results were obtained:

	Costs	Infeasibility given s_1	Infeasibility given s_2
1:	$6.11 \cdot 10^6$	4.7 Mbps	0.0 Mbps
2:	$6.13 \cdot 10^6$	3.9 Mbps	0.0 Mbps
3:	$6.18 \cdot 10^6$	0.9 Mbps	0.0 Mbps

[62] Since in real world problems the development will not be any of the states $\llcorner s \in S \lrcorner$ exactly, the assignment of the OSPF-lengths is in fact the task of the network operation planning *NOP*.

[63] The encoding simply consists in setting the facility-number in the representation to the corresponding facility-number of the deployed facility in $\mathcal{D}_1^3$ for each edge in E. The OSPF-lengths can be ignored.

The **infeasibility** of a design corresponds to the sum of load on every facility that is higher than the capacity of the deployed facility, i.e. the overload. The design of the best solution is shown in Figure 36. The execution time was about four hours per run.

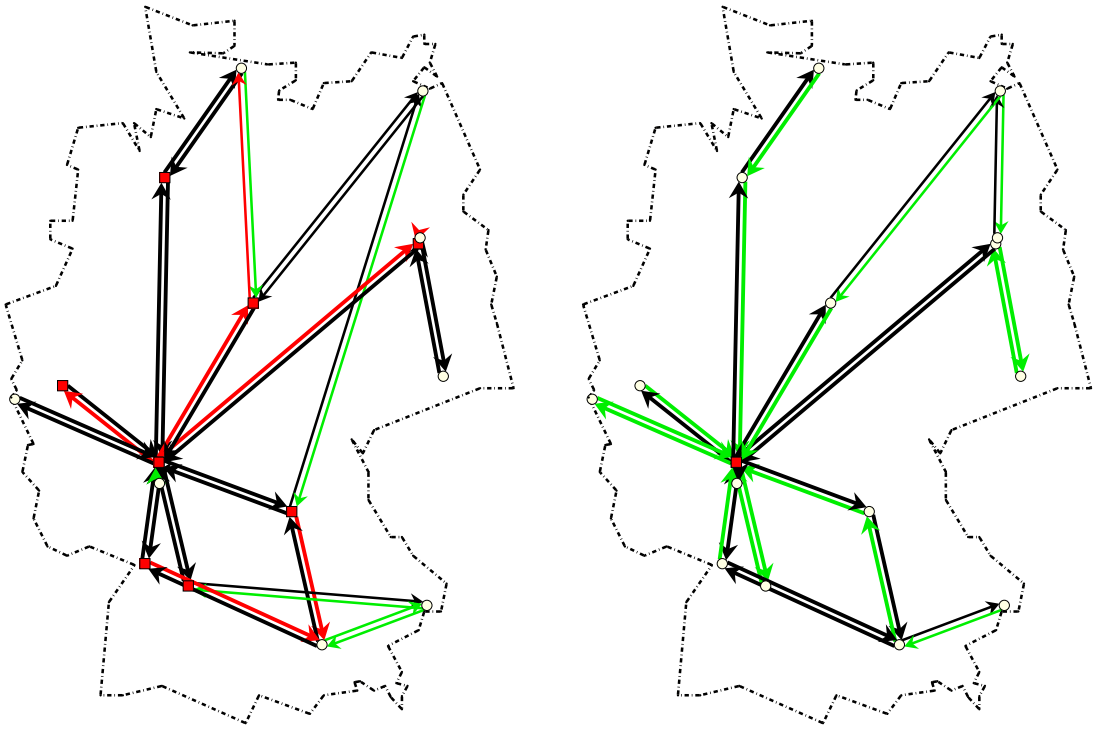

Figure 36: WiN-Robust Network Design: Worst Case Static Design $\mathcal{D}_1^3$ (left) and Robust Design $\mathcal{D}_1$ (right)

6.2.5.4 Discussion All of the designs found are able to transport the traffic requirements R_2 without loss. The less expensive the design becomes the more infeasible it is with respect to R_1. The "best" design found is even less expensive than the best optimized design of s_2.

The characteristics of the results can be influenced by the parameters γ and ω to meet the subjective preferences of the designer. For example, an optimization with smaller ω may result in a network which is less expensive, but infeasible for both states. The values used in the example are set to heavily penalize violations of feasibility. Overflow is only accepted if the entailed gain in terms of cost is highly attractive.

6.2.6 Evolutionary Network Design under Uncertainty

6.2.6.1 Introduction Analogous to the *ENDP* under certainty in Section 6.2.4, the task of the *ENDP* under uncertainty is to improve the initial static network design $\mathcal{D}_0$ (deployed during T_0) taking into account the future traffic developments (i.e. the potential network designs $\mathcal{D}_1$ deployed during T_1). Again, the requirement matrix R is assumed to be static during T_0. The traffic development during T_1 stays uncertain: Each of the possible states s_1 and s_2 can come true with equal probability.

Here the author proposes the following course of action, which solves the problem in two stages:

1. A robust network design $\mathcal{D}_1$ for the observation period T_1 is calculated. This design is identical with the network design under uncertainty introduced in the previous section.

2. The design $\mathcal{D}_0$ is optimized against the *fixed* design $\mathcal{D}_1$, i.e. the design $\mathcal{D}_0$ is modified in such a way that the discounted costs during the observation period T are optimized.

The reasons for this approach are twofold:

- First of all, the simultaneous optimization taking into account of all degrees of freedom is computationally intractable (at least if using the representations which where introduced so far).
- The uncertainties connected with a design have to be resolved in a way that the solution of the problem stays "credible". For example, it does not make sense to solve an *ENDP* having two stages, T_0 and T_1, where both stages are highly uncertain. Depending on the grade of uncertainty, the author would either solve a robust single staged network design problem under uncertainty (ignoring the second stage T_1), or rely on the notion of flexibility engineering (see Section 4.5.7) and deploy a flexible network design that can be adapted cost efficiently to changing situations.

6.2.6.2 Tabu Search Heuristic To solve the *ENDP*, the representation $\mathcal{R}_1$ and the calibration of the Tabu Search heuristic introduced in Section 6.2.3.3 can be taken over without any modifications. The Tabu Search heuristic has to solve only one *SNDP*: "Find a design $\mathcal{D}_0$ that is able to transport the traffic matrix R."

The designs $\mathcal{D}_0$ and $\mathcal{D}_1$ are linked by a joint objective function. The overall costs of the *END* are calculated by the sum of the costs of the design $\mathcal{D}_0$ during T_0 and the discounted costs of the design $\mathcal{D}_1$ during T_1, taking into account possible upgrade savings. In every iteration step the candidate set consists of modifications of the design $\mathcal{D}_0$ only.

Again, the Tabu Search heuristic considers the tradeoff between economy of scale savings of large expansion sizes versus the costs of deploying capacity before it is needed and the costs entailed with the upgrading of existing facilities. Since the design $\mathcal{D}_1$ is fixed, the degrees of freedom are pruned to the design $\llcorner\mathcal{V} := \mathcal{D}_0\lrcorner$.

6.2.6.3 Tabu Search Results In the following, the second best design $\mathcal{D}_1$ (illustrated in Figure 37) of Section 6.2.5 is chosen to be deployed during T_1 because it is only insignificantly more expensive than the best design, but is more feasible:

	Costs	Infeasibility given s_1	Infeasibility given s_2
2:	$6.13 \cdot 10^6$	3.9 Mbps	0.0 Mbps

The optimization of $\mathcal{D}_0$ yields the following results:

	END	*IND*
Costs of the initial design $\mathcal{D}_0$	$3.30 \cdot 10^6$	$3.27 \cdot 10^6$
Discounted costs of upgrade design $\mathcal{D}_1$	$5.83 \cdot 10^6$	$5.83 \cdot 10^6$
Overall costs $\mathcal{D}_0$ & $\mathcal{D}_1$	$8.14 \cdot 10^6$	$8.51 \cdot 10^6$
Upgrade savings:	$-0.99 \cdot 10^6$	$-0.59 \cdot 10^6$

The network costs of the initial design $\mathcal{D}_0$ of the joint (evolutionary) optimization is only insignificantly higher than the costs of the independent static network design. The overall savings possible with the evolutionary network design in comparison to the incremental design are as high as 5%. Figure 37 shows the resulting initial network design $\mathcal{D}_0$. The execution time of the heuristic was about one hour per run.

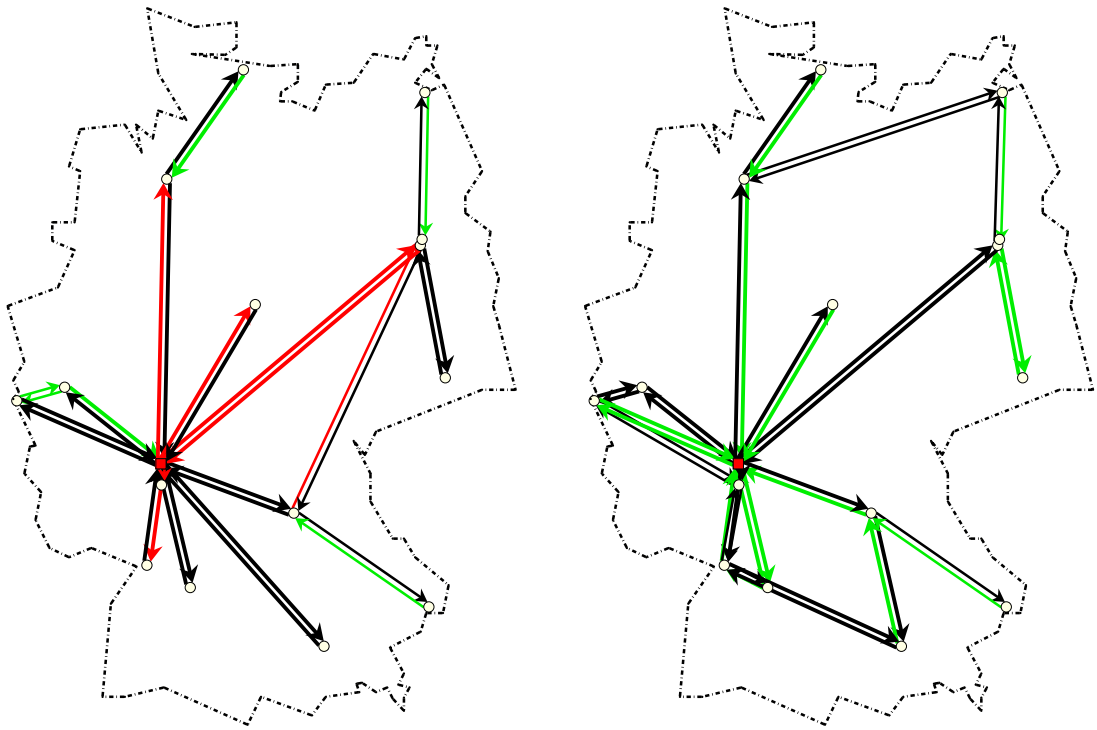

Figure 37: WiN-Robust *END*: Network Design $\mathcal{D}_0$ (left) that was Optimized against the Robust Design $\mathcal{D}_1$ (right)

6.2.6.4 Discussion The savings which are possible by the joint optimization are significant[64] (5%). However, the total amount of savings that can in fact be realized depends on the "real" traffic development which stays uncertain. Of course, the designer should prefer a design that offers the opportunity of cost savings on the long run.

6.2.7 Final Remarks

Concluding, the author wants to make the following remarks about his (subjective) observations which are common to all problems and solutions, respectively:

- The Tabu Search heuristic "converged" for all problems. All tests confirmed that the property to find "good" solutions is largely independent of the initial starting point of the search. The relative difference in terms of costs between the best and the worst solution found is rather small, which can be interpreted as an indicator for the stability of the search heuristic, as well as a consequence of the structural nature of the state space graph (see Section 6.1.2).
- The heuristic is of polynomial time complexity if a pre-defined maximum number of iterations is given.
- The topologies of the solutions found can differ considerably, albeit the difference in costs is small. It seems that there is a set of edges that can be found in every optimized design,

[64] The optimization against the first and the third best design of Section 6.2.5 did yield savings of 5% and 6%, respectively.

but except for this set there are a lot of almost equally expensive possibilities of how to arrange the remaining edges. The motivation given in Section 6.1.2 can be confirmed.

- The parameters of the Tabu Search heuristic are set by common sense and guided by some statistical testing. Systematically fine-tuning these parameters (e.g. by more statistical tests) can further improve the performance.
- The settings of the environment were *not* chosen towards achieving particular good results, but designed to reflect the reality to a high degree. Tests have shown that with other cost models the results become even more impressive. Of course, the cost model is an external influence factor that evades the sphere of the designer's influence.

7 Conclusion

This final chapter first summarizes the results obtained in the thesis, and then gives an outlook on possible directions of future research.

7.1 Summary of Results

Due to the recent developments in telecommunication there is still a high need for network design algorithms that are able to incorporate new environments and constraints. One of the most pressing problems is to cope with the uncertainty about the future (e.g. about the traffic and costs) and the variability of data traffic patterns. This thesis formulates three different classes of network design problems that take into account variability on various scales:

- The *Static Network Design Problem* (*SND*-Problem) does not consider any variability.
- The *Incremental Network Design Problem* (*IND*-Problem) tries to greedily adapt an existing network design to a changed environment.
- The *Evolutionary Network Design Problem* (*END*-Problem) is formulated to profit from knowledge of future developments explicitly.

The common goal of all problem classes consists of finding a cost efficient network design at the beginning of a new planning period.

To gain deeper insights into the problems, the influencing factors, i.e. the "*dimensions of network design*", were discussed and assessed. A decision theoretic framework is proposed that structures the problem in an uniform way and provides guidelines on how to proceed in a given setting.

Whenever projections of the future become necessary, *planning under incomplete knowledge* has to be a major concern. Due to its significance, a real-world design that ignores the influence of uncertainty is of limited use. Especially in the context of data network design, the incorporation of uncertainty is of utmost importance. The thesis discusses the uncertainties arising in network design problems. Each uncertainty is assessed with respect to its degree of influence upon the problem. Furthermore, an appropriate formalism to describe each uncertainty and a way to incorporate it into the design process is proposed.

The incorporation of uncertainty in mathematical programming is an ongoing field of research. A number of techniques are assessed with respect to their suitability in the context of network design problems. Due to the structure of the underlying problems, the well-known technique of *post-optimal sensitivity analysis* is not applicable and the technique of *stochastic programming* results in an intractable complexity. The most promising alternatives are identified as the technique of *stochastic decision trees* (which are structurally very similar to *END*-Problems) and the technique of *robust optimization*.

In the case of a highly uncertain planning environment, none of the common mathematical optimization techniques can be recommended. To allow a reasonable network design even in such environments, the notion of *flexibility engineering* in network design is introduced and discussed. However, the measure of flexibility evades an exact mathematical formulation, since it is problem-dependent and makes heavy use of the capabilities of the designer to find a set of appropriate evaluation criteria.

Moreover, a new conceptual object-oriented software framework for network design problems, *FooNet*, is presented. The main characteristics of *FooNet* are its three-layer *code* and *design reuse* hierarchy that enables its components to serve as main building blocks for almost arbitrary network design algorithms.

The last part of the thesis exemplarily applies the theoretical concepts to a realistic setting: the backbone network design task of the German Wissenschaftsnetz. The set of implemented algorithms is based on the concept of meta-heuristic optimization. Meta-heuristics in general and Tabu Search in particular are motivated to be promising solution techniques for network design problems. Then, for each problem class a *polynomial-time Tabu Search algorithm* is presented. For the given example it could be shown that the network designs found by the Meta-heuristic search outperform the "classical" design algorithms with respect to the objective function (i.e. discounted costs):

- The static network design found by the Tabu Search heuristic outperforms MENTOR-II, one of the most intensively used constructive search heuristic.
- The evolutionary network design that considers the tradeoff between economy of scale savings versus the costs of installing capacity before it is needed and the costs entailed with the upgrading of existing facilities outperforms the incremental network design.
- The concept of a "robust network design" that is a realization of the mathematical technique of robust optimization is shown to be superior to the "worst case network design".
- Finally, an algorithm that finds a cost efficient initial design in presence of an uncertain future traffic development is presented.

7.2 Outlook and Future Work

There are a number of promising directions for future research.

To continue the work of Chapter 6, the following investigations are possible:

- It would be interesting to see how the introduced concepts behave with respect to *other search heuristics* (e.g. simulated annealing). Since the representations can be taken over with only few modifications, this can be accomplished rather easily. Work is already in progress and the results will be published in [Lor00].
- The connection between the characteristics of the state space graph and the concept of *adaptive memory programming* is another field of possible research. A deeper insight may help to develop even better search heuristics and give (at least statistical) evidence on the quality of the solution found.
- Obviously, the incorporation of further *constraints* (e.g. survivability) is a possible extention to the algorithms presented.
- The design of the access network was fixed in the exemplary settings. However, the concepts introduced can be applied to designing *access networks* as well. Furthermore, it would pay off to regard access and backbone network design simultaneously. However, due to the state space explosion it is unclear to what extent this is possible. For a successful approach, the development of "smarter" representations and neighborhoods may be necessary.
- Even if the execution times are not computationally prohibitive, they can be improved by orders of magnitude by the use of *approximation algorithms* to speed up the calculation of the objective function (see property A_3 in Section 6.1.1). This would allow to use the

design algorithms in an inner loop and incorporate the *upgrade times* as a further degree of freedom.

Another necessity of future development of network design algorithms is the introduction of a generalized set of planning problems, i.e. a *benchmark*. In other fields of science accepted and comprehensive sets of problems[65] are available. A benchmark would allow a more systematic comparison of the available algorithms with respect to their performance and applicability.

The incorporation of multiple traffic classes[66] in a network design adds a further dimension to the planning task. Section 2.7.1 deals with the problem of how a suitable traffic matrix in terms of required bandwidth between the end-nodes can be obtained. Although the field of data traffic modeling is in rapid development, this works sufficiently well with classical IP traffic that uses the TCP windowing mechanism and reacts to congestion by throttling throughput. However, the increasing demand for Quality of Services (QoS) on the Internet burdens new requirements upon the network. Even a slightly congested IP network may result in a total loss of the QoS-sensitive services. As a solution, the IETF [↑] proposes the concepts of Integrated Services [↑] and Differentiated Services [↑] (see e.g. [RFC97, RFC98c]) as an extention to the classical (best-effort) IP service. Entailed with the underlying protocols is the ability of *traffic engineering* [↑], where different *classes of traffic* can be regarded separately (e.g. routed on different paths or scheduled with different priority). In this case, the algorithms have to take into account the tasks and abilities of *network management* to a much higher degree. It also remains to be studied to what degree the results of ATM network planning, which has to deal with similar problems, can be adapted to IP networks[67].

The emergence of new concepts (such as value added networks [↑VAN]) that offer various service levels at different costs will introduce new opportunities to the planner. To give an example, consider the following setting: The network designer can choose from different priorities and capacities (i.e. Service Level Agreements) at which the traffic can be transported over an edge facility. If the transported traffic violates the Service Level Agreement, it may either be transported at *higher costs*, or additional capacity may be rented (for a short period) at usage-dependent costs (*dial-up services*). The trade-off between rented and dial-up services is already known from the planning of telephone services, albeit is a new (and not yet investigated) option in data networks. The ability to rent additional services on short notice diminishes the need for over-provisioning of facilities, and therefore can help to build more cost efficient networks. This conforms the notion of *flexibility* introduced in Section 4.5.7.2. However, it is also entailed with a higher "upgrading activity", e.g. whenever a change in the traffic requirements is statistically significant. This emphasizes the necessity for algorithms that take developments over time into account, i.e. *evolutionary design*, to yield cost efficient designs.

More than one generation of researchers has dedicated itself to the problem of telecommunication networks design. It seems that there are still enough open problems to occupy a future generation, too.

[65] One could take the field of theorem proving (see [SS00]) as an example.
[66] which usually also have different statistical properties
[67] e.g. concerning Effective Bandwidth formula or the PNNI routing protocol

List of Symbols

Common Symbols

[↑]	"see glossary of terms"
i), ii), ...	enumeration in Roman numbers
✓	"OK"
↯	"false"
?	"don't know" or "don't care"
©	copyright or trademark
$\llcorner \cdot \lrcorner$	mathematical grouping operator in the text (used for legibility)

Mathematical Symbols

$:=$	the left side "is defined by" the right side
$\simeq$	the left side "is approximately equal" to the right side
i, j	arbitrary index counters
x, y	arbitrary variables
$.^\top$	the transpose operator on a vector or matrix
$\lvert \cdot \rvert$	the absolute value operator or the number of elements in a set
$\lceil x \rceil$	the smallest real number which is greater than x
$\times$	the Cartesian product of sets
$\wp(A)$	the set of all subsets of the set A, i.e. the power set of A
$\{\ldots\}$	an unordered set
$(\ldots)$	an ordered set
$\emptyset$	the empty set, i.e. $\{\}$
Ω	the universal set
$\in$	"is element of"
$\subseteq$	"is subset or equal"
$\cup$	the union of sets operator
$\cap$	the intersection of sets operator
$\setminus$	the difference of sets operator
$\sum$	the summation operator or the union of disjoint sets operator
$\forall$	"for all"
$\exists$	"there exists"
$\vee$	the logical "or"
$\wedge$	the logical "and"
$x : y$	x "such that" y
$\min \cdot$	the minimum operator
$\max \cdot$	the maximum operator
%	percentage symbol

$p(A)$	the probability of the set of events A
q	a random variable
$\overline{q} = E(q)$	the expected value of q (supposing it exists)
$\sigma^2(q)$	the variance of q (supposing it exists), i.e. $E(q^2) - E^2(q)$
$x \mid y$	x "under the condition" y
$[x, y)$	a half-open interval ($x < y$)
$[x, y]$	a closed interval ($x \leq y$)
$a[x]$	the xth element of the vector a
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^+$	the set of positive real numbers
$\mathbb{R}_0^+$	the set of positive real numbers including 0
$\mathbb{B}$	the set of boolean values, i.e. the set {"true", "false"}
$\mathbb{N}$	the set of natural numbers, i.e. the set $\{1, 2, 3, \ldots\}$
$\mathcal{B}$	the Borel set, i.e. all measurable subsets of $\mathbb{R}$
$\mathcal{B}_0^+$	the set of all measurable subsets of $\mathbb{R}_0^+$
$\succ$	a (weak or strict) total order, i.e. a comparable and transitive relation
$\rhd$	a (weak or strict) partial order, i.e. a asymmetric and transitive relation
o	objective function (Sec. 2)
$\mathcal{P}$	input data or input parameters of a model (Def. 2)
$\mathcal{V}$	set of variables (Def. 2)
$\mathcal{Z}$	set of constraints imposed on a model (Def. 2)
$S_{\mathcal{V}}$	solution space, i.e. all feasible assignments to the variables $\mathcal{V}$ (Def. 2)
c_i	objective criterion number i (Sec. 2.7.8)
w_i	relative weight of criterion c_i (Sec. 2.7.8)
c	total number of criteria (Sec. 2.7.8)

Algorithmic Symbols

n	problem size or maximum index
$\mathcal{NP}$	Non-Deterministic Polynomial
δ	approximation factor (Sec.A.6)
β	lower bound, i.e. the solution is at most β-times the optimum
$O(\cdot)$	complexity measure, i.e. $f, g : \mathbb{N} \to \mathbb{R}$ then $f = O(g)$ if $\exists n_0 \in \mathbb{N} \wedge \exists c \in \mathbb{R}^+$ such that $\forall n \in \mathbb{N} \wedge n > n_0 \ : \ f(n) \leq c \cdot g(n)$
$\mathsf{A}(\cdot)$	mathematical optimization process (Def. 54)
P_i	properties of large scale systems, $i \in \{1, 2, 3\}$ (Sec. 6.1.1)
A_i	properties of optimization algorithms, $i \in \{1, 2, 3\}$ (Sec. 6.1.1)
$G_T = (V_T, E_T)$	a decision tree (Def. 38)
P_T	a complete path in a decision tree (Def. 38)
$\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$	the state space graph (Def. 54)
v	a node in the state space graph

$\mathcal{R}, \mathcal{R}_1, \mathcal{R}_2$	representations (Def. 56)
v_r	a single representation
$\mathcal{N}$	the neighborhood operator (Def. 55)
$\mathcal{N}_C$	candidate list (Def. 55)
$S_\mathcal{R}$	the space of all possible representations (Def. 56)
$E_\mathcal{R}$	an encoding function (Def. 56)
$D_\mathcal{R}$	a decoding function (Def. 56)
$\mathcal{M}_s, \mathcal{M}_m, \mathcal{M}_l$	tabu memories (Sec. 6.1.3.4)

Symbols used in Network Design

R	the requirement matrix (Def. 8)
r_t	the number of different traffic classes (Def. 8)
V_e	the set of end-nodes (Def. 8)
V_b	the set of (potential) backbone-nodes (Def. 8)
C, C^r, C^{nr}, C^∇	cost structures (Def. 8, Def. 11)
F	the set of available facilities (Def. 6)
$\mathcal{T}$	the topology of network elements (Def. 6)
$\mathcal{C}$	the configuration of network elements (Def. 6)
$\mathcal{D} = (\mathcal{T}, \mathcal{C})$	a network design (Def. 6)
$F_\mathcal{D}$	the set of facilities used in design $\mathcal{D}$ (Def. 6)
$\mathcal{C}_f$	the configuration of the facility f (Def. 6)
T, T_0, T_1	observation periods (Def. 9)
$\llcorner t, t_0, t_{\max}\lrcorner$	observation times
$\cdot^T$	a variable that changes with time T
$\llcorner\Delta, \Delta_i\lrcorner \in T$	elements of the observation period
∇, ∇_i	the set of changes (Def. 12, Def. 21)
$\oslash$	no facility is present (Def. 11)
$\mathcal{E}$	the design environment (Def. 28)
$\mathcal{S}$	a design scenario (Def. 31)
$\mathcal{F}$	the framework operator (Sec. 3.2)
$\mathcal{A}^+$	the set of feasible solutions produced by a solver (Def. 33)
$\mathcal{A}^*$	the set of optimal solutions (Def. 33)
s_i	state s_i out of a number of possible future states
r	the rate of return (App. C)
$\mathrm{PV}(\cdot,\cdot)$	the present value function (Eq. (48))
$\mathrm{PVR}(\cdot,\cdot,\cdot)$	the discounted reoccurring costs function (Eq. (49))
α	the Economy of Scale [↑] parameter (App. C)
$G_S = (V_S, E_S)$	a supply graph (Def. 57)
$\mathrm{OSPF}_{\max}$	the maximal OSPF-length (Sec. 6.2.3)
$\mathrm{MOD}_{\max}$	the maximal OSPF-length modification (Sec. 6.2.3)

γ	the overload scaling exponent (Sec. 6.2.5.2)
ρ	a feasibility penalty function (Def. 47, Eq. (29))

A Survey of Existing Network Design Algorithms

A.1 Introduction

Known network design algorithms depend heavily on their modeling assumptions (e.g. about traffic and costs) and on the chosen level of detail. In the following, a short subjective[68] classification of network design algorithms is presented that can be used to solve either the overall design problem, or only sub-problems of it.

A.2 Manual Planning

An expert in network design is likely to produce reasonable results in very short time and without sophisticated modeling and (expensive) tools. Manual planning methods (*MaPl*) can produce almost optimal results for small-sized networks, and almost any constraint can be incorporated in the design.

A.3 Constructive Heuristics

Constructive Heuristics (*HE*) are used most intensely in network design algorithms. In general, constructive heuristics have the following characteristics:

- they are easy to understand
- they can be adapted to changing requirements with less effort
- they produce feasible and "sufficiently good" solutions
- they have acceptable time complexity
- they allow interaction with the designer
- they usually give no lower bounds on the optimum solution

Center of Mass [Ker93], Unified-Algorithm [ACM81], Cut Saturation [BF77] and MENTOR [KKG89, Cah98] are examples of well-known and intensely used heuristics in network design.

A.4 Analytical Models

In some cases, *analytical models* (*AM*) can be applied to network design problems. These analytical models, however, are known to be hard to handle and often rely on assumptions like "independence of events" or "system in steady-state" that do not hold in practice. Sometimes, approximation or relaxation (see e.g. cost approximation in Figure 38) can transform a discrete problem into a continuous one that can be tackled by analytical models. Flow deviation (see e.g. [AMO93]) and queuing theory (see e.g. [Kle75]) are examples of analytical models.

In general, network design problems belong to the field of combinatorial optimization and cannot completely be described by analytical models.

[68] Other classifications guided by different viewpoints are possible.

A.5 Mathematical Programming

Mathematical Programming [↑MP] is the core of Operations Research. Almost any network design problem can be formulated as a mathematical program [↑]. The interested reader may consult [Tah87, Dom95, KW97] for an overview.

In the following sections, a short overview from the perspective of network design will be given.

A.5.1 Linear Programming

Linear Programming (*LP*) provides solvers for the following type of model:

$$\begin{gathered}\text{minimize } o : \mathbb{R}^n \to \mathbb{R} \quad x \to c^\top x \text{ subject to } Ax \geq b \\ c \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ A \in \mathbb{R}^{(m \times n)},\ \llcorner n, m \in \mathbb{N} \lrcorner \end{gathered} \tag{33}$$

If there are non-linear constraints, they have to be transformed into linear approximations, e.g. demand constraints or Economies of Scale [↑]. Known solvers are the *simplex algorithm* or *Karmarkar's interior point method*. *LP* is known to be of polynomial time complexity, the simplex algorithm, however, has exponential worst case behavior but performs well in the average case. An example of a network design problem modeled as a linear program and enforcing integrality conditions by Branch & Bound (see next section) is given in [Gav92].

A.5.2 Integer Linear Programming or Combinatorial Optimization

Whenever there are integer conditions imposed on the vector x of equation (33), i.e. $\llcorner x_i \in \mathbb{N}_0 \lrcorner$, the *LP* becomes an *Integer Linear Program* (*ILP*). Some authors use the term *Mixed ILP* if integrality conditions are only imposed on some variables, but here both types of problems will be denoted as *ILP*. *ILP* problems are known to be strongly $\mathcal{NP}$-complete in general. Unfortunately, most network design problems fall into this category. The decision whether a facility should be deployed or not is an example of such an integer condition in a model. Solvers for *ILP* problems are e.g. *Branch & Bound* [↑], which is an implicit enumerative strategy that uses relaxation to *prune* unprofitable solutions, or *Branch & Cut* [↑] that combines Branch & Bound with *cutting plane* algorithms (see e.g. [NW88]). A whole branch of research, *Combinatorial Optimization*, is devoted to the theory for solving problems with integrality conditions. Additional material on this topic can be found in [PS82, NW88].

In the future, techniques that combine heuristics with *ILP* are expected to produce very good results in network design but due to their complexity are limited to medium sized problems. An example can be found in [BGW97].

A.5.3 Dynamic Programming

Dynamic programming is a computational method which uses recursion to solve the problem in stages. A problem is decomposed into a sequence of nested sub-problems, and the solution of one subproblem is derived from the solution of the preceding subproblem. In this sequentially dependent process, each stage involves an optimization over one variable only. Preceding decisions have to be independent.

Dynamic programming applies to some subproblems of network design, e.g. finding the cheapest realization of a certain capacity requirement given a set of discrete capacities [GK77]. In some applications, dynamic programming is applied to multi-stage planning problems that are discussed in Section 3.5.1.

A.5.4 Meta-Heuristic Optimization

Due to the intractable complexity of optimal solvers for Combinatorial Optimization problems, the field of *Meta-Heuristic Optimization* (*MHO*) has developed. Unlike the constructive heuristics of Appendix A.3, *MHO*-algorithms are not restricted to one specific problem in general. The applicability of *MHO*-algorithms to an arbitrary optimization problem depends on coding the problem in an appropriate representation.

Many of the *MHO*-algorithms are motivated by analogies in nature, e.g. *Evolutionary Algorithms*, *Simulated Annealing* or *Tabu Search*. In contrast to *ILP*-solvers like Branch & Bound, *MHO*-algorithms do not guarantee optimal solutions, but are known to be near optimal under certain conditions.

Due to their eminent importance, *Randomized Search Techniques* (*RST*) as a special case of *MHO* are regarded separately. Examples of *RST* are Simulated Annealing, Genetic Algorithms, and Randomized Tabu Search.

General characteristics of *RST* are:

- *RST* are likely to produce near optimal solutions
- *RST* can be tuned to produce very fast solutions[69] even for large problem sizes
- *RST* have difficulties with incorporating arbitrary constraints

A successful application of Simulated Annealing and Genetic Algorithms to network design problems is presented in [Sha97].

A.6 Approximation Algorithms

The theory of *Approximation Algorithms* (*AA*s) is a comparatively young field in theoretical computer science. It is concerned with finding good algorithms, i.e. of polynomial time complexity, that yield (sub-optimal) solutions to $\mathcal{NP}$-complete problems, which are bounded by being at most $\delta(n)$ times worse than the optimum solution (see e.g. [Hoc95] for an introduction). $\delta(n)$ may be a constant or depend on the problem size n.

The theory of *AA*s reveals an interesting structural classification of network design problems of size $\llcorner n := |V_e| \lrcorner$:

- no reasonable polynomial time approximation algorithm exists, i.e. $\delta(n)$ grows faster than polynomial with n
- a $\delta(n)$-*AA* exists and $\llcorner \delta(n) = O(n^x) \lrcorner$ with $\llcorner x \in \mathbb{R}^+ \lrcorner$
- a $\delta(n)$-*AA* exists and $\llcorner \delta(n) = O(\log n) \lrcorner$
- a $\delta(n)$-*AA* exists and $\llcorner \delta(n) = c,\ c \in \mathbb{R}^+, c > 1 \lrcorner$ is independent of n
- a polynomial time approximation scheme exists, i.e. for every $\llcorner \delta > 1 \lrcorner$ there exists a polynomial time approximation algorithm.

A promising application of approximation algorithms may be the generation of lower bounds.

An example of an $O(\log(|V_e|))$-approximation algorithm solving a minimum-cost network design problem is presented in [MP94], some further approximation algorithms for subproblems of network design can be found in [Kru96].

[69] especially compared to *ILP* solvers like Branch & Bound

A.7 Artificial Intelligence

Network designers have to be experts in their field. The knowledge of an expert is valuable in many situations of the network design process. Knowledge processing and engineering belong to the field of Artificial Intelligence (*AI*) (for an overview see e.g. [DAA95, KW98]).

As already mentioned, manual planning by "rules of thumb" is one way of solving network design problems. Rule or case based inference systems (see e.g. [HP98]) using expert knowledge can serve as a decision support system, offering assistance to the network designer. Decision support systems in general provide assistance by provision of relevant information, reasoning with information to provide arguments for possible decisions, and identifying qualifications, ramifications or risk associated with possible decisions. A rule based expert system for network design problems can be found in [FD95], an example of a more general inference system based on probability theory is described in [FS97].

Neuronal Networks that are capable of forecasting water and power consumption should be able to forecast network traffic as well, but there is still some additional research necessary (see e.g. [HSV+99]).

(*Fuzzy*) *Logic*, *Evidence Theory* and *Probability Theory* (see Appendix B) are the formalisms available in *AI*. Fuzzy concepts could for example be used in the objective of network design problems representing "favorable" or "non-favorable" properties. Fuzzy clustering is another technique that was already successfully applied to network design problems [Kin97, Lan99].

A.8 Scalable Architectures

In a highly structured and regular network design problem, regular graphs, such as hypercubes, shuffle exchange networks and DeBruijn Graphs (for an overview see e.g. [Ric97] and [Ric98]) may be an alternative to classical network design algorithms. Regular graphs are advantageous in routing and scalability and, depending on their type, have guaranteed upper bounds on the number of hops, capacity etc. Network designs that depend on regular or structured graphs are called *scalable architectures* (*SCA*).

Above all, WDM-technology with its almost unlimited bandwidth opportunity is a candidate for modeling light-wave networks as regular graphs. Examples can be found e.g. in [GAZ96].

A.9 Hybrid Combinations

Naturally, between the single modeling and solving-techniques no crisp line can be drawn. Most successful solvers for network design problems are combinations of different techniques, i.e. *hybrid combinations*. In many cases heuristics are used to speed up convergence. In the case of problem decomposition different solvers can be applied to each subproblem by choosing the most promising solver.

A.10 Summary and Assessment

All techniques covered in the previous sections have been successfully applied to at least subproblems of network design. They are compared with respect to the following criteria that generally hold either for the algorithms, the models upon which the algorithms are based, or the solution of the algorithms. Table 2 shows a ✓-symbol if the entailed criterion is fulfilled, otherwise it contains a ↯-symbol. If no unique answer can be given, a question mark is shown.

Optimality: A solver has a ✓-symbol if it guarantees an optimal solution (if one exists).

Feasibility: A solver fulfills the feasibility criterion if it guarantees to produce a feasible solution for a feasible problem.

Lower Bounds: An algorithm is said to produce a lower bound β if its solution is at most β-times the optimum ($\lfloor \beta \geq 1 \rfloor$ may be known or an output of the algorithm).

Adaptability: A model is said to be adaptable if it can be easily understood and adapted by network-design experts.

Completeness: A model is said to be complete if it covers all relevant aspects of the problem (e.g. incorporating all necessary constraints).

Computational Tractability: An algorithm is said to be computationally tractable if it has a (at least expected) polynomial running time.

	Opti-mality	Feasi-bility	Lower Bounds	Adapta-bility	Complete-ness	Comp. Tractability
MaPl	↯	✓	↯	✓	✓	✓
HE	↯	✓	?	✓	?	✓
LP	↯	↯	✓	↯	↯	✓
ILP	✓	✓	✓	↯	✓	↯
RST	near optimal	↯	↯	↯	?	✓
AA	↯	✓	✓	↯	?	✓
AI	↯	✓	↯	✓	✓	✓
SCA	↯	✓	✓	↯	↯	✓

Table 2: Comparison of some Algorithmic Concepts for Solving Network Design Problems

Table 2 serves as a rough overview and, quite naturally, is a simplification. For example, there may be heuristics that do not guarantee feasibility, but in general most of the heuristics used in network design do.

B Survey of Existing Uncertainty Formalisms and Measures

This section will give a short overview of available techniques that are able to formalize and measure uncertainties. No thorough introduction, only an assessment is intended. The interested reader may turn to e.g. [KW98] for a detailed introduction.

Throughout the next sections a universal (discrete)[70] set Ω is assumed. An arbitrary set is denoted by capital letters $\llcorner A, \llcorner A_i, i \in \mathbb{N}\lrcorner, B \subseteq \Omega\lrcorner$, a single element of a set will be denoted by the letter x.

B.1 Classical Set Theory

A set is a collection of objects called elements. Canonical operations on sets are union, intersection, complement, subset-hood, is-element-of, equality, and cardinality.

A useful concept from which all the canonical operations can be derived in a straightforward manner is introduced by the characteristic function χ of a set A:

$$\chi : \Omega \to \{0,1\} \quad \chi_A(x) = \begin{cases} 1 & \text{if } x \text{ is an element of } A \\ 0 & \text{if } x \text{ is not an element of } A \end{cases} \tag{34}$$

Uncertainty can be represented by sets of mutually exclusive alternatives but implies no information of which alternative is true. In this case uncertainty arises inherently from **nonspecifity** in each set. Full specification is obtained if only one alternative is possible.

An often used measure of which amount of information is necessary to resolve the uncertainty is given by the Hartley function H, that is a special case of the Shannon entropy:

$$H(A) = \log_2 |A| \tag{35}$$

B.2 Fuzzy Set Theory

Fuzzy set theory is a generalization of classical set theory. Contrary to the classical *crisp* set, the boundary of a fuzzy set is not precise. The transition from membership $\llcorner \chi_A(x) = 1 \lrcorner$ to non-membership $\llcorner \chi_A(x) = 0 \lrcorner$ in a fuzzy set A is gradual. This gradual change can be described by the characteristic function χ_A of the fuzzy set, which is a relaxation of the characteristic function in equation (34):

$$\chi_A : \Omega \to [0,1] \tag{36}$$

Fuzzy set representations are often named after their shapes, e.g. are triangular fuzzy numbers, trapezoidal fuzzy intervals, and so on. An α-cut allows the transformation from a fuzzy set to crisp set and is denoted as **defuzzification**:

$${}^{\alpha}A = \{x \in X \ : \ \chi_A(x) \geq \alpha\} \tag{37}$$

[70] Discrete universal sets Ω have the advantage that each set A in the power set $\wp(\Omega)$, i.e. the set of all subsets, is measurable. $\wp(\Omega)$ forms a canonical σ-Algebra over Ω. For continuous universal sets Ω the results can be adopted straight forwardly if a reasonable σ-algebra over Ω is assumed. See e.g. [Bau96] for an introduction.

The canonical operations correspond to operators of classic set theory, whereas a lot of reasonable[71] definitions of the operator set are possible. By far the most important (also called standard) operators used in literature are:

$$\begin{aligned} \chi_{\bar{A}} &= 1 - \chi_A \\ \chi_{A \cap B} &= \min[\chi_A, \chi_B] \\ \chi_{A \cup B} &= \max[\chi_A, \chi_B] \end{aligned} \tag{38}$$

Fuzzy sets, being a generalization of classical sets, are capable of expressing **nonspecifity**. Additionally, they are capable of expressing **fuzziness**. Typical expressions in natural language are inherently imprecise, i.e. fuzzy. Consider such terms as "high" vs. "low", "good" vs. "bad" and "cold" vs. "warm" as an example. In fuzzy sets, the membership is not a matter of affirmation or denial, but rather a matter of degree.

Uncertainty measures are available, such as fuzzy entropy:

$$\text{fuzzy\_entropy}(A) = -\sum_{x \in \Omega} [\chi_A(x) \log_2 \chi_A(x) + \chi_{\bar{A}}(x) \log_2 \chi_{\bar{A}}(x)] \tag{39}$$

Fuzzy set theory, however, appears to be more suitable for qualitative reasoning, and classification of elements into a fuzzy set than for quantitative estimation of uncertainty.

B.3 Probability Theory

Uncertainty in probability theory is expressed in terms of a measure P defined on the subsets of Ω:

$$P : \wp(\Omega) \rightarrow [0, 1] \tag{40}$$

with the following properties:

$$\begin{aligned} P(\Omega) &= 1 \\ P(\sum_{i \in \mathbb{N}} A_i) &= \sum_{i \in \mathbb{N}} P(A_i) \quad \text{if } A_i \cap A_j = \emptyset \text{ for } i \neq j \end{aligned} \tag{41}$$

$P(A)$ is called **probability** of the subset A and expresses the likelihood of the desired unique alternative to be in the subset A. The probability of A can be therefore interpreted in terms of its frequency of occurrence. Probability theory can model decision situations with conflicting degrees of belief in mutually exclusive alternatives.

It is well known that probability theory is a natural tool for formalizing uncertainty situations that are based on outcomes of a series of independent random events. A number of excellent texts on probabilistic analysis are available in the literature (for an introduction see e.g. [GT96, Bau96]).

Statistical estimation techniques involve estimating **probability density functions** (**PDF**) or moments of distributions from available data or from a collection of a large number of representative samples. Therefore, probabilistic techniques are used most widely for characterizing uncertainty in physical systems. For example, a uniform probability distribution is justified if only a range of possible values for an input but no information about which values are more

[71] i.e. obeying a number of conditions that are not further discussed here

likely to occur is available (based on *Principle of Indifference* or the *Laplace assumption*, see e.g. [SG95]). Similarly, the normal probability distribution is typically used to describe unbiased measurement errors (based on the *Principle of Independence*, see again [SG95], and the central limit theorem). Another source of statistical information in case of limited data available is provided by expert judgments that are interpreted as PDFs (see Section 4.4.2.3).

The **Shannon Entropy** S has been shown to be the only sensible measure of uncertainty in probability theory:

$$S(P) = -\sum_{x \in \Omega} P(\{x\}) \ln P(\{x\}) \tag{42}$$

This function, which forms the basis of *classical information theory*, measures the average uncertainty (in bits) associated with the prediction of outcomes in a random experiment. Its range is $[\,0, \ln |\Omega|\,]$.

B.4 Evidence Theory and Fuzzy Measure Theory

By relaxing the additivity property in equation (41), the theory of evidence can be introduced defining a **belief measure** Bel by:

$$Bel : \wp(\Omega) \to [0,1] \tag{43}$$

with the following properties:

$$\begin{aligned} Bel(\Omega) &= 1 \\ Bel(\emptyset) &= 0 \\ Bel(A_1 \cup A_2 \cup \ldots \cup A_n) &\geq \sum_{i_1} Bel(A_{i_1}) - \\ &\quad \sum_{i_1 < i_2} Bel(A_{i_1} \cap A_{i_2}) + \ldots + \\ &\quad (-1)^{n+1} Bel(A_1 \cap A_2 \cap \ldots \cap A_n)\,, \\ &\quad i_1, \ldots, i_n \in \{1, \ldots, n\} \end{aligned} \tag{44}$$

and a **plausibility measure** Pl by:

$$Pl : \wp(\Omega) \to [0,1] \qquad A \to 1 - Bel(\bar{A}) \tag{45}$$

The entailed **evidence theory** (also called **Dempster-Shafer-Theory** after its founders Dempster & Shafer [Sha76]) assigns to every set A a belief and plausibility measure. It is known to provide an adequate representation of human reasoning.

A generalization of classical measure theory leads to the **fuzzy measure theory** which contains probability theory and evidence theory as special cases.

A fuzzy measure Fm is given by:

$$Fm : \wp(\Omega) \to [0,1] \tag{46}$$

with the following properties:

$$
\begin{aligned}
&Fm(\Omega) = 1 \\
&Fm(A) \leq Fm(B)\,, \text{ if } A \subseteq B \\
&\lim_{i\to\infty} Fm(A_i) = Fm(\bigcup_{i=1}^{\infty} A_i)\,, \text{ if } A_1 \subseteq A_2 \subseteq \ldots \text{ and } \bigcup_{i=1}^{\infty} A_i \in \Omega \\
&\lim_{i\to\infty} Fm(A_i) = Fm(\bigcap_{i=1}^{\infty} A_i)\,, \text{ if } A_1 \supseteq A_2 \supseteq \ldots \text{ and } \bigcap_{i=1}^{\infty} A_i \in \Omega
\end{aligned}
$$

In fuzzy measure theory, all considered sets are crisp and the issue is the likelihood of membership to each of the sets of an object whose characterization is imprecise. In contrast, fuzzy set theory is associated with boundaries of sets and uncertainty in fuzzy measurement theory is associated with boundaries of objects.

It is obvious that a generalization of definitions allows to capture a broader variety of uncertainties than its specializations. Depending on the type of uncertainty, there are different uncertainty measures known and it is likely that with each new subtype[72] of uncertainty the number of measures will grow.

B.5 Interval Arithmetics

Sometimes it is useful to limit some factors by tight lower and upper bounds, e.g. due to imprecise measurements or application of alternative techniques which can determine the bounds directly and where no probability distribution can be obtained.

In this case, interval arithmetic provides a simple method that does not require any further information.

Given two closed intervals $_{\llcorner}a = [a_l, a_u], a_l \leq a_{u_{\lrcorner}}$ and $_{\llcorner}b = [b_l, b_u], b_l \leq b_{u_{\lrcorner}}$, the following operations are defined:

$$
\begin{aligned}
a + b &= [a_l + b_l, a_u + b_u] \\
a - b &= [a_l - b_u, a_u - b_l] \\
a \cdot b &= [\min\{a_l \cdot b_l, a_l \cdot b_u, a_u \cdot b_l, a_u \cdot b_u\}, \max\{a_l \cdot b_l, a_l \cdot b_u, a_u \cdot b_l, a_u \cdot b_u\}] \\
a/b &= [a_l, a_u] \cdot \left[\tfrac{1}{b_u}, \tfrac{1}{b_l}\right], \quad 0 \notin [b_l, b_u]
\end{aligned}
\tag{47}
$$

The primary advantage of this simple method is that except for the bounds, no further information is necessary, allowing to address problems where little information is available. However, this is traded against the fact that no assessment of uncertainty in the results is possible, i.e. all uncertainties are confined to one arithmetic interval.

[72] The tree in Figure 14 can be divided further at some of its leaves revealing that there are different kinds of fuzziness or nonspecifity.

C Introduction to Corporate Finance

C.1 Introduction

The criterion which is used most often in the decision making process between different network designs is the economic costs of a design. However, determination of the costs in terms of currency of a facility over its lifetime may depend on a lot of different organizational as well as environmental factors such as:

- its acquisition type, for example buying or leasing [↑]
- setup and installation costs
- usage-dependent fees
- the scrap value of the facility, i.e. the value after its lifetime
- the interest rate (also called cost of capital [↑]) and the rate of return of the market during its lifetime
- taxes on and depreciation of investments
- inflation
- site costs, such as the custody of external equipment, costs for power, air-conditioning etc.
- insurance fees
- equipment service contracts and/or salaries of staff for maintenance and servicing

In the PC-industry [↑PC], well-known consulting companies such as the GartnerGroup© introduced the term "Total Cost of Ownership" which aims at determination of the overall costs of a piece of equipment during its lifetime. Surprisingly enough, the result of this view on costs is that the initial (setup) costs for a PC are not the dominating costs regarding the whole lifetime of the system.

In evolutionary network design, the decision maker has to choose between different solutions that represent sequences of facility upgrades. The factors mentioned above have to be taken into account when calculating the costs of each solution. Some of the factors are subject to uncertainty analysis, however. For example, the interest rate may change, or the government may introduce a new tax-law that allows another type of depreciation.

To keep the problem tractable (entire platoons of tax experts try to find the best depreciation strategy for a corporation) and not to lose oneself into details, the problem of depreciation is omitted in the following discussion, since the choice between different solutions may not be influenced by the tax law if using the Present Value criterion introduced below.

Additionally, the costs of staff, insurance, sites etc. will be hidden in the reoccurring (e.g. monthly) costs of a facility, and a fixed interest rate as well as a fixed rate of return over the observation interval is assumed.

C.2 Principles of Corporate Finance - Present Value Criterion

One of the most important discoveries in the theory of finance is the cost of money (see e.g. [BM96] for an introduction). Every economically sound decision must take this theory into account. Even if making profit is not in the center of this study, one has to find sound criteria that allow economic comparison of the solutions of the planning process. Especially in the context

of planning networks over time where investments at different times are necessary, there is no unequivocal way to identify the economically best solution.

Here, no detailed study of financial theory will be presented, rather one appropriate, widely accepted technique based on the *Present Value*[73] is introduced. The PV rule is based on the observation that money has a time value: "100 € now are worth more than 100 € in a year". The current amount of money that is equivalent to a future sum is called **Present Value**. The Present Value depends on the rate of return r which can be guaranteed by the market[74].

Definition of Concept 58 (Present Value (PV), Discounted Cash Flow)
*Given an asset of x € in t years' time and assuming a rate of return of r, the **Present Value** (**PV**) of this asset is given by:*

$$PV(x,t) = \frac{1}{(1+r)^t} \cdot x \tag{48}$$

*$PV(x,t)$ is also called **Discounted Cash Flow** (DCF) or **discounted costs** of x.*

Definition of Concept 59 (Discounted Reoccurring Costs (PVR))
Reoccurring costs are a fixed sum x_r that is charged at the beginning of each new time period T_c (e.g. a week or month) over a specified time period T.

Assuming a fixed rate of return $\llcorner r > 0 \lrcorner$ and letting t_c be the fraction of a year which corresponds to T_c, the present value at the beginning of the time period T is:

$$\begin{aligned} PVR(x_r, T_c, T) &= \sum_{i=0}^{\lfloor |T|/|T_c| \rfloor} PV(x_r, i \cdot t_c) = \sum_{i=0}^{\lfloor |T|/|T_c| \rfloor} \frac{1}{((1+r)^{t_c})^i} \cdot x_r \\ &= \left[\sum_{i=0}^{\lfloor |T|/|T_c| \rfloor} \left(\frac{1}{(1+r)^{t_c}} \right)^i \right] \cdot x_r \\ &= \frac{1 - \left(\frac{1}{(1+r)^{t_c}} \right)^{\lfloor |T|/|T_c| \rfloor + 1}}{1 - \left(\frac{1}{(1+r)^{t_c}} \right)} \cdot x_r \end{aligned} \tag{49}$$

Remark 36
*Payments resulting from reoccurring costs during a certain period are financially equivalent to a single payment of their present value up-front the period. If the charges (or returns) are made in terms of yearly payments this asset is also called an **annuity**.*

The Present Value rule allows to discount future cash flows. One advantage of PVs is the linearity if expressed in a single currency (e.g. €s), i.e. all PVs of a project can be added up, resulting in a criterion for a sound economic decision.

In a concrete environment, it is possible to trade setup-costs of a facility against monthly costs. For example, a corporation which does not want to commit a considerable amount of investments

[73] A discussion of Present Value compared to Pay-Back rule, Average-Return-on-Book rule and Internal-Rate-of-Return rule can be found in [BM96].

[74] For instance by the Bundeswertpapiere of the Deutsche Bundesbank which have a current rate of return of about 4.5%.

in network facility hardware may finance the necessary facilities by a loan from a credit institute and consider the repayment rates as reoccurring costs over the lifetime of the facilities. Another interesting possibility which takes advantage of some quirks in national tax-law is to "sell and lease back" the necessary facilities.

As can easily be seen, there can be a vast number of economic possibilities of how a set of facilities can be acquired. The advantage of the PV rule in this regard is that it is "immune" to this kind of "economical tricks" whenever the rate of return is approximately equal to the interest rate. The effects of the tax-law may influence the absolute costs of an investment but since all designs are subject to the same tax-law, the ranking between decisions stays the same.

C.3 The Economical Objective of the *ENDP*

For simplicity, we assume zero inflation and use a static interest rate $\llcorner r > 0 \lrcorner$. It is also convenient to assume a definite project lifetime, i.e. a fixed observation period $\llcorner T = [0, t_{\text{end}}] \lrcorner$, and calculate its PV at time $\llcorner t = 0 \lrcorner$.

The overall discounted costs [+] and yields (i.e. negative costs) [−] of an evolutionary network design $\mathcal{D}$ are :

+ The initial investments of all facilities needed in the initial network design (in case of no given starting conditions) $\mathcal{D}_0$:

$$\sum_{f \in F_{\mathcal{D}_0}} C_f^{nr} \tag{50}$$

+ The discounted costs for all upgrades at time Δ_i. The upgrade costs include the special cases of introducing a non-existing facility, or terminating a facility having no successor (see Definition 10):

$$\sum_{i=1}^{d} \sum_{(f_1,f_2) \in \nabla_i^T} \text{PV}(C_{f_1,f_2}^{\nabla,\Delta_i}, \Delta_i) \tag{51}$$

+ The discounted reoccurring fix costs of each facility during its lifetime. For each facility $\llcorner f \in F \lrcorner$, c_f^r is payable at each time period t_f^r:

$$\sum_{i=0}^{d-1} \text{PV}(\sum_{f \in F_{\mathcal{D}(\Delta_i)}} C_f^r(\llcorner [\Delta_i, \Delta_{i+1}) \lrcorner) \ , \Delta_i) \tag{52}$$

with

$$C_f^r(\llcorner [\Delta_i, \Delta_{i+1}) \lrcorner) := \text{PVR}(c_f^r, t_f^r, \Delta_{i+1} - \Delta_i) \tag{53}$$

Equation 52 is correct if t_f^r is sufficiently small, or if all upgrade times Δ_i are coincident with the end of the reoccurring cost intervals of all facilities $F_{\mathcal{D}(\Delta_i)}$ in the design $\mathcal{D}(\Delta_i)$.

+ The discounted variable costs for usage-dependent facilities (such as dial-up lines).

− The discounted scrap costs when selling an owned facility. This term is zero if only leased facilities are used.

− The discounted costs for selling (unused) facilities on the market (for example by offering the unused bandwidth of a design).

C.4 Cost Models for Network Design Problems

The following cost models for usage-insensitive edge-facilities are encountered most often in practical applications:

- Linear or piecewise linear distance-based tariff: is a very simple cost approximation. The costs grow linearly with distance and capacity. The widely known rule of thumb that installing new fiber costs about 50.000€ per kilometer may serve as an example.
- Concave power-law tariff: Figure 38 shows an **approximation of leased line costs** by a (power-law) function. The underlying cost approximation model is given by:

$$\widetilde{C}(x,l) = s + b^{\alpha}(x) \cdot k \cdot l \tag{54}$$

with the variables:

s setup costs, i.e. the (non-reoccurring) costs for making the facility available
k costs of the basic capacity unit per 1 length unit
α concavity factor $\alpha \in (0,1]$ (corresponds directly to the **economy of scale** [↑EoS])
x required bandwidth and $b(x)$ returns the smallest available capacity greater than x
l distance between the end-points

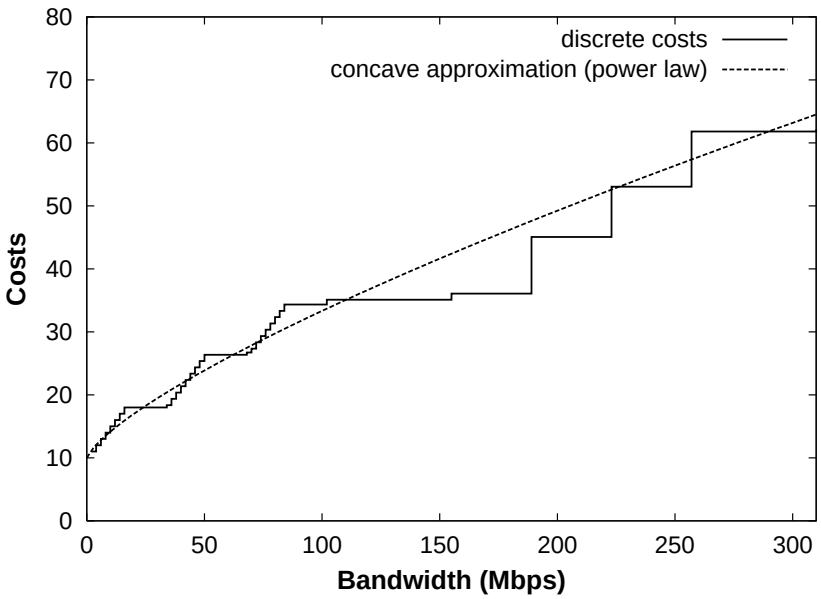

Figure 38: Continuous Approximation of Discrete Bandwidth

Remark 37
A cost function $C(x,l)$ following the power-law model is concave.

- 3-dimensional cost matrix: If only (few) discrete capacities are available, a 3-dimensional matrix indexed by source, destination, and required capacity is a possibility for the representation of tariffs.
- Tariff database: A flexible approach to organizing different tariffs is to hide them behind a tariff database. The user of the database specifies which kind of facility he would like to install and the database returns e.g. the cheapest facility which fulfills the specification (see also Section 5.3.2.4).

D The Unified Modeling Language - UML

D.1 Overview

UML, the Unified Modeling Language, is a diagram-oriented language for analyzing and designing object-oriented systems. UML notation comprises several types of diagrams:

- Use Case Diagrams
- Class Diagrams
- Sequence Diagrams
- Collaboration Diagrams
- State-chart Diagrams
- Activity Diagrams
- Implementation Diagrams

This thesis uses only one type of diagram, a static class diagram which is described in the following section to the necessary level of detail. The interested reader may refer to [Oes97] for a detailed introduction.

D.2 Static Class Diagram

Static class diagrams show the static structure and relations of the abstractions (i.e. classes) of the software design. Class diagrams can be used to show the attributes and operations of a class and the constraints for the way objects collaborate. The UML notation of a static class diagram consists of a set of nodes and edges. The nodes have the form of rectangles and the size and relative position does not matter. An overview of the symbols being used is shown in Figure 39.

Classes are symbolized by rectangles that have three compartments with the following properties: the first compartment contains the name and stereotype of the class, the second are its attributes, and the third are the operations. For convenience, the second and third compartment can be hidden in a diagram. If a class is **abstract**, its name is displayed in *emphasized* letters. UML supports also **parameterized classes** (i.e. template classes), whose parameter is specified in a dashed rectangle on the upper right.

Stereotypes, denoted by matching double brackets (also called guillements) ≪≫, are used to extend a construct at modeling time. Generally, stereotypes represent usage distinctions. Examples are the stereotype ≪interface≫, which denotes a pure abstract base class, or the stereotype ≪bind≫, which instantiates a template with a parameter.

Notes and comments are supplied by a rectangle with its upper right corner folded down. Although newer versions of UML support *design patterns* by additional annotations in the class, here they are included as comments.

The associations between classes are shown by various types of lines between the classes. An association can have a cardinality which is expressed by the number (or range) at the end of the line. If the association has a name, it is written on the top of the connecting line.

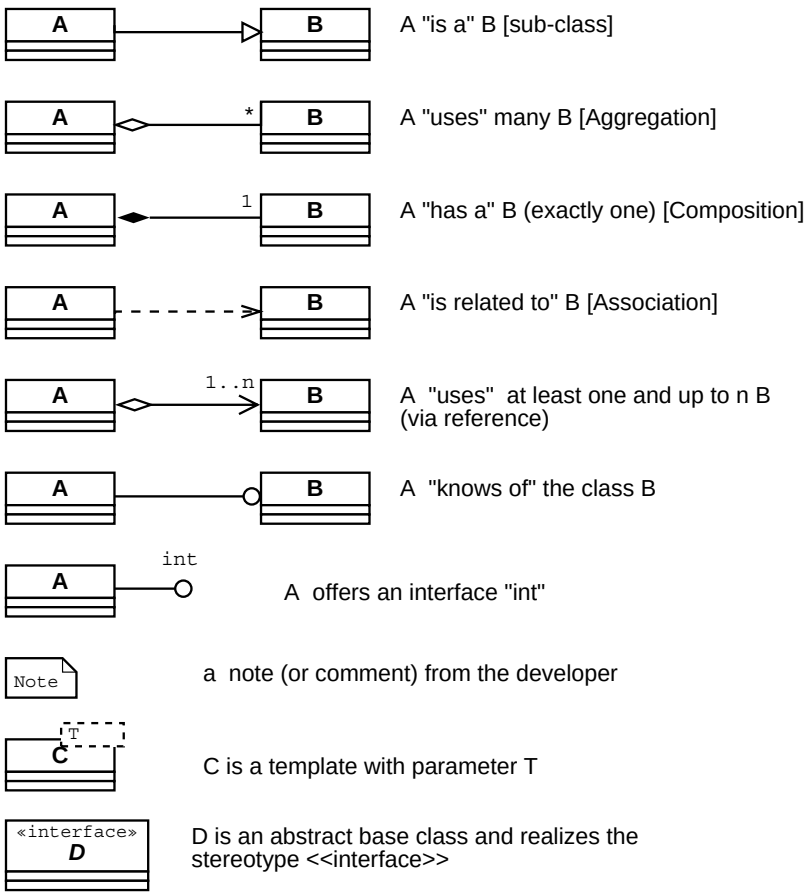

Figure 39: UML Notation

The following types of associations are used:

- **Composition** - means that an object of class `A` "owns" ("has-a" relationship) an object of class `B`, i.e. class `A` is responsible for its associated objects of class `B`, and if an object of class `A` is destroyed, all owned objects of class `B` are also destroyed. The graphical symbol is a line with a filled diamond on the side of class `A`. To express that the association is of a referenced type, the side of the associated class `B` shows an arrow.
- **Aggregation** - is a weaker form of composition ("uses-a" relationship). It means that an object of class `A` has a (temporary) acquisition of an object of class `B` without ownership and responsibility for its lifetime. To express that the aggregation is of a referenced type, the side of the aggregated class `B` shows an arrow.
- **Association** - If the modeler wants to express an association that is not specified in detail, he can use a dashed line between the associated diagram elements.
- **(Public) Inheritance** - ("is-a" relationship) is indicated with a triangle pointing up to the class from which the other is derived.
- **Interface** - ("knows of a" relationship) expresses that the objects of a class `A` know the interface of class `B`. Technically, this implies that the class `A` cannot be compiled without importing the class `B`. The graphical notation is a line with a circle at the end pointing to class `B`.

  Sometimes more than one class support a certain kind of interface. This can be indicated with a (dangling) line that has a small circle at the end of the line.

E WiN Settings and Results

E.1 WiN Settings

E.1.1 Backbone-Nodes

Backbone-node Number	weight	latitude	longitude
1	297.81	50.92	6.92
2	160.41	50.78	6.06
3	147.35	51.02	13.74
4	131.99	53.16	8.75
5	148.18	49.57	11.02
6	21.08	54.09	13.37
7	85.48	54.33	10.12
8	161.17	48.77	9.17
9	692.82	50.10	8.64
10	92.52	49.87	8.65
11	120.32	52.45	13.29
12	780.18	50.10	8.64
13	189.64	49.01	8.39
14	140.46	48.14	11.56
15	9.80	48.56	13.44
16	79.67	52.51	13.32
17	257.00	51.81	10.33

Table 3: WIN Problem: Backbone-Node Weights and Coordinates

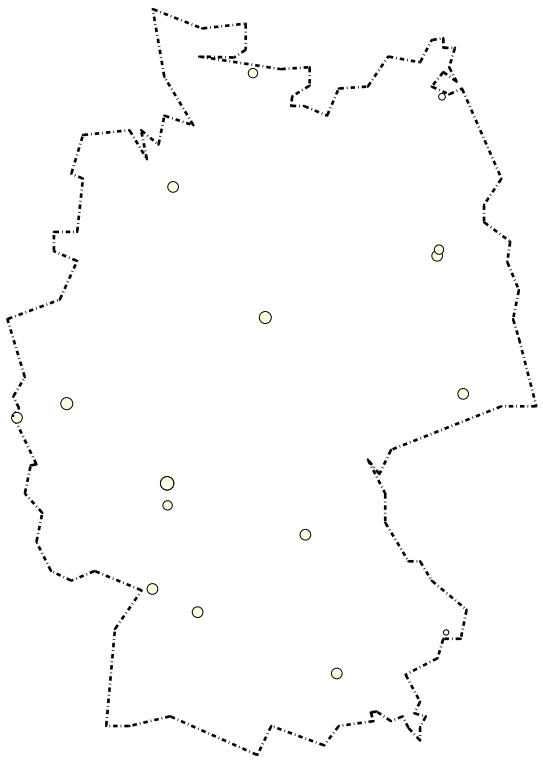

Figure 40: WiN-Problem: Backbone-Node Locations

E.1.2 Initial Traffic Matrix R

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	16.05	3.55	1.28	2.99	1.61	0.34	1.64	2.03	28.84	0.77	1.77	46.00	2.42	1.60	0.07	0.72	3.89
2	8.79	2.42	1.81	2.52	1.54	0.25	1.55	1.75	36.70	1.24	1.72	20.40	2.42	1.56	0.05	0.66	4.37
3	3.97	2.95	8.53	2.49	1.43	0.60	0.84	2.51	24.85	1.55	1.24	14.72	2.76	2.00	0.18	0.86	6.34
4	5.39	1.74	1.48	4.13	0.67	0.83	2.12	0.62	13.08	0.62	1.74	10.94	1.55	0.81	0.02	1.17	3.24
5	2.14	1.03	1.81	0.74	27.39	0.30	0.46	4.74	12.37	3.35	1.19	11.56	1.44	7.65	3.39	0.48	2.02
6	0.15	0.15	0.17	0.19	0.18	0.57	0.05	0.22	2.40	0.04	0.25	1.50	0.15	0.06	0.01	0.07	0.28
7	1.42	0.45	0.27	1.05	0.21	0.23	1.08	0.37	9.08	0.19	0.38	6.04	5.08	0.80	0.01	0.29	0.67
8	3.64	1.11	2.06	1.50	1.93	0.28	0.80	1.02	20.91	0.83	0.73	16.51	1.08	1.78	0.05	0.73	3.31
9	64.79	24.49	24.07	25.54	29.43	4.31	21.33	30.47	46.69	19.40	16.79	13.74	38.76	30.41	1.52	13.28	47.51
10	0.76	0.77	0.88	0.30	0.32	0.05	0.16	2.04	14.22	0.56	0.52	8.05	3.15	0.55	0.012	0.19	3.61
11	3.09	1.34	2.39	1.93	1.49	0.43	1.08	1.50	16.96	1.67	3.18	15.01	5.26	3.78	0.09	3.64	4.17
12	87.52	28.68	31.49	32.25	46.21	5.51	19.65	49.11	29.33	22.66	21.66	0.20	52.37	34.11	2.29	20.88	66.19
13	5.45	1.70	2.14	6.26	1.40	0.64	6.38	0.95	17.22	1.60	1.40	13.96	4.41	1.79	0.05	0.80	3.53
14	2.23	1.23	1.06	1.34	2.89	0.13	0.44	2.46	16.97	0.79	1.01	14.47	2.53	3.56	0.13	0.55	1.66
15	0.05	0.01	0.02	0.03	0.09	0.01	0.02	0.02	0.60	0.02	0.01	0.73	0.02	0.08	0.02	0.01	0.03
16	1.52	0.26	1.22	1.69	0.52	0.36	0.98	0.81	9.33	0.34	3.63	10.62	0.77	0.83	0.02	7.43	1.17
17	7.35	3.61	5.91	5.15	3.59	0.92	1.43	4.29	34.10	1.80	2.34	25.92	4.50	2.65	0.14	1.19	11.14

Table 4: WiN-Problem: Initial Traffic Matrix R (in Mbps)

E.1.3 Cost Matrix for 155 Mbps Facility-Types

$n_1 \downarrow \setminus n_2 \rightarrow$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	49	216	136	156	249	200	140	86	92	213	86	120	207	240	215	127
2	49	0	241	154	176	272	217	151	104	109	238	104	128	223	259	240	152
3	216	241	0	191	125	162	201	190	174	178	89	174	202	168	135	92	125
4	136	154	191	0	197	154	88	221	161	171	151	161	210	262	268	151	98
5	156	176	125	197	0	236	238	89	97	94	168	97	105	90	108	171	126
6	249	272	162	154	236	0	110	289	244	253	98	244	290	295	271	95	155
7	200	217	201	88	238	110	0	274	218	227	143	218	267	304	298	142	137
8	140	151	190	221	89	289	274	0	86	76	226	86	50	101	151	228	164
9	86	104	174	161	97	244	218	86	0	35	191	25	74	147	180	193	114
10	92	109	178	171	94	253	227	76	35	0	198	35	64	140	176	200	123
11	213	238	89	151	168	98	143	226	191	198	0	191	231	223	198	27	110
12	86	104	174	161	97	244	218	86	25	35	191	0	74	147	180	193	114
13	120	128	202	210	105	290	267	50	74	64	231	74	0	126	174	233	161
14	207	223	168	262	90	295	304	101	147	140	223	147	126	0	83	226	192
15	240	259	135	268	108	271	298	151	180	176	198	180	174	83	0	201	194
16	215	240	92	151	171	95	142	228	193	200	27	193	233	226	201	0	112
17	127	152	125	98	126	155	137	164	114	123	110	114	161	192	194	112	0

Table 5: WiN-Problem: Cost Matrix for 155 Mbps Line Facilities $\widetilde{C}(155, n_1, n_2, 12)$ (in 10^3 Currency Units)

E.2 *SNDP* Tabu Search Results

E.2.1 *SNDP* with symmetric lines

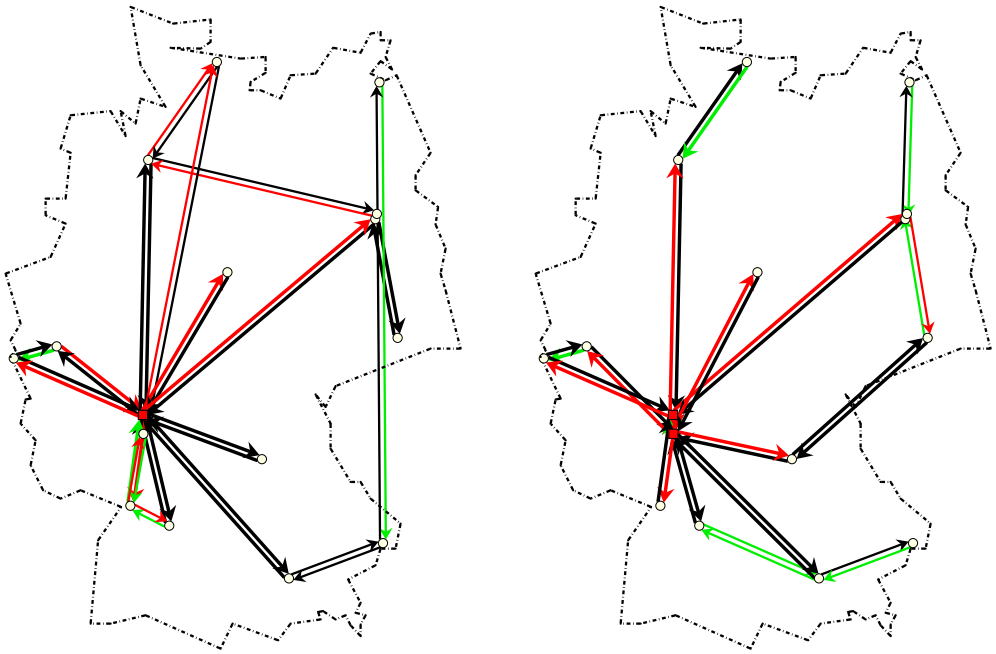

Figure 41: WiN-Problem: Static Network Design (Symmetric Facilities) [Result 2&3]

E.2.2 *SNDP* with asymmetric lines

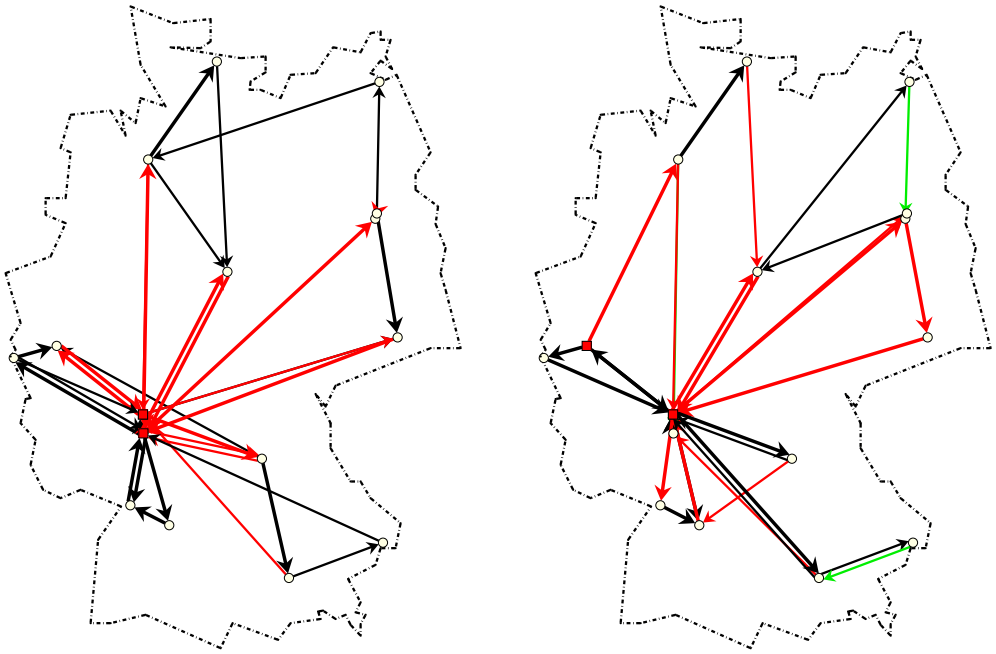

Figure 42: WiN-Problem: Static Network Design (Asymmetric Facilities) [Result 2&3]

Glossary of Terms & Abbreviations

e.g.: ex gregis.

i.e.: id est.

vs.: versus.

ABR: Available Bit Rate, an ATM traffic class.

Add Drop Multiplexer (ADM): A multiplexer capable of extracting or inserting lower rate signals from a higher-rate multiplexed signal without completely de-multiplexing the signal.

AMP: Adaptive Memory Programming.

ATM: Asynchronous Transfer Mode, a network technology based on cell switching.

Architecture: A network architecture identifies all relevant standards and the circumstances under which a certain network technology is realized (which often includes specific products).

Automatic Protection Switching (APS): The ability of an SDH network element to detect a failed working line and switch the service to a spare protection line.

BC: Bell Company.

B-ISDN: Broadband ISDN, i.e. ISDN providing higher data rates. ATM is the most common network technology deployed for providing B-ISDN services.

Branch & Bound (B&B): An implicit enumeration algorithm for solving combinatorial optimization problems. A general B&B algorithm for mixed *ILP*s operates by solving a *LP* relaxation of the original problem and then branching on a variable which has not currently on an integer value.

Branch & Cut (B&C): B&C combines B&B with cutting plane algorithms. B&C tries to eliminate progressively fractional variables by constraints which exclude fractional variables (by cutting planes).

Burstiness: Traffic whose ratio of the maximum intensity to the average intensity is very high is called bursty.

Busy Hour Traffic: Average traffic of the most busy hour in 30 days within one year (ITU-T Rec. Q.80).

B-WiN: Breitband Wissenschaftsnetz

Capacity: The maximum transmission rate (measured in kbps or Mbps) that can be carried by a facility.

Cash Flow: Receipts and Expenditures.

Carrier: A company which provides communication circuits. Carriers are split into "private" and "public". A private carrier can refuse your services, a public carrier cannot. Public carriers are regulated.

CBR: Constant-Bit Rate, an ATM traffic class.

CCITT: Comité Consultatif International Télégraphique et Téléphonique - an international standards-setting body, now part of the ITU.

CDF: Cumulative Distribution Function.

Congestion: The over-loading of a facility of a network that causes a noticeable reduction in throughput and delay is called congestion.

Connection Admission Control (CAC): Set of actions taken by the network during the call setup phase in order to determine whether a connection request can be accepted or should be rejected. CAC gives an answer to the question: "Can the requested service be accepted by the service provider (e.g. a link, a node, etc.) at its requested QoS without violating the QoS guarantees made to the on-going services?"

Connectivity: The connectivity of a network is the ratio between number of nodes and edges of a network. 2-edge (2-node) connectivity of a network means that the network stays connected if any edge (node) is removed.

Convergence: has two different meanings:

i) Strategic implications of information systems often refer to technological convergence. This phenomenon can be described as a merging of all forms of communication, voice, video and data currently with different user interfaces and transmission mechanisms towards an integrated transmission mechanism.

ii) In economical context, convergence refers to a development where the number of providers of services or goods decreases until only a few (global) players are left.

Corporate Network: Company-wide multiservice communication network. Usually consists of **Access Network** and wide-area **Backbone Network**.

Cost of Capital: The interest rate being charged to secure a loan.

Cross Connect (CC or DCS): An electronic cross-connect which has access to lower rate channels in higher rate multiplexed signals and can rearrange (cross-connect) those signals.

CTI: Computer Telephony Integration - merging of traditional telecommunication (PBX [↑]) equipment with computer applications.

DCF: Discounted Cash Flow.

Def.: Definition.

DFN: Deutsches Forschungsnetz Verein

Dial-Up Line: Accessing a service by circuit switching, involving the setup of a circuit by dialing the (telephone) number of the target.

Differentiated Services (DiffServ): uses the IP header Type of Service bits to influence the per hop behavior obtaining QoS with IPv4. Initiative by the IETF [RFC98c].

Diversity: The share of traffic flowing on physically different, i.e. disjoint, paths.

EB: Effective Bandwidth.

Economy of Scale / EoS: Responsible for the tendency of larger systems to be more cost efficient than smaller systems ([↑Grosch's law]).

***END*:** Evolutionary Network Design.

***ENDP*:** Evolutionary Network Design Problem.

Enterprise Resource Planning (ERP): is a collection of software programs which ties all of an enterprise's various diverse functions (finance, manufacturing, sales, etc.) into a cohesive data base. This software is also entailed with the analysis of this data to plan production, forecast sales, analyze quality, etc. SAP R/3© or Oracle© 8 are examples of ERP software products.

Eq.: Equation.

Erlang: Unit [Erl] that represents the average number of simultaneous calls (1 Erl = 360 call seconds = 36 CCS).

ETSI: European Telecommunication Standards Institute.

GA: Genetic Algorithms.

Game Theory: Game theory was developed under the assumption of decision making in the context of a game versus "intelligent" opponents (i.e. players). Each player has a number of possible choices called *strategies*. The objective is to reach a certain goal or the highest possible "score".

Game theory may be helpful for assessing system behavior when each individual user or process behaves selfish, i.e. tries to maximize its profit. Pricing or routing schemes should be tested under this assumption.

Gbps: Giga bit per Second. $\llcorner 10^9 \text{bit} \cdot s^{-1} \lrcorner$.

GRASP: Greedy Randomized Adaptive Search Procedures.

Grosch's Law It is a market truth that under normal circumstances the costs of a device having double performance costs only square root of two times more (also called **square root law**).

IETF: Internet Engineering Task Force.

***ILP*:** Integer Linear Program.

***IND*:** Incremental Network Design.

***INDP*:** Incremental Network Design Problem.

Integrated Services: An initiative by the IETF Integrated Services Working Group which develops a set of standards that cover how application services define their requirements, how this information is made available to routers on a hop-by-hop basis, and ways of testing and validating that the contracted QoS is maintained. RSVP [↑] is a protocol which may provide Integrated Services.

Intelligent Network / IN: Network supporting surplus services like special tariffs, VPNs, 0800 Numbers.

Internet: A global federation of networks based on the IP protocol family.

Internetworking: The process of connecting two dissimilar networks, generally whose ownership is different.

Interoperability: The ability of equipment from different manufacturers to operate together.

Intranet: A company wide network based on the IP protocol family that usually is not fully accessible from other networks.

IP/IPv4: Internet Protocol Version 4. The network layer protocol used in the Internet and Intranet [RFC81].

IPv6: Internet Protocol Version 6. The successor of the network layer protocol IPv4 [RFC98a].

ISP: Internet Service Provider - A company that sells Internet access to individuals or organizations.

ITU: International Telecommunication Union, an international standards-setting body.

ISO-OSI Reference Model: ISO is an abbreviation for "International Standards Organization", an international standards-setting body. OSI is an abbreviation for "Open Systems Interconnection". The ISO-OSI Reference Model is a stratification of communication between end systems.

kbps: Kilo Bit Per Second $\llcorner 10^3 \text{bit} \cdot s^{-1} \lrcorner$.

Landau-Symbol: The Landau-symbol $O(\cdot)$ serves as a (space, time, etc.) complexity measure: $f, g : \mathbb{N} \to \mathbb{R}$ then $f = O(g)$ if $\exists n_0 \in \mathbb{N} \wedge \exists c \in \mathbb{R}^+$ such that $\forall n \in \mathbb{N} \wedge n > n_0 \ : \ f(n) \leq c \cdot g(n)$.

Latin Hypercube Sampling: is a widely used variant of standard Monte Carlo sampling. In this method, the range of probable values for each uncertain input parameter is divided into ordered segments of equal probability. Thus, the whole parameter space, consisting of all the uncertain parameters, is partitioned into cells having equal probability, and the sampling can be performed in an "efficient" manner such that each parameter is sampled once from each of its possible segments. The advantage of this approach is that the random samples are generated from all the ranges of possible values, thus giving insight into the extremes of the probability distributions of the outputs ([Isu99]).

Layer 4 Switching: policy based switching.

Leasing: A lease is just an extended rental agreement. The owner of the equipment (the lessor) allows the user (lessee) to operate the equipment in exchange for regular (usually fixed) lease payments and a one time initial charge. There is a wide variety of possible arrangements. Operating leases are attractive to equipment users if the lease payment is less then the users equivalent annual costs of buying the equipment.

Link: A communication facility or service between two end nodes.

LLC: Logical Link Control, part of the ISO-OSI Layer 2 protocol.

***LP*:** Linear Program, see also [↑MP].

Mbps: Mega Bit Per Second. $\llcorner 10^6 \text{bit} \cdot s^{-1} \lrcorner$.

Monte Carlo Sampling: Given a number of (independent) model parameters which have uncertainties formalized by probability distributions, random sampling of parameters from their distribution and successive model runs are performed until a significant distribution of output parameters is obtained ([Isu99]).

MP: Mathematical Programming. A mathematical program according to [Gre00] is an optimization problem of the (standard) form:

$$\text{Maximize } o(x) : x \in X,\ g(x) \leq 0,\ h(x) = 0 \tag{55}$$

where X is a subset of $\mathbb{R}^n$ and is in the domain of the real-valued functions, o, g and h. The relations, $g(x) \leq 0$ and $h(x) = 0$ are called constraints, and o is called the objective function.

The sense of optimization is presented here as maximization, but it could just as well be minimization, with the appropriate change in the meaning.

Mathematical programming is the study or use of the mathematical program. It includes any or all of the following:

Theorems about the form of a solution, including whether one exists; Algorithms to seek a solution or ascertain that none exists; Formulation of problems into mathematical programs, including understanding the quality of one formulation in comparison with another; Analysis of results, including debugging situations, such as infeasible or anomalous values; Theorems about the model structure, including properties pertaining to feasibility, redundancy and/or implied relations (such theorems could be to support analysis of results or design of algorithms); Theorems about approximation arising from imperfections of model forms, levels of aggregation, computational error, and other deviations; Developments in connection with other disciplines, such as a computing environment.

MPLS: Multi-protocol Label Switching. Layer 3 switching technique proposed by the IETF [CFF+99].

MPOA: Multi-protocol over ATM. A standard proposed by the ATM-Forum [ATM99].

Multi-Drop Line: A line facility connecting more than two devices.

***NDP*:** Network Design Problem.

***NOP*:** Network Operation Planning.

Operations Cost: Reoccurring expenditures for the operation and maintenance of the telecommunication network and services.

OSPF: Open Shortest Path First, a dynamic, link-state routing protocol standardized by the IETF [RFC98b].

PBX: Private Branch Exchange. A circuit switch providing connections to both public switches and private network facilities.

PC: Personal Computer. A desktop computer, which is designed for serving the personal needs in terms of computing power, storage, memory etc. of one user at a time.

PDF: Probability Distribution Function.

PDH: Plesynchronous Digital Hierarchy [↑SDH].

Peering: Peering is the arrangement between network providers. It includes agreements that describe the terms and conditions of the exchange of traffic and router information between the peered networks.

Photonic Network: A summary of forthcoming network technologies that are based exclusively on optical components (and introduce a new "optical layer"). Today's high speed networks like SDH still have to rely on opto-electronic components.

PNNI: Private Network-to-Network Interface. A routing protocol that allows to integrate ATM switches of different vendors.

POP: Point of Presence.

POTS: Plain old Telephone Service.

Present Value (PV): The value of money at the present time, even if the money is due at some future point.

Protocol: An agreement which determines the procedure of establishing inter-operation between two parts (usually having the same task) of a computer system.

Proxy: The mechanism where one system "fronts for" another system.

PSTN: Public Switched Telephone Network.

PTT: Postal Telegraph & Telephone. National Telephone Company (sometimes monopolistic).

Public Network: A network operated by common carriers or telecommunications administration for the provision of circuit-switched, packet-switched and leased lines to the public.

Queuing: Queuing is the discipline that addresses congestion and ways to deal with it. The theory can be used to predict delay, minimize delay, estimate queue length, estimate number of required servers.

Quality of Service (QoS): QoS is the collective of service characteristics which determine the degree of user satisfaction with this service. QoS can be measured by throughput, delay, jitter, customer contentment, etc. Sometimes also called **Grade of Service**.

Reliability: Reliability in the context of networks means that crashes are more graceful, i.e. not the whole system goes down if one facility fails. Sometimes the reliability of a network is referred to as the probability that the functioning nodes are connected by working links.

Renting: [↑Leasing].

RIP: Routing Information Protocol, a static distance-vector routing protocol proposed by the IETF [RFC88].

Routing/Routing Algorithm: Routing is the act of moving information across a network from a source to a destination. Routing involves two basic activities: determining optimal routing path and transporting the information through the network. In the Internet, each intermediary node on the path from the source to the destination performs routing by passing along the information (i.e. the IP-packet) to the next node. Part of this process involves analyzing a routing table to determine the optimal path.

A metric is a standard of measurement, such as path length, that is used by routing algorithms to determine the optimal path to a destination. To aid the process of path determination, routing algorithms initialize and maintain routing tables, which contain route information. Route information varies depending on the routing algorithm used.

Routing algorithms often have one or more of the following design goals: optimality, simplicity, robustness and stability, rapid convergence and flexibility.

RSVP: Resource Reservation Protocol, a standard for Integrated Services proposed by the IETF [RFC97].

SA: Simulated Annealing.

SDH: Synchronous Digital Hierarchy, successor to PDH. Multiplexing technology used to carry wide area communication. The following hierarchy exists in SDH networks:

Technology	USA	Europe	Bandwidth
SDH	STS-1 (OC1)		51.840 Mbps
SDH	STS-3 (OC3)	STM-1	155.520 Mbps
SDH	STS-12 (OC12)	STM-4	622.080 Mbps
SDH	STS-48 (OC48)	STM-16	2488.320 Mbps
SDH	STS-192 (OC192)	STM-64	9953.280 Mbps

Sec.: Section.

Self Sizing Network: The attribute "self sizing" is given to networks that allow to reconfigure themselves dynamically in order to optimize performance (which has to be monitored) at least to some degree. It differs from *NOP* (see Section 2.1) by the time-scale of reconfiguration (self sizing networks can react to decreasing performance on short notice) and by the fact that *NOP* is a human-driven process.

Self Healing Ring (SHR): SDH configuration as a ring having working capacity and a protection capacity. SHRs have the advantage to provide very rapid service restauration.

Single/Multi Hop Virtual Path (SHVP/MHVP): A logical configuration of an ATM - network where each node can be reached from another by one or many hops, respectively.

SLA: Service Level Agreement, QoS that is guaranteed by a provider by contract.

***SND*:** Static Network Design.

***SNDP*:** Static Network Design Problem.

Survivability: Survivability is the ability of the network to be resilient against outages and is often measured in terms of traffic that survives a link or node failure (see also reliability).

Switching: Switching is generally used to describe the transfer of data from an input port to an output port where the selection of the output port is based on OSI Layer 2.

Tariff: A statement by a communications company submitted for approval to the Federal Communications Commission that sets forth the services offered and the rates, terms, and conditions for the use of this service.

Throughput: Number of information bits that the network can transport per unit of time.

Traffic Engineering: Traffic Engineering is the ability of the network operators to dedicate the path the traffic takes through the network.

Traffic Management: Traffic Management is a set of network actions that monitor and control traffic. It provides each traffic stream with its desired level of bandwidth and control over cell loss, cell delay and delay variation.

Traffic Shaping: Mechanisms that may be used to achieve a desired modification of the traffic characteristics.

TS: Tabu Search.

VAN: [↑Wide Area Networks].

VBR: Variable Bit Rate, an ATM traffic class.

Virtual Private Network (VPN): A VPN is a network which a customer perceives as being private. In reality, it is part of a larger shared infrastructure maintained by a service provider. The VPN retains all the intrinsic features of a private network including a unique addressing plan. VPNs therefore allow the exploitation of (insecure) public communication services for private communication by use of tunneling and cryptography. Sometimes also referred as "software defined networks".

VLAN: Virtual LAN. Membership to a virtual LAN is defined administratively independent of the physical network topology. A virtual LAN segment is a unique broadcast domain.

VMR: Variance Mean Ratio. Ratio between variance and average rate of traffic observed in a certain period. Traffic is called *rough* if $\llcorner \mathrm{VMR} > 1 \lrcorner$ else *smooth*.

Wide Area Networks (WANs): Services provided by WANs can be divided into two main groups:

- **Basic Services** are services in which the service provider is offering simply a circuit of a specified capacity between two points.
- **Value Added Networks (VANs)** are services, utilizing lines owned by telecommunication companies or other common carriers, which offer network services to organizations or individuals at advertised or agreed price structures. Examples are Packet Switched Services (Internet) or Frame Relay services.

WiN: Wissenschaftsnetz.

WDM: Wavelength Division Multiplexing. A forthcoming wide-area network technology that uses multiplexing of colors having different wavelengths over one fiber cable. Standardization of WDM technology is not finished at the time of writing.

Weighted Fair Queuing (WFQ): WFQ provides high priority traffic with a higher priority service than simple data traffic.

World Wide Web (WWW): Graphical Information Service provided by some hosts in the Internet. Uses the protocol HTTP and formatting language HTML.

List of Figures

List of Tables

List of Algorithms

References

[ACM81] G.R. Ash, R.H. Cardwell, and R.P. Murray. Design and Optimization of Networks with Dynamic Routing. *The Bell System Technical Journal*, October 1981.

[AMO93] Ravindra Ahuja, Thomas Magnanti, and James Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall Inc., 1993.

[ATM99] Multi-Protocol Over ATM Specification v1.1. Technical report, The ATM Forum - Technical Committee, May 1999. af-mpoa-0114.000.

[Auc99] Ben Auch. Design und prototypische Implementierung einer integrierten Plattform zur Planung von hierarchischen Netzen. Master's thesis, Technische Universität München, Institut für Informatik, 1999.

[Bau96] Heinz Bauer. *Probability Theory*. de Gruyter Inc., 1996.

[Bau97a] Thomas Bauschert. Multihour Design of Multi-Hop Virtual Path based Wide-Area ATM Networks. In *15th International Teletraffic Congress - ITC 15, Washington, USA*, pages 1019–1030, 1997.

[Bau97b] Thomas Bauschert. *Optimale Dimensionierung von ATM-Weitverkehrsnetzen mit mehrstufiger Durchschaltung*. PhD thesis, Technische Universität München, 1997. Institut für Elektrotechnik.

[BCRH+99] V. Bolotin, J. Coombs-Reyes, D. Heynab, Y. Levy, and D. Liu. IP Traffic Characterization for Planning and Control. *International Teletraffic Conference*, 16:425–436, 1999.

[BD89] Rudolf Bäßler and Andreas Deutsch. *Nachrichtennetze*. VEB Verlag Technik Berlin, 1989.

[BDG88] José Luis Balcázar, Josep Diaz, and Joaquim Gabarró. *Structural Complexity I*, volume 11. Springer-Verlag, 1988. EATCS Monographs on Theoretical Computer Science.

[Bec98] D. Beckmann. Optimierung von Kommunikationsnetzen mit Hilfe Genetischer Algorithmen. Lecture 29.7.1998 at Siemens Neuperlach, 1998.

[Ber89] Lothar Berger. *Methodenintegration zur Planung von Rechnernetzen*. PhD thesis, Universität Karlsruhe, 1989.

[Bes99] Nicoletta Bestler. Implementierung und Evaluierung heuristischer Netzplanungsverfahren. Master's thesis, Technische Universität München, Institut für Informatik, 1999.

[BF77] Robert Boorstyn and Howard Frank. Large-Scale Network Topological Optimization. *IEEE Transactions an Communications*, 25(1):29–47, January 1977.

[BGW97] Andreas Bley, Martin Grötschel, and Roland Wessäly. Modellierung zur "Topologie-, Kapazitäts und Routingpanung für das B-WiN". Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin, September 1997.

[BM96] Richard Brealey and Steward Myers. *Principles of Corporate Finance*. McGraw-Hill Inc., 1996.

[Böc97] Stefan Böcking. *Objektorientierte Netzwerkprotokolle - Grundlagen, Entwurf und Implementierung*. Addison Wesley Longman Publishing Company, 1997.

[Boo91] Grady Booch. *Object oriented Design with Applications*. The Benjamin/Cummings Publishing Company, 1991.

[BPT98] Thomas Bauschert, Maren Perske, and Lars Turczyk. Advanced Broadband Network Planning with SICAP-ATM. *Networks 98: 8th International Network Planning Symposium, Sorrento*, pages 593–598, 1998.

[Cah98] Robert S. Cahn. *Wide Area Network Design - Concepts and Tools for Optimization*. Morgan Kaufmann Publishers Inc., 1998. The Morgan Kaufmann Series in Networking.

[CFF+99] R. Callon, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A Framework for Multiprotocol Label Switching. Technical report, Internet Engineering Task Force - Network Working Group, June 1999.

[CH94] John Clark and Derek Allan Holton. *Graphentheorie - Grundlagen und Anwendungen*. Spektrum Akademischer Verlag, 1994.

[CIS96] Cisco Systems Inc. *Scaling the Internet with Tag Switching*, 1996. White Paper.

[CIS99] Cisco Systems Inc. *NetFlow Services and Applications*, 1999. White Paper.

[Cop92] James O. Coplien. *Advances C++ Styles and Idioms*. Addison Wesley Publishing Company, 1992.

[CP95] G. Collins and L. Pryor. Planning under Uncertainty: Some Key Issus. Technical Report DAI Research Paper No. 749, University of Edinburgh, April 1995.

[CSI98] Siemens AG & Newbridge Network Corporation. *MainStreetXpress - The Evolution of Virtual Private Network Services*, 1998. White Paper.

[DAA95] Thomas Dean, James Allen, and Yiannis Aloimonos. *Artificial Intelligence - Theory and Practice*. The Benjamin/Cummings publishing Company Inc., 1995.

[DH98] Bharat Doshi and P. Harshavardhana. Broadband Network Infrastructure of the Future: Roles of Network Design Tools in Technology Development Strategies. *IEEE Communications Magazine*, pages 60–71, May 1998.

[DMT97] M.A.H. Dempster, E.A. Medova, and R.T. Thompson. A Stochastic Programming Approach to Network Planning. In *International Teletraffic Conference*, number 15, pages 329–339, 1997.

[Dom95] Wolfgang Domschke. *Einführung in Operations Research*. Springer-Verlag Berlin Heidelberg New York, 1995. 3. Auflage.

[Doo94] F. van den Dool. Telecommunication System Architectures: Dealing with Complexity. *International Journal of Communication Systems*, 7, 1994.

[Dut94] Amitava Dutta. Capacity Planning of Private Networks Using DCS under Multibusy-Hour Traffic. *IEEE Transactions on Communications*, 42(7):2371–2374, July 1994.

[FB98] Jochen Frings and Thomas Bauschert. Integrated Planning of Broadband Networks: A Basic Framework. *Workshop on the Design of Reliable Communication Networks*, May 1998.

[FD95] Hany I. Fahmy and Christos Dougligeris. END: An Expert Network Designer. *IEEE Network*, pages 18–27, November/December 1995.

[Fis99] Volker G. Fischer. Design and Implementation of FooNet - a Framework for object-oriented Network Design. Technical Report TUM-I9919, Technische Universität München, Institut für Informatik, December 1999.

[Fis00] Volker G. Fischer. Foonet homepage. `http://wwwjessen.informatik.tu-muenchen.de/~fischerv/foonet`, 2000.

[FJSP99] Volker G. Fischer, Manfred Jobmann, Christian Schwingenschlögl, and Werner Pohlmann. A Simulation Study of ATM Call Admission Control. *Modellierung, Messung und Bewertung von Rechensystemen MMB 1999*, September 1999.

[Fre86] Simon French. *Decision Theory: An Introduction to the Mathematics of Rationality.* Ellis Horwood Limited, 1986.

[FS97] Volker G. Fischer and Manfred Schramm. Probabilistic Reasoning with Maximum Entropy - The System Pit. In Francois Bry, Burkhard Freitag, and Dietmar Seipel, editors, *Twelfth Workshop on Logic Programming.* Ludwig Maximilians Univeristät München, September 1997.

[FSJ99] M. E. Fayad, D. C. Schmidt, and R. E. Johnson. *Building Application Frameworks : Object-Oriented Foundations of Framework Design.* Horizon Publishers & Distributors Inc., 1999.

[Gav92] Bezalel Gavish. Topological Design of Computer Communication Networks - The Overall Design Problem. *European Journal of Operational Research*, 58:149–172, 1992.

[GAZ96] D. Guo, A.S. Acampora, and Z. Zhang. Hyper-Cluster: A Scalable and Reconfigurable Wide-Area Lightwave Network Architecture. *IEEE Global Telecommunications Conference - Globecom*, pages 863–867, 1996.

[GB96] Manfred Geyer and Markus Buchner. Strategic Optimization of Transport Networks using Netplan (SICAP-Transport). *Networks 96: 7th International Network Planning Symposium, Sydney*, 1996.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley Publishing Company, 1995.

[GK77] Mario Gerla and Leonard Kleinrock. On the topological Design of Distributed Computer Networks. *IEEE Transactions on Communications*, COM-25(1):48–60, January 1977.

[Gog99] Helmut Gogl. Lastprognose und Lastmodell für das G-WiN - Abschlußbericht. Technical report, Deutsches Forschungsnetz Verein, October 1999.

[Gog00] Helmut Gogl. *Measurement and Characterization of Traffic Streams in High-Speed Wide Area Networks (in progress).* PhD thesis, Technische Universität München, 2000. Institut für Informatik.

[Gör97] Camelita Görg. *Verkehrstheoretische Modelle und stochastische Simulationstechniken zur Leistungsanalyse von Kommunikationsnetzen*, volume 13 of *Aachener Beiträge zur Mobil- und Telekommunikation*. Verlag der Augustinus Buchhandlung, 1997.

[GP87] E.P. Gould and C.D. Pack. Communications Network Planning in the Envolving Information Age. *IEEE Communications Magazine*, 25(9):22–30, September 1987.

[Gre96] Michael Greiner. *Untersuchung und Entwicklung mathematischer Optimierungsverfahren zur Dimensionierung von Rechensystemen*. PhD thesis, Technische Universität München, 8 1996. Institut für Informatik.

[Gre00] Harvey J. Greenberg. The Nature of Mathematical Programming. `http://www.cudenver.edu/~hgreenbe/glossary/nature.html`, Januar 2000.

[Gro99] Christian Grob. Entwurf und Implementierung einer Softwareumgebung zur Analyse von Verkehrsströmen und ihre Anwendung auf Wide Area IP Verkehr. Master's thesis, Institut für Informatik, Technische Universität München, April 1999.

[GT96] Michael Greiner and Gottfried Tinhofer. *Stochastik für Studienanfänger der Informatik*. Hanser Verlag, 1996.

[Ham94] Jamed D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[HHB94] Benjamin F. Hobbs, Jeffrey C. Honious, and Joel Bluestein. Estimating the Flexibility of Utility Resource Plans: An Application to Natural Gas Cofiring for SO_2 Control. *IEEE Transactions on Power Systems*, 9(1):167 – 173, 1994.

[HHMK89] James Harris, Michael Humes, Joseph Meichamp, and O-Joung Kwon. NESTOR: Network Engineering System for TOpology and Requirements. *Expert Systems*, 6(3):134–142, August 1989.

[Him00] Michale Himsolt. GTL - Graph Template Library. `http://www.fmi.uni-passau.de/Graphlet/GTL`, 2000.

[Hoc95] Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1995.

[HP98] Anthony Hunter and Simon Parsons, editors. *Applications of Uncertainty Formalisms*. Number 1455 in Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg New York, 1998.

[HSV+99] Petr Horacek, Radek Sindelmar, Zdenek Vleck, Martin Vojtisek, and Milon Vojnar. Midrange Prediction of Data Throughput in a Communication Network. Technical report, Czech Technical University in Prague, Faculty of Electrical Engineering, 1999.

[ISO97] ISO/IEC Final Draft International Standard 14882 - Programming Language C++. Technical report, 11 1997.

[Isu99] Sastry S. Isukapalli. *Uncertainty Analysis of Transformation Models*. PhD thesis, State University of New Jersey, January 1999.

[ITU83] *General Network Planning*. International Telecommunication Union, 1983.

[ITU92a] International Telecommunication Union. *Forecasting International Traffic*, 1992. E.506 (rev.1).

[ITU92b] International Telecommunication Union. *Forecasting new Telecommunication Services*, October 1992. E.508.

[ITU93] International Telecommunication Union. *Models for Forecasting International Traffic*, 1993. E.507.

[Jac99] Ivar Jacobsen. Applying UML in the Unified Process. Technical report, Rational Software Inc., 1999. `http://www.rational.com`.

[Jes99] Eike Jessen. The DFN Gigabitwissenschaftsnetz G-WiN. Technical report, Deutsches Forschungsnetz Verein / Technische Universität München, January 1999.

[JGJ97] Ivar Jacobsen, Martin Griss, and Patrik Jonsson. *Software Reuse: Architecture Process and Organization for Business*. ACM Press, 1997.

[Joh97] Ralph E. Johnson. Frameworks = Components + Patterns - How Frameworks compare to other object-oriented Reuse Techniques. *Communications of the ACM*, 40(10):39–42, October 1997.

[JRF99] Eike Jessen, Gernot Riegert, and Volker G. Fischer. Projekt Systemplanung für das Breitband-Wissenschaftsnetz - Abschlussbericht. Technical report, Technische Universität München, Institut für Informatik, August 1999.

[Jun87] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen*. Bibliographisches Institut, Mannheim, 1987.

[JV87] Eike Jessen and Rüdiger Valk. *Rechensysteme: Grundlagen und Modellbildung*. Springer-Verlag Berlin Heidelberg New York, 1987.

[Kau81] Joseph S. Kaufman. Blocking in a Shared Resource Environment. *IEEE Transactions on Communications*, 29(10):1474–1481, 1981.

[Kel96] Frank Kelly. Notes on Effective Bandwidth. Technical report, University of Camebridge, 1996.

[Ker93] Aaron Kershenbaum. *Telecommunications Network Design Algorithms*. McGraw-Hill, 1993.

[Kin97] Thomas Kindshofer. Optimierung der Netztopologie in zellulären Mobilfunknetzen unter Berücksichtigung eines bestehenden Netzes. Master's thesis, Technische Universität München, 1997. Institut für Informatik.

[KKG89] A. Kershenbaum, P. Kermani, and G. Grover. MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing. Technical Report RC 14764 7/14/89, IBM Research Division, T.J. Watson Research Center, 1989.

[Kle75] Leonhard Kleinrock. *Queuing Systems - Volume I and Volume II*. John Wiley & Sons, Inc., 1975.

[Kno89] Paul Knottnerus. Forecasting: Kalman Filtering and Prediction Intervals. *International Teletraffic Conference*, (12):277–284, 1989.

[Kru96] Sven Olver Krumke. *On the Approximability of Location and Network Design Problems*. PhD thesis, Bayerische Justus-Maximilian-Universität Würzburg, 1996.

[Ku95] Anne Ku. *Modeling Uncertainty in Electricity Capacity Planning.* PhD thesis, University of London, 1995.

[KW97] Joseph Kallrath and John M. Wilson. *Business Optimization using Mathematical Programming.* MacMillan Press Ltd., 1997.

[KW98] George J. Klir and Mark J. Wierman. *Uncertainty-based Information - Elements of Generalized Information Theory.* Physica-Verlag, 1998.

[Lan99] Daniel Lang. Plazierung von Backboneknoten mit Fuzzy-Clustering. Master's thesis, Technische Universität München, 1999. Institut für Informatik.

[LH92] Jean-Francois P. Labourdette and George W. Hart. Blocking Probabilities in Multitraffic Loss Systems: Insensitivity, Asymptotic Behaviour and Approximations. *IEEE/ACM Transactions on Networking*, 40(8):1355–1366, August 1992.

[Lin94] Karl Lindberger. Dimensioning and Design Methods for Integrated ATM Networks. *International Teletraffic Conference*, 14:897–906, 1994.

[Lor00] Veith Lorenz. Implementierung und Evaluierung von Suchverfahren aus dem Bereich der KI bei der Topologieplanung von paketorientierten Weitverkehrsnetzen. Master's thesis, Technische Universität München, Institut für Informatik, 2000. work in progress.

[LTWW94] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar Nature of Ethernet Traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[MB90] Marvin Mandelbaum and John Buzacott. Flexibility and Decision Making. *European Journal of Operational Research*, 44:17–27, 1990.

[Mel00] Heike Melcher. Moderne Heuristische Suchverfahren in der Netzplanung. Master's thesis, Technische Universität München, Institut für Informatik, 2000.

[Min93] Daniel Minoli. *Broadband Network Analysis and Design.* Artech House Inc., 1993.

[MP94] Yishay Mansour and David Peleg. An Approximation Algorithm for Minimum-Cost Network Design. Technical report, Computer Science Department, Tel-Aviv University, Israel, 1994.

[MS81] Andranik Mirzaian and Kenneth Steiglitz. A Note on the Complexity of the Star-Star Concentrator Problem. *IEEE Transactions on Communications*, COM-29(10):1549–1552, October 1981.

[MS99] Daniel Minoli and Andrew Schmid. *Internet Architectures.* John Wiley & Sons, Inc., 1999.

[MVZ91] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenious. Robust Optimization of large-scale Systems: General modeling Framework and Computations. Technical Report 91-06-04, Department of Decision Sciences, The Wharton School, University of Pennsylvania, 1991.

[MZ94] Scott A. Malcom and Stavros A. Zenios. Robust Optimization of Power Systems Capacity Expansion under Uncertainty. *Journal of the Operations Research Society*, 45(9):1040–1049, 1994.

[NS99] Jörg Noack and Bruno Schienmann. Objektorientierte Vorgehensmodelle im Vergleich. *Informatik-Spektrum*, (22):166–180, 1999.

[NW88] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.

[Odl98a] Andrew Odlyzko. Data Networks are lightly utilized, and will stay that Way. Technical report, At&T Labs - Research, 1998.

[Odl98b] Andrew Odlyzko. The Internet and other Networks: Utilization Rates and their Implications. Technical report, At&T Labs - Research, 1998.

[Odl98c] Andrew Odlyzko. The low Utilization and high Costs of Data Networks. Technical report, At&T Labs - Research, 1998.

[Oes97] Bernd Oesterreich. *Objekt-Orientierte Softwareentwicklung mit der Unified Modelling Language*. Oldenbourg Verlag, 1997. 2. Auflage.

[OMG97] OMG - Object Management Group. *UML Notation Guide - Version 1.1*, September 1997.

[PE96] H. G. Perros and K. M. Elsayed. Call Admission Control Schemes: A Review. *IEEE Communications Magazine*, pages 82–91, Nov 1996.

[PF95] Vern Paxson and Sally Floyd. Wide Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 6 1995.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization - Algorithms and Complexity*. Dover Publications Inc., 1982.

[PW82] C. D. Pack and B. A. Whitaker. Kalman Filter Models for Network Forecasting. *The Bell System Technical Journal*, 61(1):1–14, January 1982.

[Rar95] Ronald L. Rardin. Tutorial: Heuristic Optimization. Technical report, Purdue University, May 1995. Presented to the Industrial Engineering Research Conference, Nashville.

[Ree93] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications, 1993.

[Res98] Mauricio C. Resende. Greedy Randomized Adaptive Search Procedures (GRASP). Technical report, AT&T Labs Research, December 1998. 98.41.1.

[RFC81] Internet Protocol. Technical report, Internet Engineering Task Force - Network Working Group, September 1981. RFC 791.

[RFC88] Routing Information Protocol. Technical report, Internet Engineering Task Force - Network Working Group, June 1988. RFC 1058.

[RFC96] Ipsilon Flow Management Protocol Specification for IPv4 Version 1.0. Technical report, Internet Engineering Task Force - Network Working Group, May 1996. RFC 1953.

[RFC97] Resource ReSerVation Protocol (RSVP) – Version 1: Functional Specification. Technical report, Internet Engineering Task Force - Network Working Group, September 1997. RFC 2205.

[RFC98a] Internet Protocol, Version 6 (IPv6) Specification. Technical report, Internet Engineering Task Force - Network Working Group, December 1998. RFC 2460.

[RFC98b] OSPF Version 2. Technical report, Internet Engineering Task Force - Network Working Group, April 1998. RFC 2328.

[RFC98c] An Architecture for Differentiated Service. Technical report, Internet Engineering Task Force - Network Working Group, December 1998. RFC 2475.

[Ric97] Harald Richter. *Verbindungsnetzwerke für parallele und verteilte Systeme.* Spektrum Akademischer Verlag, 1997.

[Ric98] Harald Richter. *Verbindungsnetzwerke - Strukturen und Leistungsanalysen.* Technische Universität München, Institut für Informatik, Lehrstuhl für Rechnertechnik und Rechnerorganisation, 1998.

[Rie98] Anton Riedl. A Versatile Genetic Algorithm for Network Planning. In *EUNICE'98.* Technische Universität München, September 1998.

[Rim99] Henning Rimkus. Bewertung von Quellenmodellen in Telekommunikationssystemen. Master's thesis, Technische Universität München, 1999. Institut für Informatik.

[RN95] Stuard J. Russel and Peter Norvig. *Artificial Intelligence - A modern Approach.* Prentice-Hall Inc., 1995.

[Ros76] L.M. Rose. *Engineering Investment Decisions - Planning under Uncertainty.* Elsevier Scientific Publishing Company, 1976.

[RSORS96] V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors. *Modern Heuristic Search Methods.* John Wiley & Sons, Inc., 1996.

[Sal98] Edgar Saliger. *Betriebswirtschaftliche Entscheidungstheorie.* R. Oldenbourg Verlag, 1998. 4. Auflage.

[SERW98] Frank Schweitzer, Werner Ebeling, Helge Rosé, and Olaf Weiss. Optimization of Road Networks using Evolutionary Strategies. *Evolutionary Computation*, 5:419–438, 1998.

[SG95] Manfred Schramm and Michael Greiner. Non-Monotonic Reasoning an Probability Models: Indifference, Independence & MaxEnt. Technical Report TUM-I9509, Technische Universität München, Institut für Informatik, März 1995.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[Sha97] Mhammad Shahbaz. *Zufallsgesteuerte Verfahren zur Topologieoptimierung von Telekommunikationsnetzen.* PhD thesis, Fakultät für Elektrotechnik der Rheinisch-Westfälischen Technischen Hochschule Aachen, 1997.

[Sie96] Rainer Siebenhaar. *Verkehrslenkung und Kapazitätsanpassung in ATM-Netzen mit virtuellen Pfaden.* PhD thesis, Technische Universität München, June 1996. Institut für Elektrotechnik.

[Sie98] Siemens AG. *SICAP - Tools for Telecommunication Network Engineering*, 1998.

[Som92] Léa Sombé. *Schließen bei unsicheren Wissen in der künstlichen Intelligenz.* Vieweg & Sohn Verlagsgesellschaft mbH, 1992.

[SOX99] W3C Discussion Paper. *Schema for Object-Oriented XML 2.0*, 7 1999.

[Spä99] Jan Späth. Mehr Licht! Photonische Netze: die Zukunft der Kommunikationsnetze. *c't Magazin für Computer Technik*, 1:156–167, 1999.

[SS00] Geoff Sutcliffe and Christian Suttner. The TPTP Problem Library for Automated Theorem Proving. `http://wwwjessen.informatik.tu-muenchen.de/~tptp/`, March 2000.

[Sto97] Bjarne Stoustrup. *The C++ Programming Language*. Addison-Wesley Publishing, 3rd. edition, 1997.

[Tah87] Hamdy A. Taha. *Operations Research: An Introduction*. MacMillan Publishing Company, 1987.

[TF97] Roger Tagg and Chris Freyberg. *Designing Distributed and Cooperative Information Systems*. International Thompson Computer Press, 1997.

[TGGP98] Éric D. Taillard, Luca M. Gambardella, Michel Gendreau, and Jean-Yves Potvin. Adaptive Memory Programming: A Unified View of Metaheuristics. Technical report, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, 1998. IDSIA-19-98.

[Ull99a] K. Ullmann. Ausschreibung: G-WiN-Zugangsleitungen von den Anwenderstandorten zum Kernnetz des G-WiN, May 1999.

[Ull99b] K. Ullmann. Ausschreibung: SDH-Dienst auf WDM Technologie für das Gigabit-Wissenschaftsnetz G-WiN, May 1999.

[Wöh90] Gerd Wöhlbier. *Planung von Telekommunikationsnetzen: Band I, Fernsprechnetze*. R. v. Decker's Verlag, G. Schenck, 1990.

[XCG95] Jiefeng Xu, Steve Chiu, and Fred Glover. Using Tabu Search to solve the Steiner Tree-Star Problem in Telecommunications Network Design. Technical report, University of Colorado, November 1995.

[XCG98] Jiefeng Xu, Steve Chiu, and Fred Glover. Optimizing a Ring-based Private Line Telecommunication Network using Tabu Search. Technical report, University of Colorado, January 1998.

[XML98] W3C Recommendation. *Extensible Markup Language (XML) 1.0*, 10 1998. REC-xml-19980210.

Index