A decomposition approach to solving core network design problems

Stefan Jacholke
0000-0002-6639-4116

Dissertation submitted in fulfilment of the requirements for the degree *Master of Engineering* in Computer and Electronic Engineering at the Potchefstroom Campus of the North-West University

Supervisor	Dr MJ Grobler
Co-supervisor	Prof SE Terblanche

Examination October 2017

Stefan Jacholke: *Metro-Ethernet*, Core Network Planning, 2017.

WEBSITE:
http://stefanj.me/

E-MAIL:
stefanjacholke@gmail.com

ABSTRACT

Traditionally, when automated planning is used, network planners solve multilayer core network problems in a top-down manner, solving capacities for the top-most layer, and then using these solved capacities to solve the next lower layer. This results in a suboptimal solution, yielding higher capital expenditure costs.

In this work an exact multilayer network Mixed Integer Linear Programming (MILP) model is developed that integrates multiple layers into a single model. Each layer takes the form of a multicommodity flow problem. The objective is to minimize capital expenditure costs, and the integrated network model is shown to be able to reduce costs. This however aggravates the computational burden, and as such, methods to improve scalability and tractibility are developed. This is done by decomposing the problem as per Benders decomposition and applying column generation. A heuristic warm-start is also developed based on this approach. The performance enhancements are compared to an integrated arc-based formulation.

Advances in Ethernet technologies have resulted in lower cost hardware, scalable interfaces and flexible packet services, and together with Wavelength Division Multiplexing (WDM) facilitates low cost per bit transmissions. In order to demonstrate the flexibility of an integrated multilayer network model, the general model is applied to Ethernet over WDM networks.

Keywords: *MILP, Benders decomposition, Column generation, Core network planning, Ethernet, WDM,*

CONTENTS

ACRONYMS

WDM	Wavelength Division Multiplexing
GCE	Google Cloud Engine
GUI	Graphical User Interface
BWDM	Bi-directional Wavelength Division Multiplexing
DWDM	Dense Wavelength Division Multiplexing
CWDM	Coarse Wavelength Division Multiplexing
SDH	Synchronous Digital Hierarchy
SONET	Synchronous Optical Networking
OADM	Optical add-drop multiplexer
OXC	Optical Cross-Connect
RWA	Routing and Wavelength Assignment
MILP	Mixed Integer Linear Programming
ILP	Integer Linear Programming
LP	Linear Program
IP	Internet Protocol
MEF	Metro Ethernet Forum
EVC	Ethernet Virtual Connection
UNI	User Network Interface
MAC	Media Access Control
ECN	Explicit Congestion Notification
TTL	Time to live
MPLS	Multiprotocol Label Switching
LAN	Local Area Network
VLAN	Virtual Local Area Network
WAN	Wide Area Network
QOS	Quality of Service

ATM	Asynchronous Transfer Mode
SDH	Synchronous Digital Hierarchy
MAN	Metropolitan Area Network
OADM	Optical Add Drop Multiplexer
CE	Customer equipment
AP	Access Point
UNI	User-to-network interface
LSP	Label switched path
LSR	Label switched router
DA	Destination address
SA	Source address
SHR	Self-healing Ring
MEN	Metro Ethernet Network
TSP	Travelling Salesman Problem
DAG	Directed Acyclic Graph
DFS	Depth First Search
BFS	Bread First Search
API	Application Programming Interface
SATNAC	Southern Africa Telecommunication Networks and Applications Conference

1 INTRODUCTION

1.1 INTRODUCTION

Metro and core network providers are faced with ever-growing traffic demands and consequently have to extend and upgrade their networks [1–3].

Advances in Ethernet technologies have resulted in lower cost hardware, scalable interfaces and flexible packet services, and together with WDM facilitates low cost per bit transmissions.

Ethernet has developed to be the dominant technology in Metro networks, and the Metro Ethernet Forum (MEF) is devising Ethernet standards in order to replace traditional technologies such as Synchronous Digital Hierarchy (SDH) and Synchronous Optical Networking (SONET) [4,5]. In addition to the upper layer being Ethernet, on the physical layer WDM is used with optical fiber as this reduces fiber requirements. WDM allows multiple signals to be transmitted over a single fiber by multiplexing multiple input signals onto the fiber, each on a different wavelength.

Modern telecommunications networks are designed according to a layered structure, with different layers encompassing different technologies.

When planning a multilayer network, operators commonly proceed in a top-down fashion. The top-most layer is solved for the capacity requirements, satisfying the demand requirements and other network constraints. These solved capacities are then used as demands in the next lower layer.

When solving a tradtional Internet Protocol (IP) network in a top down approach, the IP layer is separated from the other layers. The capacity of the IP layer is planned according to the traffic matrix of the required IP network. The operator then determines the required bottom transport network based on the IP layer. According to HUAWEI [6], this approach leads to wasted resources, difficulty in meeting disjoint routing requirements, complicated network structure for future networks and an unnecessary increase in network development costs.

The alternative approach is to model the network intrinsically as multilayered. The integrated approach would contain hardware and routing relations in each layer and include the layer interdependencies in a single model. This has the disadvantage of being computationally intensive, and up until recently has not been viable.

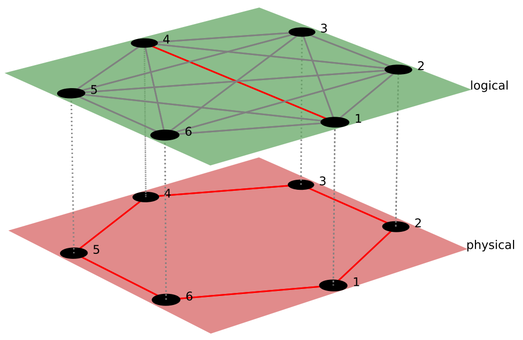


Figure 1: A simple two layer network

A common approach is to model these problems as Mixed Integer Linear Programming (MILP), which provides us with the best possible answer when minimizing the network cost. However the general class integer programming problems is NP-hard [7]. Even as computer technology becomes faster, the solver won't be able to scale to much larger instances.

With the advent of faster processing speeds and increased memory capacity, these integrated multilayer models can be solved, however, due to the complexity of these models, modern solvers still struggle. For this reason, methods are investigated in this thesis in order to improve computational tractability and scalability. In particular, decompositational approaches will be utilized.

This work focusses on minimizing the capital expenditure costs for designing a green Ethernet over WDM fiber network. This approach incorporates the entire network, including all the layers in an integrated fashion, in a MILP model. The model can then be solved using a commercial solver such as IBM ILOG CPLEX [8].

Figure 1 is a depiction of a two-layer network. The red logical link on the top most layer can be realized be sending data either clockwise from node 1 to node 4 or counterclockwise. Depending on the technologies involved the traffic can also be split to occupy both paths. In this work, the physical paths are implicitly generated for each logical link, instead of having to define them explicitly.

We compare the integrated multilayer model with the top-down approach and show that the top-down approach yields suboptimal answers. Network operators may in some instances save significantly on network planning costs. There is a tradeoff, in which a more accurate model increases the complexity, and hence number of constraints.

This work covers three main points:

- Developing a flexible approach to solving multilayer network problems.

- Applying the general approach in order to develop different Ethernet over WDM network models.
- Improving the scalability and performance of integrated multi-layer network models by applying Bender's decomposition and column generation.

1.2 MOTIVATION

When planning a multilayer network, operators commonly proceed in a top-down fashion. The top-most layer is solved for the capacity requirements, satisfying the demand requirements and other network constraints. These solved capacities are then used as demands in the next lower layer.

This approach is problematic due to the following [6, 9]:

- Minimum Cost - Difficult to approximate the cost of a logical link if the realization on the physical layer is not known.
- Survivability - A demand routed over a logically disjoint path may not be disjoint on the physical layer.
- Routing - Uncoordinated routing between layers may result in the top-most demand being routed several times over the physical layer.

As an example to the first point, when sequentially planning the network, we do not know what the physical representation will be when planning the logical layer. This is problematic because the actual physical costs may be much higher than the logical costs, leading to sub-optimal minimization of cost.

With an integrated multilayer approach, the interdependencies are combined into a single formulation. This approach is computationally expensive, however, it avoids the problems presented above. The use of MILP technology allows us to optimally solve the problem, or if necessary, explicitly provide us with an optimality gap when prematurely terminated.

The Ethernet over WDM network presented in this paper is modeled using a MILP formulation. Furthermore, this work deviates from previous work in the literature (such as [10]) by considering all possible paths implicitly, against having the physical path prespecified for each logical link. Although this is computationally more expensive it may improve the cost. The model is also formulated to represent the network explicitly as a two-layer network, allowing for a separate logical topology from the physical topology.

In order to reduce the performance and scalability divide between the traditional approach, and the integrated multilayer approach, we consider decomposition techniques.

1.3 METHODOLOGY

The relevant work in the field of network planning is considered in order to determine the current state of research in multilayer network planning, particularly using MILP techniques.

A generic modular multilayer network model is developed in an iterative manner. The model is decomposed in order to improve scalability. The improved model is verified with the original on smaller network instances. This generic model allows the capacity and cost of network equipment to be specified in terms of modules and is suited to a variety of network types and technologies. Survivability is added to the model in terms of 1+1 protection in order to improve network robustness.

The general model is modified to apply to WDM over Ethernet networks. A variety of different WDM and Ethernet network models are considered, each with a different use case and specifications. The WDM routing and wavelength assignment problem is considered as well. Decomposition techniques are applied to these extended network models as well in order to improve scalability.

The performance of different computational techniques and decomposition approaches on multilayer networks are investigated in order to determine their viability as well as performance.

Lastly we validate and verify the results delivered by the network models.

1.4 VALIDATION AND VERIFICATION

In this work multilayer network models are developed, based on multicommodity flow problems.

Unfortunately there are no tests that can be applied to determine whether a certain model is correct. No general procedure can be applied, as it is context dependent. Developing a simulation relies on understanding the underlying phenomena.

In order to verify the correctness of the models, we proceed as follows:

- Investigate the underlying hardware used in Ethernet and WDM networks, as well as the the different types of topologies.
- Investigate the common approaches to modelling network flow.
- Implement a multilayer network model that generalizes the above two points.
- Verify that the implementation of the model is correct. This is accomplished by developing two implementations and comparing the results attained.

- Unit testing - Test the over a range of input parameters and compare with the correct answer.
- Face validity - Determine whether the input output relationship of the model is acceptable. A Graphical User Interface (GUI) web interface is developed that can be used to plan multilayer network interactively. The tool visually displays the potential input network as well as the solution. When determining survivability, we ensure that there exists a backup path.

This framework is based on suggestions by Carson [11] and Sargent [12].

We use the Haskell language in order to help ensure the implementation is robust and correct. Haskell has strong static typing[1] based on Hindley-Milner. In addition variables are immutable and functions are pure[2], which helps avoid many problems encountered in concurrency and parallelism. Unit tests are also developed for specific parts of the CPLEX-Haskell library, as well as certain parts of models.

In order to ensure that the mathematical models are correct, small numerical models are worked out, in order to ascertain whether the output makes sense for the given input. This also helps ensure the implementation is correct. This also forms part of the face validation.

1.5 CONTRIBUTIONS

Some of the work mentioned here was featured in Southern Africa Telecommunication Networks and Applications Conference (SATNAC):

- Initial multilayer network model, work in progress paper, Development of a Multi-Layer Model for Optimal Core Ethernet Resource Planning [13].
- A multilayer approach for solving the Ethernet over WDM network design problem [14].

1.6 OVERVIEW

Chapter 2 aims to familiarize the reader with the relevant mathematical techniques and algorithmic approached employed in the paper.

Chapter 3 briefly covers the relevant technical background such as network hardware and architecture relevant to core networks. The chapter then covers related work and relevant research.

1 The compiler of a language with static typing has a static type checker that analyzes the program to ensure the program satisfies some type safety properties. This is a limited form of program verification

2 The return value of a pure function is only determined by its inputs

In Chapter 4 we develop a simple generic single layer network model using the standard multicommodity flow approach. The single-layer model is then extended to a generic multilayer model that incorporates demand routing and hardware constraints. The model is then extended to cover survivability.

In Chapter 6 the model is extended to cover different WDM over Ethernet networks. We model the logical layer as Ethernet with WDM with wavelengths that can be installed over The model includes survivability constraints and the routing wavelength and assignment is solved separately. A separate model is developed that emulates the traditional layer-by-layer approach in order to demonstrate the gap from optimality.

In Chapter 5 computational techniques are developed in order to improve the scalability of the problem. In particular, Bender's decomposition is applied in order to reduce the number of constraints and column generation is used in order to reduce the number of variables currently in the basis. A simple primal heuristic based on Bender's decomposition is used in order to find a fast upper bound for the problem. We see that some of the cuts can be rounded. To the author's knowledge, this is the first work that uses a Bender's framework in order to solve a path-based multilayer model.

In Section 6.3 we discuss the results obtained by comparing the top-down model with the integrated multilayer Ethernet over WDM model. This comparison focusses on evaluating the difference in capital expenditure costs between the two approaches, reducing costs for network planners.

In Section 5.4 we compare the performance benefits from using Benders decomposition with column generation over a generic arc-based model. We also investigate some other techniques to improve the performance, such as using a warm-start (improving the upper bound) and strengthening the Benders cuts (improving the lower bound).

2 PRELIMINARIES

This chapter introduces the basic notions used in the process of network planning and presents some fundamental ideas behind it.

2.1 GRAPH THEORY

In network planning key concepts from Graph Theory are used. In this section, we will briefly review some of these concepts and the notation that will be used in the remainder of this thesis. Additionally, we fix the terms and definitions used[1].

Definition 2.1.1 (Graph). *A graph* $G = (V, E)$ *consists of a set of vertices* V *and a set of pairs of elements of* V *representing the edges* E.

A *simple graph* has no loops or parallel edges in the same direction. We will mainly be concerned with simple graphs.

A *directed graph* consists of a set of vertices V and a set of directed edges E where the elements of the edges are ordered pairs of vertices.

Definition 2.1.2 (Walk). *A walk is a finite sequence of the form:*

$$v_{i0}, e_{j1}, v_{i1}, e_{j2}, \ldots, e_{jk}, v_{ik}$$

A walk is called open if $v_i0 \neq v_{ik}$, otherwise it is called close.

Definition 2.1.3 (Trail). *A walk is called a trail if any edge is traversed at most once.*

Definition 2.1.4 (Path). *A trail is a path if any vertex is visited at most once.*

Definition 2.1.5 (Connected Vertices). *Vertices* u *and* v *are said te be connected if there exists a walk that starts at* u *and ends at* v.

From this it follows that connections are transitive, that is if u and v are connected, and v and w are connected, then u and w are connected.

Definition 2.1.6 (Connected Graph). *A graph* G *is called connected if all of the vertices of* G *are connected.*

Definition 2.1.7 (Component). *The subgraph* G_1 *of graph* G *is a component of* G *if* G_1 *if:*

1 In order to avoid ambiguity, as many authors have different definitions

- G_1 *is connected*
- G_1 *is trivial or* G_1 *is the subgraph induced by edges in* G *that have a end vertex in* G_1

Definition 2.1.8 (Circuit)**.** *A closed path is called a circuit.*

Definition 2.1.9 (Cycle)**.** *A closed trail is called a cycle. A Hamiltonian cycle is a cycle that visits every node exactly once.*

We present these graph theoretic terms as definitions, however in the context of networks the same words may be reused with a looser definition. This will be pointed out when necessary.

We start with the following, we present the following terms and their relations with the stricter graph-theoretic version.

A *network* is a simple graph or digraph. The *edges* of a network are undirected and correspond the edges of a graph. The *arcs* of a network are similarly directed and correspond to the arcs of a graph. A *node* or *vertex* corresponds to a location in the network. Hence *edges* and *arcs* correspond to a connection between nodes. The term *link* is commonly used to denote a connection on the logical layer of the network and a *edge* is commonly used to denote a connection on the physical layer of the network.

2.2 TIME COMPLEXITY

The time complexity of an algorithm is a measure of the amount of time taken by the algorithm as a function of the length of the input. In this work, we use Big O notation in order to indicate the worst-case running time. Formally one writes:

$$f(n) = O(g(n)), \qquad \text{as } n \to \infty$$

if and only if there exists some number M such that:

$$|f(n)| \leqslant M|g(n)|, \qquad \forall n \geqslant n_0$$

indicating that $f(n)$ is less than some multiple of $g(n)$, i.e. it is bounded.

Algorithms can be classified depending on their running time. An algorithm is said to take constant time ($O(1)$) when $f(n)$ does not depend on the size of the input. An algorithm is said to take logarithmic time when $f(n) = O(\log n)$. A linear time algorithm has time complexity $O(n)$. A polynomial time algorithm is bounded by a polynomial expression, that is $f(n) = O(n^k)$.

The variable n representing the size of the input is taken to mean the number of bits required to represent the input. Hence if the input to an algorithm is a list of n $32\mathtt{bit}$ numbers, then the number of bits would be $x = 32n$. For algorithms that operate on arrays or adjacency lists the distinction is not noted, however it breaks down when an algorithm operates on numbers.

A prime number n is a natural number greater than 1, that only has two divisors, 1 and n

Consider the naive[2] algorithm given by Algorithm 1 that computes whether a number is prime or composite.

Algorithm 1 Naive algorithm to compute whether a number n is prime

```
ISPRIME(N)
1  for $i \in \{2, 3, \dots, n-1\}$
2      if $n \bmod i = 0$
3          return False
4  return True
```

Deceptively, one could think that this algorithm runs in polynomial time, since the for loop runs in $O(n)$ and the amount of work inside the loop is at most polynomial as well, thus one could think that this algorithm runs in $O(n^k)$ time, however the n used here is the numeric value of the number, and not the number of bits used to represent the number. Since $n = 2^x$ we actually have $O(2^{kx})$. Thus the naive prime-checking[3] algorithm actually runs in pseudopolynomial time.

An algorithm is said to run in *pseudopolynomial* time if the runtime is polynomial in the numeric value of the input, but exponential in the number of bits of the input. Another example is the Knapsack dynamic programming algorithm [15].

Theoretically big O describes only an upper bound. An algorithm that runs in $O(n)$ is also $O(n^2), O(n^3), O(2^n)$ and so on. Practically, we sometimes want to know what the lowest upper bound is.

Similarly, the lower bound is given by Ω, thus an algorithm that runs in $\Omega(n)$ is also $\Omega(\log(n))$ and $\Omega(1)$.

Θ is used when an algorithms upper bound and lower bound is the same. Thus an algorithm is $\Theta(n)$ when it is $\Omega(n)$ and $O(n)$.

In practice we tend to use the tightest big O bound, which is closer Θ.

Algorithms may belong to different complexity classes. These classes are commonly explained using the concept of a Turing Machine.

A *Turing Machine* is a machine that executes a tape of instructions and contains infinite memory. The machine stores the current state, and the next state is determined by the current state and the next instruction to be read from the tape. A deterministic Turing Machine may only advance to a single unique state after each instruction. In

2 There are many improvements that can be made, such as only checking up to $\sqrt{n}$

3 The AKS primality test is the first algorithm to check whether a number is prime in actual polynomial time

contrast, a non-deterministic Turing Machine may advance to multiple states after each instruction, simultaneously. A deterministic Turing Machine can be thought of as an idealistic computer with infinite memory. It is a structure which allows computation, it does not specify additional details such as input and output. The executation time of Turing Machines are used when determining complexity classes.

A complexity class contains a set of problems with similar resource complexity; a set of problems that take a similar range of space or time to solve. Problems are proven to be in a complexity class using an abstract model of computation, such as a Turing Machine. There is a large number of complexity classes, and the interested reader can refer to Aaronson's complexity zoo [16]. In this work we are mostly concerned with problems in P, NP, NP-Complete and NP-Hard:

- The class P contains all decision problems solvable using a polynomial amount of computation time.
- The class NP contains all decision problems for which the answer can be verified by deterministic computations in polynomial time. Equivalently said, the problem, only needs to be solvable in polynomial time by a non-deterministic Turing machine.
- A problem x in NP, is said to be in NP-Complete if and only if every problem other than x in NP can be reduced into x, in polynomial time.
- A problem x is said to be in NP-Hard if and only if every algorithm in NP can be reduced in polynomial time to x. Note that x need not be in NP.

When we say reduce, we mean to apply a reduction. A reduction is an algorithm for transforming one problem into another. Formally A is reducible to B under F if:

$$\exists f \in F \,.\, \forall x \in \mathbb{N} \,.\, x \in A \iff f(x) \in B$$

Given subsets $A, B \subseteq N$, and F contains the set of functions $f : N \to N$.

Loosely, when given a problem Π, if there exists a polynomial time reduction from Π to Θ, and we know that Θ is in P then we can conclude that Π is in P as well.

Complexity classes are only determined for the domain of decision problems, that is, a question in some system that can be answered binary, either yes or no dependent on the input. Most problems, including the problems presented in this thesis, are not decision problems, but optimization problems. In contrast to a decision problem, an optimization problem has the goal of finding the best possible answer for the given input.

There are standard reductions for transforming an optimization into a decision problem, in most cases, when minimizing, we can ask whether a solution exists that is at most K. Thus a bound is imposed on the value to be optimized.

When talking about the time complexity of an optimization problem, what we really refer to, is the time complexity of the equivalent decision problem.

2.3 LINEAR PROGRAMMING

The problem of solving linear inequalities dates at least back to Fourier. The Fourier-Motzkin elimination method is a method for eliminating variables from a system of linear inequalities [17]. Linear programming was only developed later, in order to obtain the best outcome in a model subject to a certain criterion. The outcome and criterion has to be linear.

Linear programming is an optimization technique whereby a problem is described by a linear objective function and linear inequality constraints. A linear programming problem maximizes or minimizes a linear function over a convex polyhedron[4] that is specified by linear constraints.

Linear programs are usually expressed in symmetric form as:

$$\begin{array}{ll} \text{maximize} & c^T\vec{x} \\ \text{subject to} & Ax \leqslant \vec{b} \\ & \vec{x} \geqslant \vec{0} \end{array}$$

The objective function is a linear combination that is maximized or minimized, and is subject to a set of constraints.

The original problem is commonly called the primal problem, and the variables will be denoted primal as well.

Von Neumann introduced the theory of duality by relating linear programming to his own work in game theory. Given a linear program that is in symmetric form, we obtain the dual as:

$$\begin{array}{ll} \text{minimize} & b^T\vec{y} \\ \text{subject to} & A^T\vec{y} \geqslant \vec{c} \\ & \vec{y} \geqslant \vec{0} \end{array}$$

which contains dual variables, each corresponding to a row in the constraints of the primal problem. Likewise, each variable in the primal problem corresponds to a constraint in the dual problem.

Theorem 2.3.1 (Weak Duality). *For any feasible solution $\vec{x}$ for the primal problem, and $\vec{y}$ for the dual problem we have that:*

$$c^T\vec{x} \leqslant b^T\vec{y}$$

4 In a convex polyhedron all interior angles are less than or equal to 180 degress

From which we can see that if the optimal objective value in the primal tends to become infinitely large then the dual problem is not feasible.

Theorem 2.3.2 (Strong Duality)**.** *The primal problem has an optimal solution if and only if the dual problem does. Given optimal solutions* $\bar{x}$ *and* $\bar{y}$ *for the primal and dual respectively, then* $c^T\bar{x} = b^T\bar{y}$

Lemma 2.3.1 (Farkas' Lemma)**.** *For* $A \in \mathbb{R}^{M\times N}$ *and* $b \in \mathbb{R}^M$ *then only one of the following statements are true:*

1. *There exists a vector* $\vec{x} \in \mathbb{R}^N$*, such that* $\vec{x} \geqslant 0, A\vec{x} = \vec{b}$
2. *There exists a vector* $\vec{y} \in \mathbb{R}^M$*, such that* $\vec{b}^T\vec{y} < 0$ *and* $A^T\vec{y} \geqslant 0$

The proofs are not presented here, but the interested reader can refer to popular texts such as [18–21]

2.4 SIMPLEX METHOD

Dantzig invented the simplex method in order to solve these linear programming models generally [22].

For a linear problem in the standard form

$$\begin{aligned} \max \quad & c^T\vec{x} \\ \text{s.t.} \quad & Ax \leqslant \vec{b} \\ & \vec{x} \geqslant \vec{0} \end{aligned}$$

The feasible region is defined by

$$\text{s.t.} \quad Ax \leqslant \vec{b}, \qquad \vec{x} \geqslant \vec{0}$$

and is a convex polytope. For a linear program in standard form, the objective function has the maximal value inside the feasible region, and in particular, the maximal value is on the extreme points of the polytope.

When the objective function on an extreme point is not maximal, -there exists an edge containing this point such that the objective function is strictly increasing on the edge away from the point [21].

The original simplex method has been shown to be *exponential time*, however, it is efficient in practice [18]. More efficient versions of the simplex algorithm exist, however, the ellipsoid algorithm was the first algorithm to show worst-case polynomial time for linear programming problems [23].

The precise steps can be found in [18]. Modern solvers implement an improved version of the simplex method. Interior point methods solve in polynomial time, although in practice the simplex method is faster for most problems.

2.5 MIXED INTEGER PROGRAMMING

Linear programming allows us to determine the existence of optimal solutions; if the feasible region of a problem is a convex polyhedron then for a given convex objective function, the local minimum is the global minimum[5].

Linear programs can be solved efficiently, however, a large class of problems cannot be modeled as such. Integer programming requires that the decision variables be integral. Mixed integer programming relaxes the need for variables to be only integral and allows variables to be continuous as well. The branch and Bound method explores the integer state space by constructing a tree [24], where the integrality condition is first relaxed, and then bounded. Once bounded the problem is solved as a Linear Program (LP) problem. This allows a larger number of problems to be solved, however integer programming as well as mixed integer programming is shown to be NP-hard [7].

An Integer Linear Programming (ILP) in canonical form is expressed as:

$$\begin{array}{rc} \max & c^T\vec{x} \\ \text{s.t.} & A\vec{x} \leqslant \vec{b} \\ & \vec{x} \geqslant \vec{0} \\ & \vec{x} \in \mathbb{Z}^n \end{array}$$

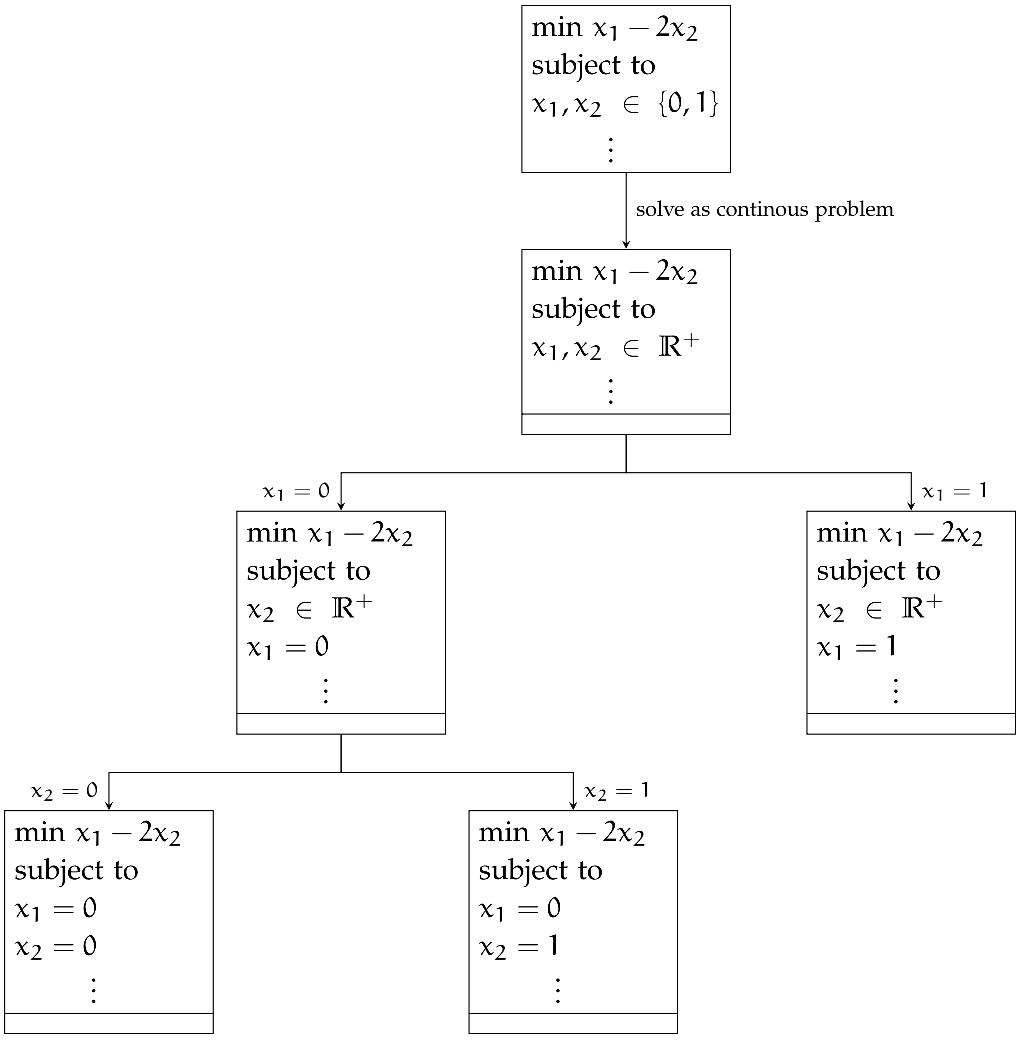
The special form MILP is obtained when only some of the variables are constrained to be integer.

$$\begin{array}{rc} \max & c^T x + f^T y \\ \text{s.t.} & A\vec{x} + B\vec{y} \leqslant b \\ & \vec{y} \geqslant 0 \\ & \vec{x} \geqslant 0 \\ & \vec{x} \in \mathbb{Z}^n \end{array}$$

2.6 BRANCH AND BOUND

Branch and Bound is an algorithmic paradigm that tries to find candidate solutions by searching through the search space in a systematic manner [24]. The enumeration through the search space happens in a tree like structure. The full solution space occurs at the root node of the tree. Each branch of the tree constrains the problem and represents a subset of the solution space; thus each node represents the original problem with additional constraints. When enumerating possible solutions, the branch is checked against the lower bound and

5 Likewise, the local maximum is the global maximum given a concave function



min $x_1 - 2x_2$ subject to $x_1, x_2 \in \{0,1\}$ ⋮
solve as continous problem
min $x_1 - 2x_2$ subject to $x_1, x_2 \in \mathbb{R}^+$ ⋮
$x_1 = 0$
$x_1 = 1$
min $x_1 - 2x_2$ subject to $x_2 \in \mathbb{R}^+$ $x_1 = 0$ ⋮
min $x_1 - 2x_2$ subject to $x_2 \in \mathbb{R}^+$ $x_1 = 1$ ⋮
$x_2 = 0$
$x_2 = 1$
min $x_1 - 2x_2$ subject to $x_1 = 0$ $x_2 = 0$ ⋮
min $x_1 - 2x_2$ subject to $x_1 = 0$ $x_2 = 1$ ⋮

Figure 2: Binary Branch and Bound Example

upper bound of the optimal solution, if the branch cannot produce a better solution than the current best, it is discarded.

The Branch and Bound pattern is used in many different algorithms. For the work considered here Branch and Bound is employed for solving ILP and MILP problems. Since the Simplex algorithm is a good fit to solve continuous LP problems, the task is to reduce a MILP to a continuous linear programming problem that can be solved using the Simplex method. In order to do so, each variable is constrained in a branch; the branch variable is constrained to a specific value. When all of the integer variables are fixed to a certain value, the branch problem can be solved as a simple LP. Figure 2 shows an example of a binary integer programming problem. At each node a variable is constrained to a binary (or integer) value.

2.7 BENDERS DECOMPOSITION

Benders decomposition is a technique for solving linear programming problems that have a block structure. The problem is divided into two subsets, the reduced master problem and the subproblem. The solution of the reduced master problem is used in the subproblem. If the subproblem determines the decisions are infeasible, then Benders cuts are added to the master problem and the problem is resolved until no cuts can be added.

According to [25], for a mixed integer programming problem in the following format:

$$\begin{array}{rc} \min_x & c^T x + f^T y \\ \text{subject to} & Ax + By \geqslant b \\ & y \in Y \\ & x \geqslant 0 \end{array}$$

The minimization problem can also be written as

$$\min_{y \in Y} \left[f^T y + \min_{x \geqslant 0} \{ c^T x | Ax \geqslant b - By \} \right]$$

And the dual of the LP is given by:

$$\begin{array}{rc} \max_u & (b - By)^T u \\ \text{subject to} & A^T u \leqslant c \\ & u \geqslant 0 \end{array}$$

The algorithm then proceeds as follows. If the difference between the upper bound and the lower bound is some positive number ϵ, we solve the subproblem (which is a LP). From the LP we obtain the solution of the dual (which is easy to obtain using most solvers). The benders cut $z \geqslant (b - By)^T u$ is then added to the master problem. The problem is then resolved. This is repeated until the difference between the upper bound and lower bound is nonzero.

Thus a problem can be subdivided if it has a block-like structure. In most of the cases in this work, $Y = \mathbb{Z}$, that is the problem is partitioned by keeping constraints containing integers in the master problem, and solving the subproblem as a LP.

2.7.1 Example

A simple problem is chosen in order to demonstrate the concept. The integer variables will be separated.

The overall problem is to

$$\min \quad y_1$$

s.t.

$$3x_1 + 7x_2 > 3y_1$$

$$x_1 + x_2 > 10$$

$$y_1 \in \mathbb{Z}$$

$$x_1, x_2 \in \mathbb{R}$$

The problem can be split into two parts, the integral reduced master problem, and the real subproblem.

The goal of the reduced master problem is to

$$\min \quad y_1 + z$$

$$z \in \mathbb{R}$$

and the goal of the subproblem is to

$$\min \quad 0x_1 + 0x_2,$$

The subproblem is also subject to

$$3x_1 + 7x_2 > 3y_1^*$$

$$x_1 + x_2 > 10$$

where y_1^* is the solved integer variable obtained from the master problem. This can be obtained at a branch and bound node.

The primal subproblem has the dual form of

$$\max \quad 3y_1^* a_1 + 10a_2,$$

s.t.

$$3a_1 + 1a_2 < 0$$

$$7a_1 + 1a_2 < 0$$

The Benders feasibility cut to add each iteration to the reduced master problem, is then obtained as,

$$z \geqslant 3y_1 a_1^* + 10a_2^*,$$

where $a_1^* \in \mathbb{R}^*, a_2^* \in \mathbb{R}^*$ are the solved solution variables of the dual LP subproblem.

2.8 LOCAL SEARCH

MILP may provide the optimal solution for linear programming problems with integer variables, however the running time may be unacceptable for large instances. Local search is a heuristic method for solving optimization problems. These methods cannot guarantee optimality, however they may find an initial solution quicker, or solve larger problem instances easier. Local Search algorithms are also employed when there is a combinatorial explosion, as is common with problems in the NP classes.

Local search algorithms include (but are not limited to):

1. Gradient Descent [26] - Finds the local minimum, by taking steps in proportional to the negative of the gradient of the objective function.

2. Simulated Annealing [27] - Tries to find the state with least amount of energy. The algorithm tries to avoid some local minima in order to approximate the global optimum, it does so by accepting worse solutions, however the probability of doing so decreases over time.

3. Genetic Algorithms [28] - A metaheuristic based on the idea of passing over genes and natural selection. A population of candidates are created, with each candidate representing a possible solution. Candidates are subject to genetic operates which may modify and alter them. Solutions are kept in the population based on their fitness[6].

Another class of local search algorithms would be expert system algorithms. These algorithms try to emulate what a human expert would do. While many local search algorithms such as Genetic Algorithms are known as black box algorithms, as we cannot always see how the problem is solved, expert systems are more transparent and are usually comprised of a set of rules.

Local search algorithms and heuristics may be used to find a good initial solution[7] for the MILP problem, and in combination may provide a speedup. A solution is called a warm start solution, when provided to the MILP solver as a starting point. In this work a heuristic based on Benders Decomposition and Column generation is used as a warmstart.

6 The fitness is determined by the objective function
7 The generated solutions would need to be faster, or yield a higher quality solution than the initial solution generated by the solver in order to be of use

2.9 ALGORITHMIC IMPLEMENTATION

The problems presented in this work are modelled as MILP problems, and are solved using a special purpose high performance solver such as CPLEX [8].

There are algorithmic subtleties involved, when doing Benders decomposition and column generation, as a separate LP problem needs to be solved at each Branch and Bound node. Details of this will be expounded in later chapters. Some of the required underpinnings will be presented henceforth in this section.

When solving a specific problem, the formulations are generated programatically. The data is read and parsed, whereafter a MILP is then generated and solved, via an Application Programming Interface (API) that interacts with the solver. For some of the decompositions we require the ability to solve suproblems such as finding the shortest path. Hence we require flexibility and utilize a programmatic framework.

Since for the most part, the problems are based on graphs, we utilize graph algorithms and data structures. Dijkstra's algorithm is covered in more detail in a later chapter, see Algorithm 4.

2.9.1 Graph data structures

Since a graph is a collection of nodes and edges, we need some data structure to store nodes and their connections. Three common ways of storing graphs are:

- Using an adjacency matrix
- Using an adjacency list
- Using objects and pointers

An adjacency matrix is a $N \times N$ boolean matrix where a true value at (i, j) indicates an edge between i and j. In an undirected graph, the matrix will be symmetric.

An adjacency list is an extendable array where $arr[i]$ returns a list of outgoing nodes from i. This is the preferred data structure for sparse graphs. An adjacency list is only filled as necessary, whereas an adjacency matrix would be filled with excessive zeroes.

In terms of space complexity an adjacency matrix takes up space on the order of $O(n^2)$, while an adjacency list takes of $O(n + m)$ where n is the number of nodes and m is the number of edges.

For many search algorithms the list is more efficient, as in the matrix the node's row needs to be iterated through in order to find all of its neighbors.

Objects and pointers represent the graph as an object. Each node contains references to its children. This approach is not commonly used, as it is cumbersome to obtain an arbitrary node's neighbors. When using this approach, one would first need to traverse the graph up to a certain node, in order to find its neighbors. Random access representations are typically preferred.

2.9.2 Lookup

A graph G is determined by the pair (V, E). A weight of an edge is some quantity. It can be the length of the edge (the distance between the edge end nodes), or it could be a monetary cost associated with the edge or other measuruble quantities. These weights are stored separately. Since an edge can be represented as a tuple (i, j) we can use a data structure with fast lookup, either a search tree or a hash table. A balanced binary search tree provides a lookup time of $O(\log(n))$ [29]. A hash table provides a worse time lookup of $O(n)$, however the amortized (averaged) lookup time is $O(1)$[8].

In a search tree, the key for each node is greater than the keys of the subtrees on the left, and less than those on the right. Thus a binary search algorithm is commonly used to quickly lookup a key, running in $O(\log(n))$ time when the tree is balanced. In general we simply need a preference $(\prec)$ between two elements in order to build a tree.

A hash table maps keys to values by using a hash function. The hash function assigns each key to a unique bucket. Sometimes the same hash is obtained for different keys, resulting in a collision, in this case two (or more) values will occupy a bucket, this results in a worse time lookup of $O(n)$, though in practice a good hash table facilitates lookup times of $O(1)$ on average.

For a tuple of integers $\{(i, j), i \in \mathbb{Z}, j \in \mathbb{Z}\}$ it is possible to define a preference[9], as well as a hash, so both methods can be employed as a method to lookup the weight or length of an edge. In this manner the weight of an edge in the graph can be quickly obtained.

In addition to providing quick lookup times for weights, hash tables can be used when storing MILP index variables and constraints.

2.9.3 Graph search

The two most common way of searching through a graph is with a Depth First Search (DFS), or a Bread First Search (BFS). With a DFS the search first goes deep - it explores all of the children first, recursively, and then broad. A BFS first goes broad and then deep. With the BFS each neighbors is visited, and only afterwards are the children explored. Since BFS searches level by level, it can be used to

8 There is also a cost for hashing, which for tuples would be small, but for items with dynamic input size such as strings a length variable would arise

9 One such preference could be the following, for $a = (i, j), b = (k, l)$ with $i < k$ or if $i = k$ and $j < l$ then we can write $a \prec b$

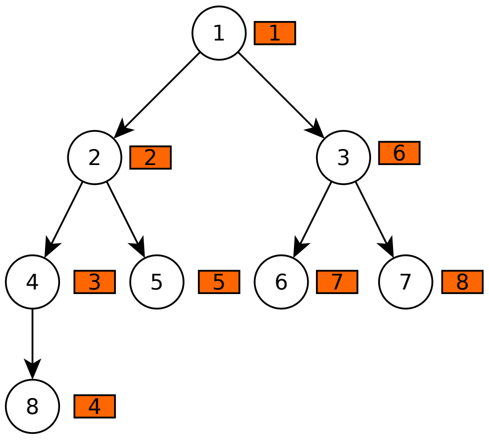


Figure 3: Example of DFS

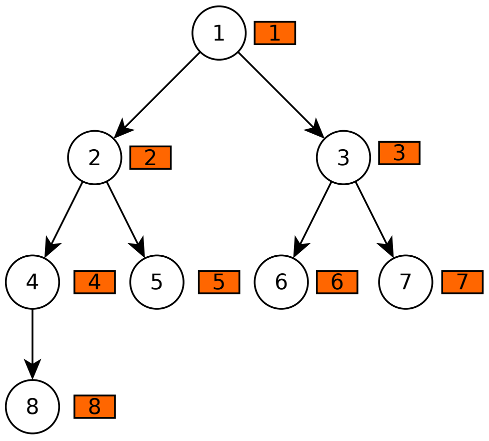


Figure 4: Example of BFS

find the shortest path between two nodes, when distance is considered as the number of nodes hopped.

Both algorithms have a worst case time of $O(|N| + |E|)$, where $|N|$ is the amount of nodes and $|E|$ is the amount of edges.

DFS has a convenient recursive implementation and can be modified to find all paths between two nodes.

Figure 3 shows the order in which a DFS algorithm visits nodes on an example tree, likewise figure 4 shows the order in which a BFS would visit nodes. The orange colored boxes denote the numeric order.

In this work a backtracking graph search algorithm is used to find all paths connecting a b for a commodity $k = (a, b)$. These commodity paths are used in the vanilla path-based formulations.

3 BACKGROUND AND LITERATURE

3.1 BACKGROUND

3.1.1 Ethernet

Ethernet refers to a family of local-area network technologies that is covered by the IEEE 802.3 standard. Data is divided into pieces called frames when transmitted over Ethernet. The frame contains the source and destination address, error checking data, protocol headers and the payload.

Traditionally Ethernet was mostly used in Local Area Networks (LANs). A LAN is a computer network that interconnects devices within a limited area such as an office, laboratory or residence. The transmission speeds of Ethernet is continually improving and is starting to facilitate use cases that require greater distance. Carrier Ethernet is a high-bandwidth Ethernet Technology that provides connectivity to government, business and academic networks.

There are several architectures available to carry Ethernet frames across metro networks, with the two popular approaches in the industry being either using MPLS as the transport technology or extending the native Ethernet protocol (Provider Bridged Networks) [30]. Metropolitan Ethernet is Carrier Ethernet in a metropolitan area network. In addition to the greater speeds provided by Carrier Ethernet, Metro Ethernet employes bandwidth management and other control functionality. Metro Ethernet is used to connect LANs to a WAN.

Ethernet brings with it improved properties such as cost effectiveness, rapid provisioning, ease of interworking and good adoption along with it [31]. The focus of Metro Ethernet is to provide solutions for the shortcomings of Ethernet such that it can be used in the enterprise domain. Some of these shortcomings include lack of Quality of Service (QOS) guarentees, protection mechanisms and performance monitoring.

Most connections between LANs are still performed by a combination of Asynchronous Transfer Mode (ATM) or SDH whereby layer 3 packets are transported. New technologies in Carrier and Metro Ethernet will extend the reach supported of Ethernet and will allow point to point connections. This will yield cost benefits as the network design will be simplified and the number of layers will be reduced, resulting in a more scalable homogeneous Metropolitan Area Network (MAN) [32].

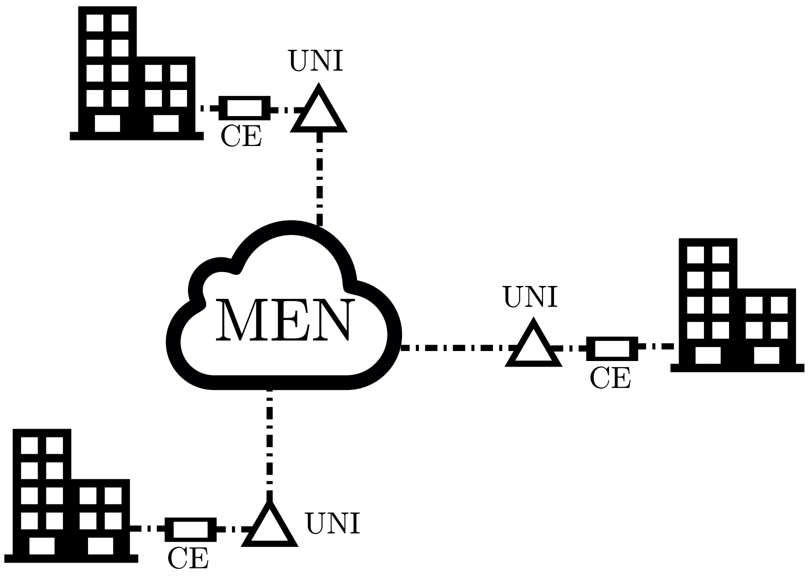


Figure 5: Simple metro ethernet network

3.1.2 Carrier Ethernet

Carrier Ethernet is Ethernet that has been developed from regular Ethernet employed in local area networks, but is aimed specifically for use in a wide area. Carrier Ethernet has a number of modifications in order to be suited for this wide transport application, namely [33]:

- Enhanced equipment redundency.
- Traffic engineering techniques to scale network services.
- Implementation of Ethernet services such as virtual private LAN services that facilitates multipoint Ethernet.

3.1.3 Metro Ethernet Services

An Ethernet service is provided by a Metro Ethernet Network (MEN) provider. In a standard network the Customer equipment (CE) connects to a User-to-network interface (UNI) using a standard Ethernet interface. This is depicted in figure 5.

Ethernet Virtual Connection

Inside a metro network connectivity between UNI is provided by a Ethernet Virtual Connection. The actual connectivity of the virtual connection is provided by a lower layer architecture such as SDH or WDM, however the perspective of the subscriber is that the the network is Ethernet based.

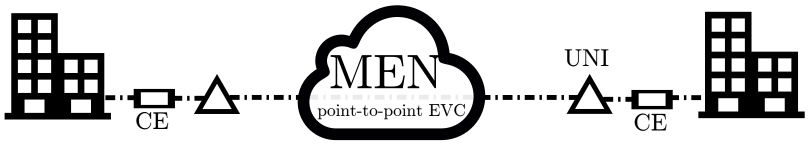


Figure 6: Example of ethernet line service

Ethernet Virtual Connection

A Ethernet Virtual Connection (EVC) is a connection between two or more UNIs. The EVC connects two or more UNIs and allows Ethernet service frames to be transferred between them. Similarly data transfer is disallowed between UNIs that are not part of the EVC. An EVC may be used to construct a private layer 2 line, and this may be point-to-point or multipoint-to-multipoint.

Ethernet Line Service

The Ethernet Line Service provides a point-to-point EVC between two UNIs. This is depicted in figure 6. The service may be multiplexed and more than one line service may be offered at one of the UNIs. Service frames may also be relayed, allowing two UNI to be connected through another UNI. This allows the creation of more complex topologies such as ring networks.

Ethernet LAN Service

In contrast to the Ethernet Line Servce, The Ethernet LAN Service provides multipoint connectivity for UNIs. Data sent from a UNI can be received by the other connected UNIs. Each UNI is connected to a multipoint EVC. Thus the MEN is viewed as a conventional LAN from the point of view of the subscriber. Similar to the Ethernet Line Service, the Ethernet LAN Service may provide multiplexing at the ports of some of the connected UNIs. An example is shown in figure 7. It is important to note that when connecting multiple UNIs, they share the same EVC under the Ethernet LAN Service; when connecting multiple UNIs under the Ethernet Line Service, a separate EVC is required for each UNI. Such as Ethernet Line Service scheme with frame relay used to construct a LAN is shown in figure 8.

3.1.4 Architectures

Ring topologies are the preferred architecture by architectures for implementing MAN networks as they are easier to deploy and manage than meshed networks.

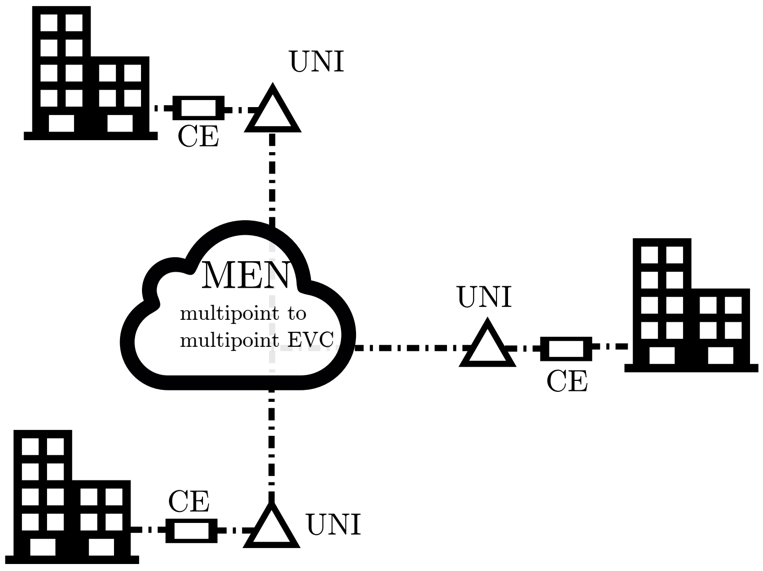


Figure 7: Example of Ethernet LAN Service

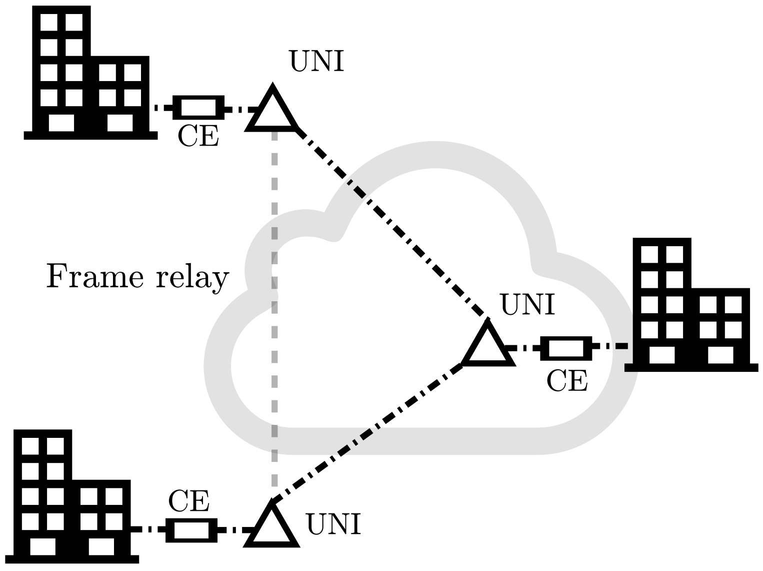


Figure 8: Frame relay on Ethernet Line Service

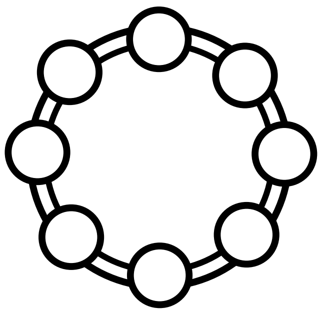


Figure 9: Example of SHR

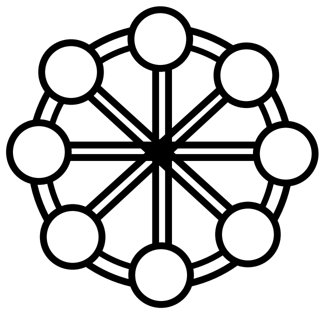


Figure 10: Example of a fully meshed network

Self-healing ring

A Self-healing Ring (SHR) is a circular network topology. Using a loop structure provides redundency. In a circular network structure, when an edge fails, there is still a backup path in the other direction. The system contains bidirectional links between any two nodes. Under normal operating conditions, network traffic is sent from the source along the shortest path toward the destination. In the event of a node loss, or when a link gets severed, the traffic can be routed through the other direction in the loop. This provides survivability to the network. Figure 9 shows an example of such a ring.

Meshed network

In a mesh network topology, each node relays data to other nodes in the network. Routing is employed in order to direct packets to the correct destination.

A fully meshed network is a topology where each node is linked to every other node. When a node or link is broken, a routing algorithm ensures that the message is propagated along another path. The number of links, increases rapidly as the number of nodes increase, hence a similar rapid increase in cost follows. Figure 10 shows an example of such a network. A fully meshed network provides a high degree of survivability and robustness.

3.1.5 WDM

WDM provides multiplexing for fiber-optic networks. WDM uses multiplexing to join several signals together, and a demultiplexing to recover individual signals; see figure 11. The optical signals are multiplexed onto a optical fiber using different wavelengths. This allows network providers to easily add bidirectional communication on a single fiber, as well as upgrading the capacity of existing fiber installations.

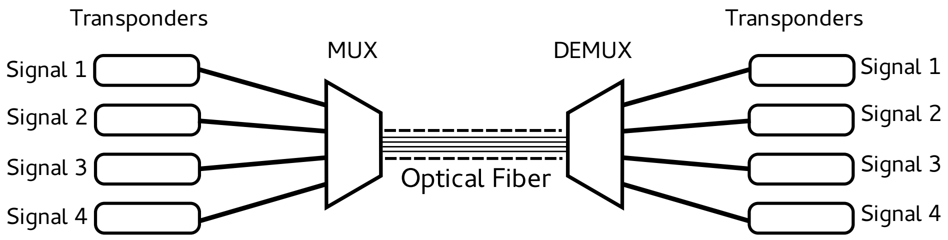


Figure 11: Example of WDM system

Different types of WDM systems exist such as Bi-directional Wavelength Division Multiplexing (BWDM), Coarse Wavelength Division Multiplexing (CWDM) and Dense Wavelength Division Multiplexing (DWDM). BWDM is commonly referred to as just WDM and uses two wavelengths on a single fiber. CWDM provides up to 16 channels. DWDM uses the least amount of spacing between wavelengths and provides the most number of channels, usually 40 or 80. New technologies such as Ultra Dense are being developed that may allow up to 12.5GHz spacing between channels.

WDM offers a low cost per bit transmission capability. When using Ethernet over WDM an arbitrary logical Ethernet topology is used that is based on WDM lightpaths, which is independent of the underlying physical topology.

Many service providers are moving away from SONET/SDH networks towards DWDM networks, as this reduces fiber requirements. WDM allows multiple signals to be transmitted over a single fibre by encompassing each on a different wavelength.

Siemans has estimated that the capital savings of Ethernet over DWDM to be approximately 40% [34], and up to 70% capital saving when Carrier Ethernet replaces legacy ATM access networks. Siemans identifies 5 reasons why network providers should consider Ethernet switching on top of DWDM [34]:

1. Reducing network layers, which also reduces equipment costs.
2. Improve better bandwidth efficiency.
3. Simplify End-to-End provisioning.
4. Better network management and reduction in operating expenses.
5. Better detection of network problems.

Two configurations are commonly used:

1. Opaque - Lightpaths terminate at each node and there is no transparent bypass, hence the logical topology mimicks the physical topology.
2. Meshed - Transparent bypass of lightpaths through optical nodes. This is accomplished using reconfigurable optical add drop multiplexers or optical crossconnects.

WDM devices provide multiplexing and demultiplexing capability and operate at the nodes of the network. Some of these devices have simple functionality such as retransmitting or regenerating the signal, others allow wavelength channels to be added or dropped.

1. Optical Add Drop Multiplexer (OADM) - This device is used in WDM systems for multiplexing and routing channels of light into or out of a single mode fiber. A reconfigurable OADM consists of remotely configurable optical switches in the middle stage. An OADM can be viewed as a Optical Cross-Connect (OXC) with a to-node-degree of two. The device has the capability to add (or drop) wavelength channels to an existing WDM signal. In this work these devices are sometimes referred to as optical nodes

2. Optical Cross-Connect - A device to switch high speed opticals signals in fiber network. Different types exist, namely:
   - Opaque OXC - Optical input signals are converted into electric signals. Optical signals are converted to electronic signals, these electronic signals are switched by an electronic switch module which are lastly converted back into optical signals. These devices are not transparent to the network protocols used, however they have the advantage of regenerating the optical signal.
   - Photonic cross connect - Transparent OXC - Demultiplexes optical signals; these wavelengths are switched by optical switch modules where afterwards they are multiplexed on the output fibers by optical multiplexers.
   - Translucent OXCs - Combination of Opaque OXC and Transparent OXC. Is capable of regenerating the signal when needed and provides optical signal transparency otherwise.

3.1.6 Multiprotocol Label Switching

Multiprotocol Label Switching (MPLS) is a data carrying technique that directs data from a one node to another based on short path labels. These labels identify virtual links between nodes. MPLS is capable of encapsulating various different network protocols and supports a range of access technologies.

A Label switch router is located in the middle of the network and is responsible for switching the labels and routing the packets. The router uses the label and looks up the Label switched path (LSP) from a lookup and swaps the original label with the new corresponding label indicating the next hop.

A Label edge router operates at the edges of the MPLS network. The router either pushes a label if it acts as an entry point (ingress) or pops the label if it acts as an exit point (egress). An egress router needs to contain routing information based on the packets' payload (since there are no labels left to lookup).

3.2 LITERATURE

In this section we briefly review work in the network planning literature, as well as related work on multilayer network planning.

3.2.1 Network Planning

Mixed Integer Linear Programming (MILP) can be used in network planning to solve a variety of problems, including minimization of energy consumption [35], survivability [36, 37], minimization of capital expenditure [9, 38], traffic engineering [39], dimensioning [40] and so on. Other methods such as heuristics exist to solve these problems, however an exact framework such as MILP provides the optimal solution for the model, and for large instances not solved in time, may provide the percentage gap from optimality.

Most commonly networks are modelled as a multicommodity flow problem, Gendron *et al.* [41] provides further information and possible formulations of capicated multicommodity flow problems in network design.

In this work we are mostly concerned with determining the optimal topology for which the capital expenditure cost is minimal. The network is then realized using multicommodity flow optimization [41].

Three main steps are commonly involved in the network planning process:

- Topological design - Determinining the topology of the network and what components and network devices to utilize.

- Network synthesis - Determining the specifications of the components used and the performance criteria, as well as transmission costs and routing details.

- Network realization - Determining the capacity requirements and reliability of the network.

In this work we are mostly concerned with the topological design of the network and the network realization, though some parts of the network synthesis may feature as well. We incorporate ideas from Graph Theory and Discrete Optimization when planning the network.

3.2.2 Mutlilayer Networks

Network planning research has traditionally focussed on single layer planning even though networks were practically composed of multiple layers. Only recently has computational power increased enough in order to solve multiple layered network models.

Orlowski and Wessäly [9] proposed a model that integrates hardware, capacity, routing, and grooming decisions. The focus is on multiplexing and grooming. The goal is to minimize the capital expenditure costs. No computational results are provided. Idzikowski *et al.* [42] proposed an IP over WDM model. The objective is to minimize power consumption. The model is based on a single layer arc-based formulation with network equipment constraints and is based on previous work by Orlowski [10].

Engel and Autenrieth [43] proposed a multilayer network for minimizing cost for a network provided by Swisscom. They found that the cost of the IP layer topology is dependend on the ratio of equipment costs and recommend keeping the topology as a design parameter. The actual model, however, is not provided.

Kubilanskas *et al.* [44] develop three formulations for two-layer networks that carry elastic traffic. Network flow can be reconfigured on the case of link failures.

Rizzelli *et al.* [35] present an IP over WDM MILP model with a arc-based formulation for minimizing power consumption. They account for equipment and include rack/shelf model of the IP layer. Wavelength assignment is also included. The study finds that the IP layer accounts for the majority of the power usage. The authors of the study do not explicitely formulate the model as multilayer.

Baier *et al.* [45] evaluate a metro ethernet ring network. An arc-based formulation is used in order to minimize the cost of installed network cards. The network is protected against single link failures by incorporating 1+1 protection in the model. It was found that for transparent networks the connection-oriented Transport Ethernet is more cost-efficient. When accounting for equal survivability requirements Transport Ethernet was found to be more cost-efficient in both cases.

With the exception of Orlowski [9, 10] most of the literature does not explicitely model multiple layers. Some work such as [10] explicitely define possible physical paths for each logical link. This has the advantage of reducing computational requirements, but results in a higher objective value when minimizing cost. In this work possible physical paths are generated implicitely for logical links, and any physical path may be used that shares the same node endpoints as the logical link.

3.2.3 Survivability

Network survivability, is the ability of the network to remain operational in the event that one or more network components fail, and is critical in present day networks.

The two main models of survivability used in the MILP network literature are dedicated 1+1 protection and diversification. Protection is often given against single link failures, as the introduction of additional survivability requirements increases computational effort. This is the motivation behind work such as [46] which uses decomposition techniques and primal heuristics in order to improve the scalability of models that include survivability.

The dedicated 1+1 protection used in this work, sends double the amount of flow d required (i.e. $2d$ is sent), and ensures that the maximum flow over a link not exceed d. This ensures that should a link fail, a backup path exists over which the required flow may traverse. This approach is used in [36].

Alternatively diversification may be used, this limits the maximum flow over a link to a certain percentage λ. In the event of a link failure, the demand may not be fully satisfied, however some part of it would remain intact. This approach is used in Terblanche *et al.* [38] and is used in some of the models in this work.

3.2.4 Decomposition and Heuristics

The general problem of integer programming is as hard as the hardest problems in NP [7, 47]. Thus much work is dedicated to improving scalability.

MILP models may have a large number of constraints and variables. A greater burden is placed on the solver when there are more constraints and variables. Cutting plane methods focus on reducing the number of constraints and column generation focus on reducing the number of variables.

Lagrangian relaxion[1] [48] penalizes violations of inequality constraints using a Lagrange multiplier. These added costs are then used in place of the inequality constraints in the original problem. Benders decomposition [49] is a technique for solving large problems which have a block structure, such as many MILP problems. Benders decomposition is a type of row-generation technique since it adds more constraints to the problem as it progresses toward a solution. As such it may reduce the number of constraints of the master problem. The general technique is explained in Chapter 2. Column generation [50, 51] is used to solve problems where there are a large number of variables and the technique tries to determine which variables should be in the basis.

1 A relaxation approximates the problem to a easier version

Fortz and Poss [52] present a general multilayer network model that is based on an path flow formulation. They employ Benders decomposition as well as mixed integer rounding cuts in order to speed up the algorithm. A similar approach is used in this work.

The authors of [46] present a general single-layer multicommodity flow problem that incorporates survivability. They study the case of a single node or edge failure in which the flow will be rerouted. Benders decomposition and column generation is used in order to improve scalability. In addition, a primal heuristic is proposed which derives a feasible integer solution from a non-feasible one.

4 BASIC MATHEMATICAL MODEL

4.1 INTRODUCTION

Optimally designing several layers in an integrated step has not been possible until recently due to a lack of suitable mathematical models, algorithms, and computing power.

A common approach in practice has thus been to decompose the multilayer planning problem into a series of singlelayer planning problems: first, the topology, capacities, and routing are planned in the topmost layer; the capacities of this layer then have to be routed as demands in the next lower layer, and capacities have to be determined for this routing, and so on.

As stated previously, planning a network layer by layer results in several inefficencies. When layer by layer network planning is used to build a transport network with IP, the IP services need to be forwarded at intermediate routers. This results in demand for increased capacity of the core routers. The increase in capacity of IP layer equipment leads to demand for large capacity of the optical transport equipment at the bottom layer. This results in a larger than necessary capital expenditure cost in network construction for operators. As mentioned previously, further difficulties are encountered when trying to keep the physical routing paths disjoint when planning the logical layer.

In this chapter we develop a basic, general mathematical formulation for Multilayer networks. The multilayer model is based on the multicommodity flow problem, extended to multiple layers. On each layer equipment is encapsulated by modules, which provide capacity to the network at a certain cost. The goal of these models, is to minimize the capital expenditure costs; the total network equipment cost. The problem is formulated as an arc-based formulation and a path-based formulation. The advantage of the path based formulation is that we can decompose the problem using Bender's decomposition and column generation in order to improve scalability on larger problem instances.

4.2 SINGLE-COMMMODITY FLOW PROBLEMS

A flow network is a directed graph where each edge has a capacity and a flow may traverse over each edge. The amount of flow over an edge may not exceed the capacity. The amount of flow into a

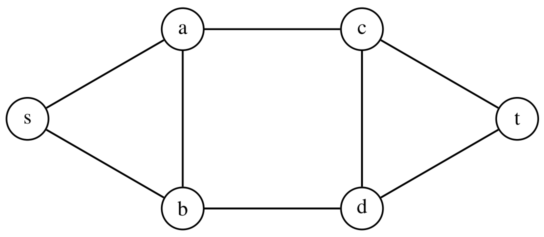


Figure 12: A simple network graph with a source and target

node must equal the amount of flow out of the node, that is, the amount of flow must be conserved. The flow network can be used to model many use cases, such as traffic systems, eletric circuits, fluids, circulation and network traffic.

In the case of the singlecommodity flow problem, a source may only have net positive outgoing flow, and a sink (or target) may only have net positive incoming flow. The source and target pair is commonly known as a commodity. In the singlecommodity maximum flow problem the goal is to obtain the maximum amount of flow between the source and target.

Given a graph $\mathcal{G} = (N, A)$, we want to maximize the total flow, given that the flow f_{ij} may not exceed the capacity u_{ij} over arc ij. The goal is to,

$$\max \quad F, \tag{4.1}$$

subject to conserving the flow (4.2) and restricting the flow to be less than the capacity (4.3), that is,

$$\sum_{j \in N} f_{ij} - \sum_{j \in N} f_{ji} = \begin{cases} F & i = s \\ -F & i = t\,, \\ 0 & \text{else} \end{cases} \qquad \forall i \in N \tag{4.2}$$

$$0 \leqslant f_{ij} \leqslant u_{ij}, \qquad \forall ij \in A, \tag{4.3}$$

where A is the set of all arcs.

The problem, formulated as a LP, can be solved by the simplex method. However, a more efficient algoritm such as the Ford-Fulkerson algorithm is commonly used [53].

4.3 MULTICOMMODITY FLOW PROBLEM

Many problems commonly have more than a single commodity, in which case the problems are referred to as multicommodity flow problems. In most of the network problems discussed in this work, the cost will be minimized, as is the case of this example as well. For the multicommodity flow problem, the Ford-Fulkerson algorithm cannot be applied anymore, and the problem is commonly solved using LP.

Given a graph $\mathcal{G} = (N, E)$, we want to minimize the total cost, given that each edge $e \in E$ has a cost c_e associated with it.

A commodity $k \in K$ is a pair (a, b), $a \in N, b \in N$ with a specific demand d_k that needs to be satisfied, with K denoting the set of all commodities.

In the single-commodity case, flow was modelled over edges. An alternative approach is to model the flow over paths. The former is commonly called a arc-based based formulation and the latter is known as a path-based based formulation[1].

A graph theoretic path is written as a finite sequence of the form (recall section 2.1)

$$v_{i0}, e_{j1}, v_{i1}, e_{j2}, \ldots, e_{jk}, v_{ik}$$

where each edge and vertex is traversed at most once (except possibly the start and end vertices). For notational convenience we commonly only write the edges, e.g. $\{e_1, e_2, \ldots, e_n\}$, as it is usually clear in which direction the path is going and we only consider simple graphs without parallel edges.

SMC (Singlelayer Multicommodity):

The objective is to

$$\min \quad \sum_{a \in E} c_a u_a, \tag{4.4}$$

s.t.

$$\sum_{p \in P_k} f_p = d_k, \qquad \forall k \in K, \tag{4.5}$$

$$\sum_{p \in P_a} f_p \leqslant u_a, \qquad \forall a \in E. \tag{4.6}$$

The set of paths P contains all paths of the graph. The set P_k contains all paths for commodity k, and the set P_a contains all paths that go over over edge a. Constraint set (4.5) requires the sum of flow for each commodity to match the required demand. Constraint set (4.6) indicates that the capacity on edge e must be greater (or equal) to the sum of flows over that edge. This is similar to the single commodity case, however each commodity pair $k = (s, t)$ contains a demand d_k that is required to flow from s to t. Intuitively, we try to minimize the required capacity over each edge, while still trying to

1 This is sometimes abbreviated as just a path or arc formulation

match the demand for each commodity. For a commodity $k = (s, e)$ the paths can be generated using DFS or BFS, see section 5.1.1 for an example algorithm. Generating all paths is computationally difficult for most problems. Column generation will be applied later in order to generate paths only when necessary.

4.4 CAPACITATED NETWORK DESIGN

The two models of sections 4.3 and 4.2 are capacitated network problems. The realized flow over an edge may not exceed the capacity. The network structure is hence determined by the installed capacities. Similarly nodes may also have capacities which may not be exceeded. The network is said to support the demand if there exist a set of flows which do not exceed the respective capacities. In capacitated network design we try to find the optimal capacities that support the demand, for a given objective function [54]. The capacities are commonly integer. Thus the structure of the Capacitied Network Design problem is commonly:

CND (Capacitated Network Design):
The goal is to

$$\min \quad \sum_{a\in E} c_a u_a, \tag{4.7}$$

s.t.

$$\sum_{p\in P_k} f_p = d_k, \qquad \forall k \in K, \tag{4.8}$$

$$\sum_{p\in P_a} f_p \leqslant u_a, \qquad \forall a \in E, \tag{4.9}$$

with $u_a \in \mathbb{N}$

When the integrality constraint on the capacity is removed, the problem reduces to $|K|$ shortest path problems. The LP is then solved by sending each demand $k = (s, t)$ from s to t over the shortest path on a graph with with weights c_a

With the integrality constraint the structure is much more difficult computationally and this depends on the actual data of the problem. From Chopra [55] we know that **CND** is NP-hard. Furthermore, the lower bound of the LP relaxation[2] is weak, which will cause the branch and bound tree to be large.

By applying Farkas' lemma (Eq (2.3.1)) to the LP relaxation of **CND** we obtain the Japanese theorem

2 Removing the constraint that requires the variables to be integer

Theorem 4.4.1 (Japanese theorem). *The capacity vector* u *supports the demand iff*

$$\sum_{a\ inE} \mu_a u_a \leqslant \sum_{k \in K} l_\mu(k) d_k$$

Where $l_\mu(k)$ *is the shortest path of between the end points of commodity* k *with respect to weights* μ

The Japanese theorem is used when applying Bender's decomposition to the path-based formulation of multi commidity flow problems.

4.4.1 The general flow problem

Given a directed graph $G = (V, A)$ with capacities u_a for each arc $a \in A$. Given two nodes $s,\ t \in V$ we wish to find a flow that fulfills the following constraint:

$$\sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0, \qquad \forall\, v \in V \backslash \{s, t\}$$

where $\delta^+(v) = \{(v, w) \in A\}$ and $\delta^-(v) = \{(w, v) \in A\}$ which represent in the in-arc and out-arcs respectively, and the vector x is called a flow.

The flow is feasible if:

$$x_a \leqslant u_a \qquad \forall\, a \in A$$

One can alternatively formulate the constraints using paths. Let P_{st} be the set of all paths from s to t and let P_a be a subset for which each path $p \in P_a$ has $a \in p$.

Thus a feasible path flow is one that:

$$\sum_{p \in P_a} f_p \leqslant u_a, \qquad \forall\, a \in A$$

Figure 13 shows a simple solution of a capacitated singlecommodity flow problem. The edge labels denote the cost. For a given demand, either the mid path or the bottom path (shown in red), for commodity (A,B) would be used[3], as they have the lowest cost.

4.4.2 Example formulations

Aside from the previously covered multicommodity and singlecommodity formulations, we present below a few other capacitated formulations, which are presented below.

3 Or possibly both, with the flow being divided

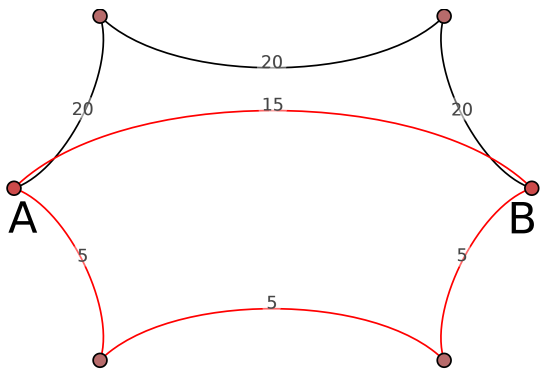


Figure 13: Example solution of capacitated singlecommodity flow problem

Maximum flow arc-based formulation

Using arcs, the multicommodity problem is formulated as,

$$\max \quad \sum_{i=1}^{k} \left(\sum_{a \in \delta^{+}(s_i)} x_a^i - \sum_{a \in \delta^{-}(s_i)} x_a^i \right)$$

s.t.

$$\sum_{a \in \delta^{+}(v)} x_a^i - \sum_{a \in \delta^{-}(v)} x_a^i = 0, \qquad \forall\, v \in V \backslash \{s, t\},\ i \in K,$$

$$\sum_{i=1}^{k} x_a^i \leqslant u_a, \qquad \forall\, a \in A,$$

$$x_a^i \geqslant 0, \qquad \forall\, a \in A, i \in K.$$

Maximum flow path-based formulation

Using paths, the previously mentioned singlecommodity problem (section 4.2) is formulated as

$$\max \quad \sum_{p \in P} f_p,$$

s.t.

$$\max \quad \sum_{p \in P_a} f_p \leqslant u_a, \qquad \forall\, a \in A,$$

$$0 \leqslant f_p, \qquad \forall\, p \in P.$$

Minimum cost multicommodity flow

Commonly we want to minimize the capacity needed. For each flow there is a cost c_a for each $a \in A$ involved. The demand for flow needs to be satisfied for demand values $d_k \geqslant 0$ for $k \in K$. Hence the goal is to

$$\min \quad \sum_{p \in P} \sum_{a \in p} c_a f_p,$$

s.t.

$$\sum_{p \in P_k} f_p = d_k, \qquad \forall\, k \in K$$

$$\sum_{p \in P_a} f_p \leqslant u_a, \qquad \forall\, a \in A,$$

$$0 \leqslant f_p, \qquad \forall\, p \in P.$$

A minimum capacity cost formulation

Capacity is commonly an expensive resource and has a cost associated with it. In place of flow having a cost, the installed capacity has a cost, and we wish to minimize the installed capacity.

For each arc $a \in A$ we have a cost c_a.

The goal is to

$$\min \quad \sum_{a \in A} c_a u_a$$

where u_a is the capacity of edge a,

s.t.

$$\sum_{p \in P} f_p = d_k, \qquad \forall k \in K,$$

$$\sum_{p \in P_a} f_p - u_a \leqslant 0, \qquad \forall a \in A,$$

where $P_a = \{p \in P : a \in p\}$, with bounds

$$u_a \in \mathbb{Z}^+, \qquad \forall\, a \in A,$$

$$f_p \geqslant 0, \qquad \forall\, p \in P.$$

The problem is a MILP as the capacities are positive integers

Figure 14 shows an example simple singlecommodity network, which will be used to demonstrate the formulation. Note that a singlecommodity network is a multi commodity network with $|K| = 1$.

For the example, let us assume the following:

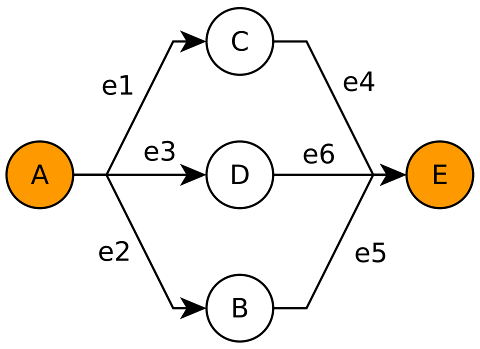


Figure 14: Example singlecommodity network

- There is a single commodity consisting of nodes A and B with a demand of d_0
- For each edge i there exists a cost c_i per unit capacity.

For each edge, there exists a capacity to be solved. The capacities will be given by $u_1, u_2, u_3, u_4, u_5, u_6$ for edges $e_1, e_2, e_3, e_4, e_5, e_6$.

From the example we can see that there are three paths from A to B. The first path consists of $p_1 = \{A, e_1, C, e_4, E\}$, the second path consists of $p_2 = \{A, e_3, D, e_6, E\}$, and the last path consists of $p_3 = \{A, e_2, B, e_5, E\}$. There exists a flow variable f_i for each path i, which for the example will be f_1, f_2, f_3 for the respective paths.

The decision variables are the flow f_i and the capacities u_a. These are the variables to be solved. The inputs are the costs of each edge c_a and the demand for the commodity.

Thus, the example can be formulated as:

$$\min \quad \sum_{i=1}^{6} c_i u_i$$

s.t.

$$f_1 + f_2 + f_3 = d_0$$

$$f_1 \leqslant u_1$$

$$f_1 \leqslant u_4$$

$$f_2 \leqslant u_3$$

$$f_2 \leqslant u_6$$

$$f_3 \leqslant u_2$$

$$f_3 \leqslant u_5$$

with u_i being positive integers and the flows being real numbers greater or equal to zero.

Here the flow over a path may not exceed the capacity of any constituent edge.

When we have values for the costs and demands, the problem can be given to a MILP solver, which will utilize the branch and bound method together with the simplex method, in order to calculate the flows and capacities.

This is an extremely simple example; even this basic formulation can be slow to solve for larger problem instances[4].

For more commodities and larger graphs it becomes extremely difficult to enumerate all of the possible paths, and hence an arc-formulation is preferable. Alternatively the problem can be decomposed and column generation can be used, to only calculate the paths as necessary. This will be explained in later sections.

4.5 TRAVELLING SALESMAN PROBLEM

The Travelling Salesman Problem (TSP) can be phrased as follows: given a list of cities, and the distance between each pair of cities, what is the shortest route that visits each city exactly once, and returns to the starting city?

The solution to the problem is the Hamiltonian cycle with the least weight. The problem can be formuled as a integer linear program:

$$\min \sum_{i=0}^{N} \sum_{j=1, j\neq i}^{N} c_{ij} x_{ij}$$

s.t.

$$\sum_{i=1, i\neq j}^{N} x_{ij} = 1, \qquad \forall j \in S$$

$$\sum_{j=1, j\neq i}^{N} x_{ij} = 1, \qquad \forall i \in S$$

$$u_i - u_j + N x_{ij} \leqslant n - 1, \qquad 2 \leqslant i \neq j \leqslant N$$

$$u_i \in \mathbb{Z}$$

Where $S = \{1, 2, \ldots, N\}$ and $x_{ij} \in \{0, 1\}$ equals 1 when there is a path from i to j and 0 otherwise.

The problem has signficance as it is related to survivability in later sections, although not explicitely used. The physical layer of Metro networks is commonly a ring structure, and each commodity pair needs to be connected.

4 A general problem for NP problems as they face the combinatorial explosion

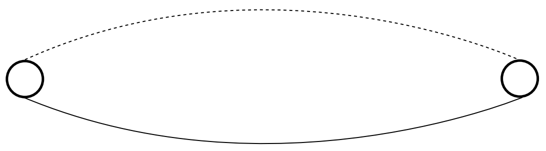


Figure 15: A cycle, with the top edge being broken

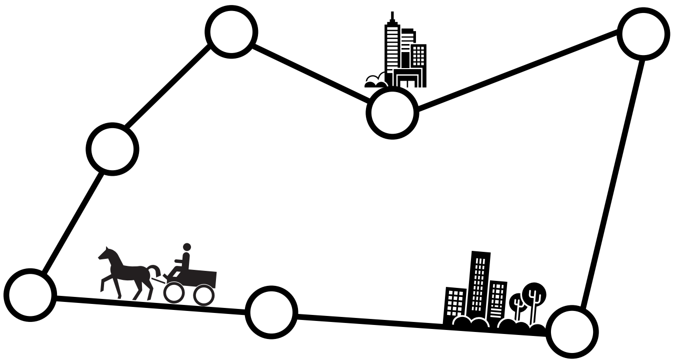


Figure 16: A salesman wanting to travel to of his destination cities and back

Survivability through 1+1 can be achieved by constructing a cycle through all of the required nodes. The required nodes are the commodities on the logical layer, and the logical layer links on the physical layer. This, however, is a heuristic approach, as there may possibilities to share cycles between protected nodes. When a Hamiltonian cycle is constructed on protected nodes, there will exist a backup path if a single edge fails. This is shown in figure 15. For a cycle closing on node v_0: $v_0, e_0, \dots, e_{j-1}, v_i, e_{j+1}, \dots, v_0$ the graph can be partitioned into $v_0, e_0, \dots e_{j-1}, v_i$ and $v_i e_{j+1}, \dots, v_0$, with each being a path. Hence v_0 has two paths to v_i. Since a cycle is a closed circuit, each edge is traversed at most once. Hence the two partitioned paths are edge-disjoint and provide protection for a single edge failure.

Figure 16 displays a solution to a TSP problem. In this simple example a greedy approach connecting the nearest cities would suffice. However for larger problem instances such an approach yields suboptimal

4.6 BASIC MULTILAYER FORMULATION

The basic multicommodity flow model is used to develop a multilayer multicommodity flow model. This model is based on the capacitated multi commodity flow problem, extended to cover two layers. The flow on each layer is constrained by the capacities that the edges can provide. The layers are then interconnected, by requiring that the flows match for each layer. Each layer is modeled after the multicommodity flow problem.

Consider the following two graphs, $\mathcal{P} = (N, E)$ and $\mathcal{L} = (N, L)$, which denote the physical layer and logical layer respectively.

The node set is shared between the two graphs. This is an important point, as it allows us to generate the physical paths for each logical link implicitely. In an explicit approach, the physical paths for each link would have to be predetermined.

The constraint sets (4.8) and (4.9) are added for the logical layer,

$$\sum_{p \in P_k} f_p^L = d_k, \qquad \forall k \in K \tag{4.10}$$

$$\sum_{p \in P_l} f_p^L \leqslant u_a, \qquad \forall l \in L \tag{4.11}$$

Since there are two layers of multicommodity flow problems, each has its own path flow. The flow on the logical layer for a path p is given by the decision variable f_p^L, and similarly f_l^P describes the flow on the physical layer.

In a similar manner to constraint set (4.11), for the physical layer we add:

$$\sum_{p \in P_a} f_p^P \leqslant \sigma_a, \qquad \forall a \in E \tag{4.12}$$

The physical flow over a logical link must be equal to the logical flow over it:

$$\sum_{p \in \delta(l)} f_p^L = \sum_{p \in P_l} f_p^L, \qquad \forall l \in L \tag{4.13}$$

We will change constraint set (4.13) later when decomposing the problem.

This can be seen as two multicommodity flow problems, one on the physical layer and another on the logical layer. On the top layer, the logical flow

As a reminder, in this model, the possible physical paths for each logical link is determined implicitely, and the best possibility is selected. An alternative method, which is used by Orlowski [10] is to explicitely predefine possible physical paths for each link. This places a burden on the network planner, and yields a worse solution, however in some circumstances it may be specified, and required, by the network operator.

4.7 DECOMPOSITION USING BENDERS

In this section we apply Bender's decomposition to the simple multilayer formulation in order to demonstrate the basic process. The process is the same for more involved models.

The problem in section 4.6 will be divided into a master problem and two subproblems using Benders decomposition. Each subproblem is a simpler LP problem.

The logical and physical constraints need to be set independent of each other in order to divide them. This is done by modifying constraint set (4.13) to:

$$\sum_{p \in \delta(l)} f_p^E \geqslant u_l, \qquad \forall l \in L$$

The reason for changing this is twofold. Firstly there might be additional traffic on the physical layer (such as control packets), which results in greater physical flow. The second reason is that we need to loosen the dependence between the physical and logical constraints for the decomposition.

The result is that the physical flow should be greater than or equal to the logical capacity. It must be, as it needs to support the logical traffic. In this model, there are no additional traffic requirements on the physical layer, and equality should suffice.

MBEND (Multilayer Benders decomposition)

This results in the following model, where the goal is to

$$\min \quad \sum_{a \in E} c_a \sigma_a, \tag{4.14}$$

s.t.

$$\sum_{p \in P_k} f_p^L = d_k, \qquad \forall k \in K, \tag{4.15}$$

$$\sum_{p \in P_l} f_p^L - u_a \leqslant 0, \qquad \forall l \in L, \tag{4.16}$$

$$\sum_{p \in P_a} f_p^E - \sigma_a \leqslant 0, \qquad \forall a \in E, \tag{4.17}$$

$$\sum_{p \in \delta(l)} f_p^E - u_l \geqslant 0, \qquad \forall l \in L. \tag{4.18}$$

As stated in section 2.7, decomposing this problem involves separating the problem into a relaxed master problem and into a subproblem(s). The relaxed master problem only contains integral constraints (or none as per the basic multilayer formulation).

RMP (Reduced Master Problem):

$$\min \quad \sum_{a \in E} c_a \sigma_a + y + z, \tag{4.19}$$

with integral capacities,

$$\sigma_a \forall a \in E.$$

With an empty constraint, to which Bender cuts will be added.

The current feasible solution of the **RMP** is used for the subproblems. For the logical subproblem:

LOG (Logical)

$$\min \quad \sum_{p \in P} 0 f_p^L \tag{4.20}$$

s.t.

$$\sum_{p \in P_k} f_p^L = d_k, \qquad \forall k \in K \tag{4.21}$$

$$\sum_{p \in P_l} f_p^L \leqslant u_l^*, \qquad \forall l \in L \tag{4.22}$$

Associate dual variables $\pi_k \in \mathbb{R}, k \in K$ and $\theta_l \in \mathbb{R}^+, l \in L$ to constraint sets (4.21) and (4.22) respectively. Then, the cut we need to add is

$$y \geqslant \sum_{k \in K} \pi_k^* d_k + \sum_{l \in L} \theta_l^* u_l \tag{4.23}$$

The simplification of the linear combination of weights in the objective of **LOG** is simply zero, as such, only a feasibility problem is solved. From hereforth, the goal of subproblems will simply be written as $\min 0$.

For the physical subproblem:

PHYS (Physical)

$$\min \quad 0 \tag{4.24}$$

s.t.

$$\sum_{p \in \delta(l)} f_p^E \geqslant u_l^*, \qquad \forall l \in L \tag{4.25}$$

$$\sum_{p \in P_a} f_p^E \leqslant \sigma_a^*, \qquad \forall a \in E \tag{4.26}$$

Associate dual variables $\eta_l \in \mathbb{R}, l \in L$ and $\mu_a \in \mathbb{R}^+, a \in E$ to constraint sets (4.25) and (4.26) respectively. Then, the cut we need to add is

$$z \geqslant \sum_{l \in L} u_l \eta_l^* + \sum_{a \in E} \sigma_a \mu_a^* \tag{4.27}$$

In order to avoid having to calculate an initial feasible solution, we add an α to each subproblem, which serves as a capacity shortfall. We rearrange some terms and obtain our final decomposition as:
RMP':

$$\min \qquad \sum_{a \in E} c_a \sigma_a \tag{4.28}$$

integers:

$\sigma_a \forall a \in E$

LOG'

$$\min \qquad \alpha \tag{4.29}$$

s.t.

$$\sum_{p \in P_k} f_p^L = d_k, \qquad \forall k \in K \tag{4.30}$$

$$\sum_{p \in P_l} f_p^L - \alpha \leqslant u_l^*, \qquad \forall l \in L \tag{4.31}$$

The cut we need to add is

$$\sum_{l \in L} \theta_l^* u_l \leqslant - \sum_{k \in K} \pi_k^* d_k \tag{4.32}$$

PHYS'

$$\min \qquad \alpha \tag{4.33}$$

s.t.

$$\sum_{p \in \delta(l)} f_p^E \geqslant u_l^*, \qquad \forall l \in L \tag{4.34}$$

$$\sum_{p \in P_a} f_p^E - \alpha \leqslant \sigma_a^*, \qquad \forall a \in E \tag{4.35}$$

The cut we need to add is

$$\sum_{a \in E} \mu_a^* \sigma_a + \sum_{l \in L} u_l \eta_l^* \leqslant 0 \tag{4.36}$$

Benders decomposition is applied in a branch and bound framework, where we obtain a feasible integral solution and use this fixed value in the subproblems. The process is repeated if we obtain a feasible master problem. The algorithm is described in (2). Each iteration of the algorithm is applied on a branch and bound node, after we obtain a feasible solutions for the integers variables in the master problem. The implementation may vary across solvers, For example, in CPLEX we utilize the lazy callback, to call the procedure given in (2). The lazy callback in CPLEX called when CPLEX finds a new integer feasible solution and when CPLEX finds that the LP relaxation at the current node is unbounded.

Algorithm 2 Bender's decomposition for Multilayer

```
if $UB - LB > \epsilon$
    Obtain the solved variables of the **RMP'**
    Solve **LOG' if** $\alpha_{LOG'} > 0$
        Add cut (4.32)
    Solve **PHYS' if** $\alpha_{PHYS'} > 0$
        Add cut (4.36)
```

Benders decomposition could also have been used to seperate all the integer variables from the continuous flow variables. This would have resulted in a single subproblem for all the layers together. The advantage of having two subproblems is, that we can solve them separately, in parallel.

4.8 COLUMN GENERATION

The time complexity to find all paths in a Directed Acyclic Graph (DAG) tends to be exponential in the worst case. For both the logical and physical subproblems, the number of columns increase greatly as the problem size increases. Using Benders decomposition above we would not be able to handle large problem instances.

We use column generation to address this problem. We initially restrict the number of flow variables available, and only add additional flow variables if they improve the solution. The flow variables determine which paths are available, hence the initial set of paths are reduced, and this set expands. This drastically reduces the number of flow variables in the basis, and is preferred, as in the original (non-column generation) case many of these flow variables would have been assigned a value of zero regardless.

We restrict the paths to $P' \subseteq P$. For the logical paths we have $P'^L \subseteq P'^L$, likewise, for the physical paths we have $P'^E \subseteq P'^E$

Logical

For the logical subproblem, we then have:

$$\min \quad \alpha \tag{4.37}$$

s.t.

$$\sum_{p \in P'_k} f_p = d_k, \qquad \forall k \in K \tag{4.38}$$

$$\sum_{p \in P'_l} f_p^L - \alpha \leqslant u_l^*, \qquad \forall l \in L \tag{4.39}$$

Taking the dual, we obtain:

$$\max \quad \sum_{k \in K} \pi_k d_k + \sum_{l \in L} u_l^* \theta_l \tag{4.40}$$

s.t.

$$-\sum_{l \in L} \theta_l = 1 \tag{4.41}$$

$$\pi_k + \sum_{l \in L_k} \theta_l \leqslant 0, \qquad \forall k \in K, p \in P'_k \tag{4.42}$$

Where equation (4.42) can be seen as a shortest path problem:

$$\pi_k = \min_{p \in P_k} \left\{ \sum_{l \in L} -\theta_l \right\} \tag{4.43}$$

For every commodity $k = (a, b)$ we calculate the shortest path from a to b on graph $\mathcal{L}$ given edge weights $\theta_l, \forall l \in L$.

The goal is to see if we can find a shorter path length than π_K for commodity k. If this is the case then constraint (6.40) is violated and we need to add the path, and reiterate.

Let SP$(\mathcal{G}, W, a, b)$ be a function returning the shortest path between a and b on graph $\mathcal{G}$ given edge weights W. Let (d, p) denote the result, where d is the distance and p is the path.

For every commodity k we solve the shortest path using Dijkstra's algorithm, we add the resulting path p if $d < \pi_k$ to P'^L.

Physical

Similarly we proceed with the physical paths. Notice that both problems share the same structure.

We obtain the dual as:

$$\max \quad \sum_{l \in L} u_l^* \eta_l + \sum_{a \in E} \sigma_a^* \mu_a \tag{4.44}$$

s.t.

$$-\sum_{a \in E} \mu_a \leqslant 1 \tag{4.45}$$

$$\eta_l + \sum_{a \in E_l} \mu_a \leqslant 0 \qquad \forall l \in L, p \in P_l' \tag{4.46}$$

For every logical link $l = (q, w)$ we solve the shortest path between q and w on the on graph $\mathcal{P}$ with weights μ and add the resulting path to P'^E if $d < \eta_l$. Benders decomposition with column generation is described in Algorithm 3.

As described previously, algorithm (3) would be called at a branch and bound node, when we have obtained feasible integer solutions for the variables in the master problem.

Orlowsky [10] considers an edge flow formulation for Multi Layer networks since many subproblems with a path based formulation are NP-hard. Orlowsky and Pioro [37] have shown that for multiple logical link failures the pricing problem for path variables is NP hard for most path-based survivability mechanisms. We deviate from these suggestions as we only consider single link failures.

4.9 PATH FLOW MODULE BASED FORMULATION

We expand the path-based model presented in the previous section with modules. Commonly we don't only want to determine the least

Algorithm 3 Benders decomposition with column generation

```
1  Solve the **RMP′**
2  Obtain the solved variables of the **RMP′**
3  *logical*: Solve **LOG′** **for** $k = (s, t) \in K$
4      *Add path:*
5      $(p, \mathrm{dist}_k) \leftarrow \mathrm{SP}(\mathcal{L}, \{\theta_l : \forall l \in L\}, s, t)$
6      **if** $\mathrm{dist}_k < \pi_k$
7          $P^L \leftarrow P^L \cup \{p\}$
8          Repeat *Add path*
9  **if** path was added
10     **goto** *logical*
11     Repeat *Add path* **if** $\alpha_{LOG} > 0$
12     Add cut (4.32)
13 *physical*: Solve **PHYS′** **for** $l = (v, w) \in L$
14     *Add path:*
15     $(p, \mathrm{dist}_l) \leftarrow \mathrm{SP}(\mathcal{P}, \{\mu_a : \forall a \in E\}, v, w)$
16     **if** $\mathrm{dist}_l < \eta_l$
17         $P^E \leftarrow P^E \cup \{p\}$
18 **if** path was added
19     **goto** *physical* **if** $\alpha_{PHYS} > 0$
20     Add cut (4.36)
21 Solve **RMP′** again with added cuts
```

capacity of an edge, but to determine what is the optimal network configuration to realize the edge.

The broad spectrum of available equipment is vast for any type of technology. In this section we develop a general and modular Multilayer formulation which can be more easily adapted to accomodate certain technological requirements. A path based formulation is more expressive and certain path requirements cannot be expressed in a arc-formulation.

There is a wide range of hardware that can be installed at the nodes of the network. The hardware commonly has a modular structure which allows additional components to be inserted. A link can be established by adding the same type of interface cards to both routers at the node endpoints.

In a network a large part of the cost is incurred by the transmission equipment. We assume every line card provides a single port, hence the cost of the logical link can be attributed to it. This is similar for physical equipment. The cost of the multiplexing and switching can be aggregrated with the cost dependent on the length of the physical medium. The actual costs and capacities provided are dependent on the underlying technology.

For these reasons we encapsulate the the various costs of equipment, and the capacity they provide as modules. A module on a edge provides a certain capacity at a certain cost (which may be calculated depending on the length of the edge and the equipment involved). We use this generalization for both the physical and logical layers. Later we will specifically apply this general multilayer model to certain technologies.

We denote this model by **MLPILNS** (Multilayer path-based no survivability).

The aim again is to minimize the total capital expenditure:

$$\min \qquad \sum_{a \in E} \sum_{m \in M_a} c_m x_a^m + \sum_{l \in L} \sum_{m \in M_l} c_m y_l^m \tag{4.47}$$

Where c_m is the cost of module m. The decision variables $X_a^m \in \mathbb{N}^0$ and $Y_l^m \in \mathbb{N}^0$ determine how many modules of type m are installed on physical edge a and logical link l respectively determine how many modules.

We proceed with the usual constraints, replacing the occurence of capacity with the total capacities provided by all modules on a link or edge.

s.t.

The total flow for each commodity must be equal to its demand.

$$\sum_{p \in P_k} f_p^L = d_k, \qquad \forall k \in K \tag{4.48}$$

The flow cannot exceed the logical capacity

$$\sum_{p\in P_l} f_p^L \leqslant \sum_{m\in M_l} k_l^m y_l^m, \qquad \forall l \in L, \tag{4.49}$$

where k^m gives the capacity of module m, and the physical path flows are determined by the logical capacity

$$\sum_{p\in \delta(l)} f_p^E \geqslant \sum_{m\in M_l} k_l^m y_l^m, \qquad \forall, l \in L \tag{4.50}$$

with the physical flow constrainted by the capacity

$$\sum_{p\in P_a} f_p^E \leqslant \sum_{m\in M_a} k_a^m x_a^m, \qquad \forall a \in E. \tag{4.51}$$

The number of installable modules on each layer is integral,

$$X_a^m \in \mathbb{Z}^+, \forall a \in E, m \in M$$

$$Y_l^m \in \mathbb{Z}^+, \forall l \in L, m \in M$$

In the above model we assume that any integral number of modules may be installed.

We may alternatively explicitely require that only a single module may be installed on link or edge. For this we may add the additional constraint of:

$$\sum_{m\in M_l} Y_l^m \leqslant 1, \qquad \forall l \in L \tag{4.52}$$

where Y_l^m is binary, i.e. $Y_l^m \in \{0, 1\}, \forall l \in L, m \in M$

The same can be done for physical capacities.

$$\sum_{m\in M_a} X_a^m \leqslant 1, \qquad \forall a \in E \tag{4.53}$$

where X_a^m is binary, i.e. $X_a^m \in \{0, 1\}, \forall a \in E, m \in M$

In order to avoid unnecessary redundency, we develop the Bender's decomposition and column generation framework for the case when diversification constraints are added, in section 4.11.2.

4.9.1 Example

Consider the network given by figures 17 and 18. The set of nodes is shared between the two. The orange nodes represent commodity nodes. The yellow colored nodes on the physical layer represent end points of logical links. We denote a module with a tuple, $m_i = (c_i, k_i)$, where c_i is the cost of module m_i and k_i is the capacity of module m_i.

Let us assume the following:

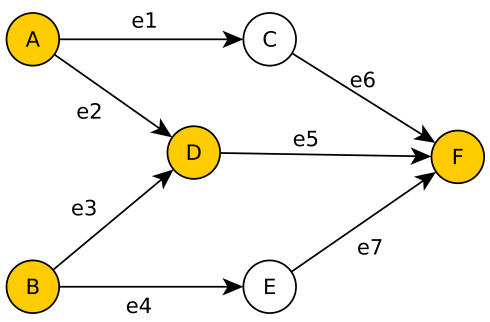


Figure 17: Physical layer of example multicommodity network

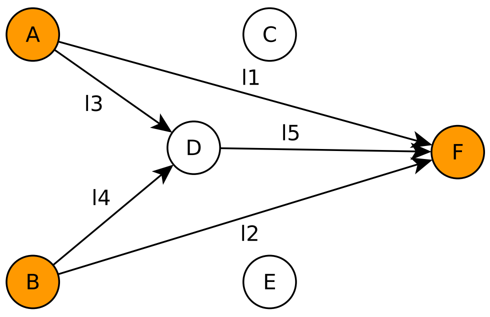


Figure 18: Logical layer of example multicommodity network

- There are two commodities (A, F) and (B, F) with demands of d_0 and d_1 respectively.
- There is a universal set of modules for the physical edges $M^A = m_1, m_2$. Hence $M_a = M^A \forall a \in A$. This assumption is not made for the datasets used in later chapters, where each edge may have its own unique modules, where the cost may be dependent on the edge length and so on. This is convenient for the example, however.
- Similarly, there is a universal set of modules for the logical links $M^L = m_2, m_3$. Again $M_l = M^L \forall l \in L$.

The decision variables are x_a^m and y_l^m indicating whether module m is (or how many) installed on the edge or link respectively.

In the multilayer setting we need to determine paths for all of the possible links. We label them as follows:

- $p_1^E = \{A, e_1, C, e_6, F\}$
- $p_2^E = \{A, e_2, D, e_5, F\}$
- $p_3^E = \{B, e_3, D, e_5, F\}$
- $p_4^E = \{B, e_4, E, e_7, F\}$

- $p_5^E = \{A, e_2, D\}$
- $p_6^E = \{B, e_3, D\}$
- $p_7^E = \{D, e_5, F\}$

The set of paths on the physical layer is thus $P^E = \{p_1^E, p_2^E, p_3^E, p_4^E, p_5^E, p_6^E\}$

For the logical paths we have $P^L = \{\{A, l_1, F\}, \{A, l_3, D, l_5, F\}, \{B, l_4, D, l_5, F\}, \{B, l_2, F\}\}$ Let us denote, in numerical sequence, these paths by $p_1^L, p_2^L, p_3^L, p_4^L$. The respective flows are $f_1^L, f_2^L, f_3^L, f_4^L$.

We also need to determine the set of physical paths available for each logical link. This is given by the function $\delta(l)$. Hence we have:

- $\delta(l_1) = \{p_1^E, p_2^E\}$
- $\delta(l_2) = \{p_3^E, p_4^E\}$
- $\delta(l_3) = \{p_5^E\}$
- $\delta(l_4) = \{p_6^E\}$
- $\delta(l_5) = \{p_7^E\}$

Thus for the formulation we have:

$$\min \quad \sum_{i=1}^{7} \sum_{m \in M^A} c_m x_i^m + \sum_{j=1}^{5} \sum_{m \in M_L} c_m y_j^m$$

For the demands from constraint set (4.48) we have:

$$f_1^L + f_2^L = d_0$$

$$f_3^L + f_4^L = d_1$$

We determine the logical capacities according to constraint set (4.49):

$$f_1^L \leqslant \sum_{m \in M_l} k_l^m y_1^m$$

$$f_4^L \leqslant \sum_{m \in M_l} k_l^m y_2^m$$

$$f_2^L \leqslant \sum_{m \in M_l} k_l^m y_3^m$$

$$f_3^L \leqslant \sum_{m \in M_l} k_l^m y_4^m$$

$$f_2^L + f_3^L \leqslant \sum_{m \in M_l} k_l^m y_5^m$$

From constraint set (4.50) we obtain:

$$\sum_{p \in \delta(l_1)} f_p^E \geqslant \sum_{m \in M_l} k_l^m y_1^m,$$

$$\sum_{p\in\delta(l_2)} f^E_p \geqslant \sum_{m\in M_l} k^m_l y^m_2,$$

$$\sum_{p\in\delta(l_3)} f^E_p \geqslant \sum_{m\in M_l} k^m_l y^m_3,$$

The summations can be expanded by using the $\delta(l)$ paths we derived above.

The physical capacities are determined according to constraint set (4.51):

$$f^E_1 \leqslant \sum_{m\in M_a} k^m_a x^m_1$$

$$f^E_2 + p^E_5 \leqslant \sum_{m\in M_a} k^m_a x^m_2$$

$$f^E_3 + p^E_6 \leqslant \sum_{m\in M_a} k^m_a x^m_3$$

$$f^E_4 \leqslant \sum_{m\in M_a} k^m_a x^m_4$$

$$f^E_2 + f^E_3 + f^E_7 \leqslant \sum_{m\in M_a} k^m_a x^m_5$$

$$f^E_1 \leqslant \sum_{m\in M_a} k^m_a x^m_6$$

$$f^E_4 \leqslant \sum_{m\in M_a} k^m_a x^m_7$$

And the number of modules we install is integer.

In this example we can intuitively reason that the link paths containing D will not be used, since they have more modules in total, and the set of modules are the same between links, which would simply mean minimizing the cost by using the least number of modules. In our datasets this is not the case, D may be a central node with cheap connectivity, or it may be already exist (Implying its module costs might be low).

4.10 ARC-FLOW MODULE BASED FORMULATION

Here we consider an alternative formulation that focusses on the edge flow. This formulation is denoted by **MLAILNS** (Multilayer arc-flow no survivability).

The same objective function is used, that is

$$\min \quad \sum_{a\in E} \sum_{m\in M_a} c_m x^m_a + \sum_{l\in L} \sum_{m\in M_l} c_m y^m_l. \qquad (4.54)$$

We introduce variables $a^k_{(v,w)} \in \mathbb{R}^+$, $b^l_{(v,w) \in \mathbb{R}^+}$ that specify the arc flow over a logical link and physical edge respectively.

The total arc flow over a logical link for a specified commodity is zero; the arc flow in the forward direction over an edge must be the same as the arc flow in the backward direction over the edge. Exceptionally, when the arc flow terminates on sink, we require negative demand; likewise when the arc flow comes from a source, we require positive demand.

$$\sum_{\substack{w \in N \\ (v,w) \in L}} a^k_{(v,w)} - \sum_{\substack{w \in N \\ (w,v) \in L}} a^k_{(w,v)} = \begin{cases} d_k & v = s \\ -d_k & v = t \\ 0 & \text{else} \end{cases}, \qquad \begin{array}{r} \forall v \in N \\ \forall k = (s,t) \in K \end{array} \tag{4.55}$$

The total flow for each commodity cannot be greater than the supplied logical capacities. Logical capacity is the sum of the capacity provided by the link modules.

$$\sum_{k \in K} a^k_{(s,t)} + \sum_{k \in K} a^k_{(t,s)} \leqslant \sum_{m \in M_l} h_m y^m_l, \qquad \forall l = (s,t) \in L \tag{4.56}$$

Similar to constraints (4.55), we require the sum of physical arc flow over a physical edge to be zero, except when we are starting at or terminating a logical link.

$$\sum_{k \in K} \left(\sum_{\substack{w \in N \\ (v,w) \in E}} b^k_l(v,w) - \sum_{\substack{w \in N \\ (w,v) \in E}} b^k_l(w,v) \right) = \begin{cases} \sum_{m \in M_l} h_m y^m_l & v = s \\ -\sum_{m \in M_l} h_m y^m_l & v = t \\ 0 & \text{else} \end{cases}$$

$$\begin{array}{r} \forall v \in N \\ \forall l = (s,t) \in L \end{array} \tag{4.57}$$

We ensure that that the physical edge can provide sufficient capacity for the flow.

$$\sum_{k \in K} \left(\sum_{l \in L} b^k_l(s,t) + \sum_{l \in L} b^k_l(t,s) \right) \leqslant \sum_{m \in M_a} h_m x^m_a, \qquad \forall a = (s,t) \in E \tag{4.58}$$

with integers:

$$x^m_a \in \mathbb{Z}^+, \qquad \forall a \in E, m \in M$$

$$y^m_l \in \mathbb{Z}^+, \qquad \forall l \in L, m \in M$$

The arc formulation is not as intuitive as the path flow formulation, furthermore it suffers furthermore the limitation that it is difficult to restrict allowable paths. With the arc formulation we cannot restrict paths to have a maximum hop count.

4.11 SURVIVABILITY

In this section we review basic survivability. A failure state indicates what set of network equipment fails simultaneously. In our models, this indicates what set of edges or links fail and hence are unavailable. Commonly, for protection, we redefine our demands constraints such that commodities fullfill their demands for all failure states. Thus for the multicommodity problem we have:

$$\sum_{p \in P_k} f_p = d_k, \qquad \forall s \in S, k \in K$$

$$\sum_{p \in P_a} f_p \leqslant x_a, \qquad \forall s \in S, a \in E \cap s$$

In this thesis we only cover single edge failures. Thus the set of failure states is equivalent to the set of edges.

4.11.1 Single layer diversification

In this section we develop a simple method of protection against single link failures.

For any single link failure, we still want to be able to route the full demand of a commodity.

In place of sending only the demand for a commodity, we send double the demand. At the end node, the router then discards the extra data. We restrict the network flow on any path, to not exceed the demand. Thus for a commodity k, with a demand of d_k, we route $2d_k$, but ensure that on any link l, no more than d_k can be routed for that commodity. Thus should link l fail, there is another path that can handle the demand d_k.

We extend model **SMC** with modules, and add the above survivability.

SMPSD:

min

$$\sum_{l \in L} \sum_{m \in M_l} c_m y_l^m$$

s.t.

$$\sum_{p \in P_k} f_p = 2d_k, \tag{4.59}$$

$$\sum_{p \in P_l} f_p \leqslant \sum_{m \in M_l} k_m y_l^m, \qquad \forall l \in L \tag{4.60}$$

$$\sum_{\substack{p \in P_k \\ l \in p}} f_p \leqslant d_k, \qquad \forall k \in K, l \in L \tag{4.61}$$

Where constraints (4.61) ensures that we don't send more than d_k of traffic over any single link, for a commodity. Note that in constraint (4.59), the total flow must equal to twice the demand.

In order for the path flow model to remain computationally tractable, we decompose the problem using Bender's decomposition and generate columns as necessary.

We obtain the restricted Benders master problem as:

$$\sum_{l \in L} \sum_{m \in M_l} c_m y_l^m$$

to which cuts will be added.

For the subproblem, with paths $P' \subseteq P$:

$$\min \qquad \alpha$$

s.t.

$$\sum_{p \in P'_k} f_p = 2d_k, \tag{4.62}$$

$$\sum_{p \in P'_l} f_p - \alpha \leqslant \sum_{m \in M_l} k_m y_l^m, \qquad \forall l \in L \tag{4.63}$$

$$\sum_{\substack{p \in P'_k \\ l \in p}} f_p \leqslant d_k, \qquad \forall k \in K, l \in L \tag{4.64}$$

We obtain the dual of the subproblem:

$$\max \qquad \sum_{k \in K} 2d_k \pi_k + \sum_{l \in L} \sum_{m \in M_l} k_m y_l^m \mu_l + \sum_{l \in L} \sum_{k \in K} d_k \gamma_l^k \tag{4.65}$$

s.t.

$$-\sum_{l \in L} \mu_l = 1 \tag{4.66}$$

$$\pi_k + \sum_{l \in L} \mu_l + \sum_{l \in L} \gamma_l^k \leqslant 0, \qquad \forall k \in K, p \in P'_k \tag{4.67}$$

The cut we need to add is:

$$\sum_{l \in L} \sum_{m \in M_l} k_m y_l^m \mu_l^* \leqslant -\sum_{k \in K} 2 d_k \pi_k^* - \sum_{l \in L} \sum_{k \in K} d_k \gamma_l^{k*} \tag{4.68}$$

Since we are using a subset of paths $P' \subseteq P$, constraints (4.65) must be satisfied; which provides us with the pricing problem.

Thus for a commodity k:

$$\pi_k = \min_{p \in P_k} \left\{ -\sum_{l \in L} \mu_l + \gamma_l^k \right\} \tag{4.69}$$

which states that π_k is the length of the shortest path of graph $\mathcal{G}$ with edge weights $\mu_l + \gamma_l^k$ for edges l.

Note that the constraints are the same as **MLPILNS**, with survivability being added from Section 4.11.1. See the relevant sections for an explanation of the constraints and variables.

Thus in order to find a path that improves the solution value, we need to determine if there exists a shorter path for commodity k that is less than π_k, if there is one, we add it it P' and reiterate.

4.11.2 Multilayer diversification: Path formulation

Survivability for a two-layer network takes a different form than singlelayer survivability.

For a two-layer network, given that a physical link fails, we can either reroute the flow on the physical layer (preserving the logical path), or reroute the flow on the logical layer in order to avoid the failed physical edge.

The model is **MLPILNS** decomposed using Bender's decomposition and column generation in order to add both physical diversification and logical diversification.

The model is split into a reduced master problem and two subproblems, one for the logical flow and another for the physical flow, in the same manner as section 4.7.

The multilayer model with both physical and logical diversification which will be developed is denoted by **MLPILPLS**

For the reduced master problem we have the objective function as:

$$\sum_{a \in E} \sum_{m \in M_a} c_m x_a^m + \sum_{l \in L} \sum_{m \in M_l} c_m y_l^m$$

For the logical subproblem we have:

min α

s.t.

Double the demand is sent. The end node can then has to decide which flows to use (Commodity k will in the end only utilize a single d_k).

$$\sum_{p \in P_k} f_p^L = 2d_k, \qquad \forall k \in K \tag{4.70}$$

$$\sum_{p \in P_l} f_p^L - \alpha \leqslant \sum_{m \in M_l} k_m y_l^m, \qquad \forall l \in L \tag{4.71}$$

The flow is constrained on each link, to be less than or equal to d_k for commodity k. Should a link fail, the sum of all the other paths will be at least d_k

$$\sum_{\substack{p \in P_k \\ l \in p}} f_p^L \leqslant d_k, \qquad \forall k \in K, l \in L \tag{4.72}$$

Associate dual variables π_k, θ_l, γ_l^k with constraints sets (4.70), (4.71) and (4.72) respectively.

The dual constraints are obtained as:

$$-\sum_{l \in L} \theta_l = 1 \tag{4.73}$$

$$\pi_k + \sum_{l \in L} \theta_l + \sum_{l \in L} \gamma_l^k \leqslant 0, \qquad \forall k \in K, p \in P'^L \tag{4.74}$$

And we solve the pricing problem by finding the shortest path:

$$\pi_k = \min_{p \in P^k} \left\{ \sum_{l \in L} -(\theta_l + \gamma_l^k) \right\} \tag{4.75}$$

The Bender's is obtained as:

$$z \geqslant 2 \sum_{k \in K} d_k \pi_k^* + \sum_{l \in L} \sum_{m \in M_l} k_m y_l^m \theta_l^* + \sum_{k \in K} \sum_{l \in L} d_k \gamma_l^{*k} \tag{4.76}$$

Proceeding similarly for the physical subproblem:

$$\min \quad \alpha$$

s.t.

Send double the required physical flow. We need to still be able to match the logical capacity should a physical edge fail.

$$\sum_{p \in l} f_p^E \geqslant 2 \sum_{m \in M_l} k_m y_l^m, \qquad l \in L \tag{4.77}$$

$$\sum_{p \in P_a} f_p^E - \alpha \leqslant \sum_{m \in M_a} k_m x_a^m, \qquad a \in E \tag{4.78}$$

Dont send more than the capacity on a single physical edge:

$$\sum_{\substack{p \in l \\ a \in p}} f_p^E \leqslant \sum_{m \in M_l} k_m y_l^m, \qquad \forall l \in L, a \in E \tag{4.79}$$

Associate dual variables η_l, μ_a and δ_a^l with constraint sets (4.77), (4.78) and (4.79) respectively.

The dual constraints of the problem are:

$$-\sum_{a \in E} \mu_a = 1 \tag{4.80}$$

$$\eta_l + \sum_{a \in E} \mu_a + \sum_{a \in E} \delta_a^l \leqslant 0 \tag{4.81}$$

Similarly we solve the problem of

$$\eta_l = \min_{p \in P_l} \left\{ \sum_{a \in E} -(\mu_a + \delta_a^l) \right\} \tag{4.82}$$

The Bender's cut is obtained as:

$$z \geqslant \sum_{a \in E} \sum_{m \in M_a} k_m x_a^m \mu_k^* + \sum_{l \in L} \sum_{m \in M_l} 2 k_m y_L^m \eta_l^* + \sum_{a \in E} \sum_{l \in L} \sum_{m \in M_l} k_m y_l^m \delta_a^{*l} \tag{4.83}$$

Note that we can remove the logical diversification by simply removing the factor of 2 from (4.70). Similarly we can disable the physical diversification by removing the factor of 2 from (4.77). In the case that that the diversification is disabled, constraints (4.72) and (4.79) serve no feasibility purpose.

4.11.3 Multilayer diversification: Arc model

The model **MLAILNS** is extended to cover both physical- and logical survivability. The resultant model is denoted by **MLAILPLS** (Multilayer arc-flow with physical- and logical survivability).

$$\min \quad \sum_{a\in E}\sum_{m\in M_a} c_m x_a^m + \sum_{l\in L}\sum_{m\in M_l} c_m y_l^m \tag{4.84}$$

s.t.
Logical
In order to protect the network against a single logical link failure we send double the demand.

$$\sum_{\substack{w\in N\\(v,w)\in L}} a^k_{(v,w)} - \sum_{\substack{w\in N\\(w,v)\in L}} a^k_{(w,v)} = \begin{cases} 2d_k & v=s \\ -2d_k & v=t\,, \\ 0 & \text{else}\end{cases} \qquad \begin{array}{r}\forall v\in N\\ \forall k=(s,t)\in K\end{array} \tag{4.85}$$

$$\sum_{k\in K} a^k_{(s,t)} + \sum_{k\in K} a^k_{(t,s)} \leqslant \sum_{m\in M_l} h_m y_l^m, \qquad \forall l=(s,t)\in L \tag{4.86}$$

Ensure the flow on each logical link does not exceed the demand.

$$a^k_{(a,b)} + a^k_{(b,a)} \leqslant d_k, \qquad \forall\, l=(a,b)\in L, k\in K \tag{4.87}$$

Physical
Alternatively, to protect the physical layer against a single edge failure, we ensure that we send double the logical capacity required.

$$\sum_{k\in K}\left(\sum_{\substack{w\in N\\(v,w)\in E}} b^k_l(v,w) - \sum_{\substack{w\in N\\(w,v)\in E}} b^k_l(w,v)\right) = \begin{cases} 2\sum_{m\in M_l} h_m y_l^m & v=s \\ -2\sum_{m\in M_l} h_m y_l^m & v=t \\ 0 & \text{else}\end{cases}$$
$$\forall\, v\in N, \forall\, l=(s,t)\in L \tag{4.88}$$

$$\sum_{k\in K}\left(\sum_{l\in L} b^k_l(s,t) + \sum_{l\in L} b^k_l(t,s)\right) \leqslant \sum_{m\in M_a} h_m x_a^m, \qquad \forall\, a=(s,t)\in E \tag{4.89}$$

Also ensure that the flow on a physical edge does not exceed the demand.

$$b^k_l(a,b) + b^k_l(b,a) \leqslant \sum_{m\in M_l} k_m y_l^m, \qquad \forall (a,b)\in E, l\in L, k\in K \tag{4.90}$$

The decision variables, denoting the number of modules on each layer, are integral:

$$x_a^m \in \mathbb{Z}^+, \qquad \forall\, a \in E, m \in M$$

$$y_l^m \in \mathbb{Z}^+, \qquad \forall\, l \in L, m \in M$$

5 COMPUTATION

In the knapsack problem, a set of items is given, each with a weight and a value. The goal is to determine the number of each item to include, such that the total weight is less than some specified amount, and the total value is as large as possible. The subset-sum is a specialized version of the knapsack problem where the weights and values are the same, and is NP-complete [56], and this extends to the decision version of the knapsack problem[1]. The optimal dynamic programming solution of the knapsack problem admits a pseudo-polynomial solution, which implies that the problem is weakly NP-complete.

Assigning modules to edges or nodes is a special form of the knapsack problem, as each module has a value (cost) and weight (capacity). The goal is to minimize the cost and to determine a capacity sufficiently big for the network flow. The dynamic programming solution for this is pseudo-polynomial.

Furthermore, the decision problem version of the multicommodity flow problem is NP-complete under integer flow [57]. The capacitated multicommodity network design problem is NP-complete even with real flows [58–60].

As such, the model described in Section 4.6 is NP-complete as it contains the capacitated network design problem on both layers, and the knapsack problem for module selection. Thus the problem admits no polynomial solution.

The difficulty of the problem motivates the Benders decomposition approach utilized in Section 4.7 and the column generation shown in Section 4.8, in order to improve the scalability for larger problems.

In this chapter an overview of the algorithmic approach utilized, is given, along with some implementation details. Since larger instances might be computationally intractable, we strenghten the Benders feasibility cuts and implement a simple primal heuristic based on the decompositions used.

Lastly we cover the computational improvements and the results. These results are based on the general models given in Chapter 4.

1 The decision version asks whether a value of V can be attained without exceeding some weight W

5.1 HEURISTICS

In this section a heuristic is introduced which can be used to find a good guess of what the optimal solution should be. The solved solution of the decision variables of the heuristic are then used as a warm-start in order to provide a better upper bound on the problem. Finding a better upper bound saves the solver time, provided a better upper bound is obtained in a smaller amount of time.

5.1.1 Limited paths

In the regular multicommodity path flow formulation, all paths are present for each commodity. In the multilayer case, we also require all physical paths for each logical link. This can be generated with a BFS or DFS backtracking algorithm. However this is not feasible for most problems. In previous sections it was demonstrated how column generation can be used to only generate paths as necessary for the dual LP. This idea can be extended to a simple primal heuristic.

A simple heuristic yielding good results is to reduce the number of paths available. Only having a small static subset of available paths will not yield an objective value close to optimal.

An alternative approach is to decompose the problem using Benders' decomposition and apply column generation. However the the amount of iterations is limited for searching for paths, therefore the number of paths, and path flow variables are limited.

The algorithm is similar to the one described in Algorithm 3. For every iteration the number of path variables generated is limited, as such the procedure *Add path* is run a maximum number of times per iteration. Heuristically, a value of between 1 and 4 yielded good results, as the solution value is close to optimal and is obtained fast. For most of our experiments we only add a single path variable per iteration, as this is fast and obtains a good upper bound for the master problem when used as a warm-start.

The overall idea here is to preserve the same constraints as in the regular MILP problem; that is, the heuristic always provides a feasible solution to the regular MILP. Since only a subset of paths is covered, there may exist some paths which yield a better objective value, and these are skipped in favor of computation time. Colunm generation is thus used heuristically.

An alternative version of this heuristic would be to use the path-based formulation of the problem, and only include path variables relating to the shortest path for the links and commodities. This would be faster than the heuristic discussed above, but would yield a worse objective value.

5.2 STRENGTHENING CUTS

The Bender cuts generated for **SMPSD** are weak. We follow a simple way of strengthening them, based on [52].

We review the single layer multicommodity with survivability problem:

$$\min \quad \sum_{l \in L} \sum_{m \in M_l} c_m y_l^m$$

s.t.

$$\sum_{p \in P_k} f_p = 2d_k, \qquad \forall k \in K$$

$$\sum_{p \in P_l} f_p \leqslant \sum_{m \in M_l} k_m y_l^m, \qquad \forall l \in L$$

$$\sum_{\substack{p \in P_k \\ l \in p}} f_p \leqslant d_k, \qquad \forall k \in K, l \in L$$

The Bender's cut for for the problem is taken from the dual, and is obtained as:

$$\sum_{l \in L} \sum_{m \in M_l} k_m \theta_l^* y_l^m \leqslant -\sum_{k \in K} 2d_k \pi_k^* - \sum_{l \in L} \sum_{k \in K} d_k \gamma_l^* \tag{5.1}$$

The metric inequality 5.1 is weak as the capacities are modular and integral. We strengthen the metric inequality by dividing by the biggest possible integer and rounding up to the nearest integer on the right hand side.

Let $\gcd(a, b)$ denote the greatest common divisor between two numbers, a and b, that is, it returns the largest positive integer that divides a and b without a remainder. The function can be extended to three or more positive integers, and returns the largest divisor shared by all of them.

Since $k_m \theta_l^*$ is integer; we determine the greatest common divisor of $\{k_m \theta_l^* : l \in L, m \in m_L\}$ and denote it with $\gcd(k\theta)$.

We divide both sides of 5.1 by $\gcd(k\theta)$ and round up the right hand side, obtaining:

$$\sum_{l \in L} \sum_{m \in M_l} \frac{k_m \theta_l^*}{\gcd(k\theta)} y_l^m \leqslant \left\lceil -\frac{\sum_{k \in K} 2d_k \pi_k^*}{\gcd(k\theta)} \right\rceil + \left\lceil -\frac{\sum_{l \in L} \sum_{k \in K} d_k \gamma_l^*}{\gcd(k\theta)} \right\rceil \tag{5.2}$$

Strengthening the Bender's cuts on the multi layer model is the same as for the singe layer model. Recall that we have two Bender subproblems, each with its own cut.

5.3 COLUMN GENERATION IMPROVEMENTS

5.3.1 Dijkstra

In order to find variables with negative reduced cost in the pricing problem encountered in most of our column generation decompositions, we need to solve the shortest path problem, which is simply a path that minimizes the sum of the weights of the constituent edges. Dijkstra is commonly used to optimally find the shortest path in weighted graphs and originally had $O(|V|^2)$ running time.

Algorithm 4 Dijkstra's algorithm

```
DIJKSTRA($G = \mathtt{Graph}, s = \mathtt{vertex}$)
 1  for each vertex $v \in V_G$
 2      $dist[v] \leftarrow \infty$
 3      $parent[v] \leftarrow$ NIL
 4  $dist[s] \leftarrow 0$
 5  $Q \leftarrow V_G$
 6  while $Q \neq \emptyset$
 7
 8      $u \leftarrow$ EXTRACT-MINQ
 9      for each edge $e = (u, v)$
10          if $dist[v] > dist[u] + weight[e]$
11              $dist[v] \leftarrow dist[u] + weight[e]$
12              $parent[v] \leftarrow u$ return $\mathtt{parent}, dist$
```

Dijkstra's algorithm is described in Algorithm 4. Dijkstra's algorithm is a greedy algorithm for finding the shortest path, and proceeds as follows:

1. Assign to each node v a distance value, stored in *dist*, this is initialized to infinity (or a large number) for all nodes at the start of the search.
2. Keep a list of nodes that are unvisited. The exclusion from Q determines whether a node is unvisited.
3. Set the starting node u as current.
4. For every neighbor of the current node, calculate its distance from the current node (by adding the weight of the edge to the calculated distance of the current node). Update the stored distance of the node to be the minimum of its current value and the newly calculated value.
5. Mark the current node as visited, by removing it from Q. End when the destination node is visited.

6. Mark the node with the smallest stored distance as current, and repeat from 4.

When computing the shortest path between two points, the search can be halted once u is the target vertex.

In the pseudocode of the algorithm, given in Algorithm 4, the procedure *EXTRACT-MINQ* performs the task of selecting a node with the smallest stored distance (step 6), as well as removing it from the set of unvisited nodes Q (step 5). This structure represents a queue, and in particular a priority-queue is able to obtain the smallest element. When a priority is implemented using a Heap structure, it is able to do so in $O(\log(|V|))$ time.

A functional variant of Dijkstra's single-source shortest path algorithm is used, that is based on Priority Search Queues. In order to improve the original algorithm from $O(|V|^2)$ to $O((|V|+|E|)\log(|V|))$ we need to be able to find the minimal element faster. Heaps allow the minimum element to be found in $O(\log(n))$, which is what we need. The term priority queue is often used to refer to a non-specific datastructure which allows us to determine the minimum element, but does not specify the implementation. We will be using a functional equivalent, priority search queues.

The implementation of priority search queues used is based on work from Hinze [61], which is a combination of finite maps and priority queues. This data structure is used since it features $\Theta(\log(n))$ lookup time for the minimum value and $\Theta(\log(n))$ for decreasing the value of a key (for updating the frontier distances in Dijkstra's algorithm).

Overall, the implementation of Dijkstra has a worst-case running time of $\Theta((|V|+|E|)\log|V|)$ using Priority Search Queues.

Dijkstra's algorithm is mainly used in the pricing problem for identifying additional paths to be included when doing column generation. For the logical subproblem we check whether each commodity $k \in K$ has a path shorter than it's dual value π_k. The task is actually embarrassingly parallel[2]. We calculate the shortest path for each commodity $k \in K$ in parallel. For a small number of commodities the overhead of parallelization is not worth the time gained however there is a positive payoff for a large number of commodities.

Similarly, we can calculate the shortest path for the physical subproblem, that is, finding the shortest physical path realization of logical link (the shortest grooming path). That is we evaluate $\{SP(\mathcal{P}, \theta_l, s, t) : l = (s,t) \in L\}$ in parallel.

2 An embarrassingly parallel problem can be parallelized trivially, as there are little to none dependencies between tasks, as such each task can be run in parallel with little change

5.3.2 Removal of paths

Column generation is used in order to avoid generating all path variables: A subset of paths are used, and paths are only added if they (possibly) improve the solution value. Over time however, some paths might not be required anymore and only bloat the problem, increasing the computation time required to solve the problem.

In order to avoid a unnecessary inflation in the computation time required to solve the problem, we remove unused paths. If a path has not been used in the basis of the LP for n amount of iterations, we remove the path.

In order to implement this, for each $p \in P'$, we track the number of iterations since it was used in the basis with lcb_p. If $lcb_p > n$, then we remove p from that set of available paths: $P' \leftarrow P' \setminus \{p\}$

We do this for both the physical subproblem, and the logical subproblem.

The value of n is problem specific however, with some small values negatively affecting solution time, thus the optimal value of n needs to be determined experimentally per problem instance. A large value, typically does not worsen the computation time and is typically set to 500, removing a path if after 500 iterations it was not in the basis.

5.4 RESULTS

5.4.1 General model verification

In order to verify that the model is correct, it was implemented it in two different ways.

For the first, a Haskelll library[3] that interfaces with the CPLEX C Callable library was developed, together with a complete implementation. The library serves as the API in which we are able to interface with CPLEX in Haskell. Haskell is a pure, lazy, functional programming language with an emphasis on strong typing. These strong types help ensure program correctness.

Secondly, the model was implemented with C++ using CPLEX's CONCERT library. This will serve as a reference with which the Haskell implementation can be benchmarked against.

For the dataset, the physical layers were obtained from SNDLibs problem instances. The weights were converted into single modules, containing capacity and cost. Both implementations were run against this small dataset. The capacity is a random number in the set $\{1000, 2000, 3000, 4000\}$. The logical layer is based on the physical layer. A quarter of the physical layer edges were samples to form the logical layer links. For the costs and capacities guassian noise

3 The library can be found at `https://github.com/stefan-j/cplex-haskell`

was added in order to deter a transparent topology. Table 1 some characteristics of this particular dataset.

Table 1: SNDlibs datasets used for comparing the Haskell and C++ implementation

Dataset	|V|	|E|	|L|	|D|
newyork	16	49	13	120
polska	12	18	5	33
abilene	12	15	5	66
germany	17	26	7	60
dfngwin	11	47	12	11
diyuan	11	42	11	105
atlanta	15	22	6	11
geant	22	22	6	112
pdh	11	34	9	12

Both of these implementations are based on the path-flow module formulation, **MLPILNS**, as given in Section 4.9. For these comparison, single module constraints aren't used, as there is only a single module per edge.

The results are given in table 2. The objective value and time (in seconds) is given.

The performance difference is due to an implementation detail. In the Haskell implementation, the shortest path problem was solved over multiple commodities in parallel. Using parallelization has some overheads and is detrimental for small datasets. The C++ implementation does not make use of any parallelization and only runs on a single core. C++ is in general a faster language than Haskell, however since most of the time is on the branch and bound process, and solving LP problems within CPLEX, this performance difference is negligible.

Table 2: Comparison between the Haskell and C++ implementations

	C++		Haskell	
Dataset	Objective value	Time (s)	Objective value	Time (s)
newyork	960800	874.03	960800	621.31
polska	55841	22.08	55841	20.47
abilene	20118	24.70	20118	18.05
germany	39126	24.61	39126	23.53
dfngwin	31162	14.57	31162	18.07
diyuan	311100	78.12	311100	53.02
atlanta	73213	20.21	73213	23.50
geant	41821	97:36	286611	71.50
pdh	28661	38.03	28661	36.50

The objective value is the same for both instances, confirming that the implementations match. From hereforth we proceed only with the Haskell based implementations.

5.4.2 Benders and column generation comparison with arc-based

The Benders decomposition model with column generation was tested against the arc-based model. Ideally we would like to compare the Benders only decomposition, however it faces the same problems as the path-flow model, in that there are simply too many paths. For the general path-flow model all possible physical paths need to be generated for each link, and all possible logical paths for each commodity. This is infeasible for multilayer models, as there are simply too many.

Benders decomposition with column generation (**MLPILNS**) is thus compared against the arc-based model (**MLAILNS**). The dataset is similar to that of the previous, as can be seen in table 3. The number of logical links was increased, as well as the number of demands. Logical links and physical edges were allowed to have multiple modules. In this test more than one module may be selected.

Table 3: SNDlibs datasets used for comparing Benders decomposition with arc-based model

Dataset	$\|V\|$	$\|E\|$	$\|L\|$	$\|D\|$
newyork	16	49	26	200
polska	12	18	10	66
abilene	12	15	10	112
germany	17	26	14	101
dfngwin	11	47	24	22
diyuan	11	42	22	140
atlanta	15	22	12	22
geant	22	22	12	262
pdh	11	34	18	24

Table 4 shows the results. The objective value of each method and the time (in minutes) is given. The maximum memory usage is also shown. Each instance was fully solved to optimality. The objective value for both is the same, and only shown once.

These benchmarks were run on Google Cloud Engine (GCE) virtual Instance with 40 vCPUs, 20 cores Xeon Skylake processor, with 64GB Memory. There is no clear trend, however Benders decomposition and column generation uses less memory for larger instances. The arc-based method is slightly faster. Column generation only adds path variables to the basis as needed. We calculate how many iterations the path variables weren't in the basis. For both the logical and physical subproblem, path variables get removed from the basis if they weren't used in the last 40 iterations. This variable is a hyperpa-

Table 4: Comparison Bender's decomposition with column generation against the Arc-based formulation

		Arc		Benders + column	
Dataset	Objective value	Time (m)	Max Mem	Time (m)	Max Mem
newyork	820700	423.77	43107	365.55	32414
polska	45151	73.28	5032	202.37	6061
abilene	20318	92.87	12200	100.03	1024
germany	22126	81.56	9341	163.43	9291
dfngwin	41862	50.56	3021	66.07	3601
diyuan	261920	312.48	40123	412.03	2012
atlanta	73233	20.93	3203	44.22	4021
geant	21421	844:31	51123	713.84	32021
pdh	20261	38.82	4132	71.66	3123

rameter and can be adjusted per problem instance, however for our experiments we keep it fixed at 40 for all problems.

The arc-based model fares surprisingly well, however when formulating the problem as arc-based, the solver, CPLEX 12.6 applies multiple cuts which help find a much better lower bound. For example, on the geant instance CPLEX is able to find 3 Multicommodity flow cuts and, 47 Flow cuts, 35 Mixed integer rounding cuts and 6631 implied bound cuts.

5.4.3 Benders decomposition with- and without warm-start

A warm start was developed that applies Benders decomposition and column generation, as described in Section 5.1.1. In regular column generation we keep adding path variables until no improvement can be found. For this experiment we only add one path per iteration for each link and commodity. Using a heuristic we are able to reduce the upper bound the solver starts with. The datasets remain the same as in the previous section, as per table 3.

Table 5 compares the results between the vanilla Bender's decomposition with column generation without a warm start and with a warm-start.

We omit giving a percentage improvement, as the warmstart only appears to be giving a fixed time improvement, however this varies per problem instance.

5.4.4 Benders decomposition with- and without rounding cuts

The Benders feasibility cuts were strengthened, as described in Section 5.2. The dataset used for comparison is the same as in the previous section and is summarized by table 3. The base comparison

Table 5: Comparison between not using a warm-start and using a warm-start

		Without warm-start	With warm-start
Dataset	Objective value	Time (m)	Time (m)
newyork	820700	365.55	321.42
polska	45151	202.37	189.44
abilene	20318	100.03	95.09
germany	22126	163.43	140.81
dfngwin	41862	66.07	61.35
diyuan	261920	412.03	390.03
atlanta	73233	44.22	51.13
geant	21421	713.84	700.91
pdh	20261	71.66	69.84

Table 6: Comparison between with rounding cuts and without rounding cuts

		Without cuts	With cuts	
Dataset	Objective value	Time (m)	Time (m)	Improvement
newyork	820700	365.55	302.17	1.21
polska	45151	202.37	170.81	1.18
abilene	20318	100.03	97.59	1.02
germany	22126	163.43	151.99	1.07
dfngwin	41862	66.07	59.00	1.12
diyuan	261920	412.03	370.47	1.11
atlanta	73233	44.22	42.79	1.03
geant	21421	713.84	612.25	1.17
pdh	20261	71.66	67.22	1.07

is Bender's decomposition with column generation (**MLPILNS**). No warm-start was used.

Table 6 compares the results between rounding the Bender's cuts and regular Bender's cuts. The Improvement is measured as the time of the old version (only Benders with column generation), divided by the time of the improved version (strengthened cuts), or $\frac{T_{\text{benders}}}{T_{\text{cuts}}}$. The results are quite signficant for larger problem instances.

5.4.5 Survivability

Survivability is covered in Section 4.11. The dataset is the same as the one shown in 3, however diyuan, newyork and geant were omitted. Bender's decomposition with column generation was used to solve these instances. No warm-start or rounding cuts were added.

The results are given in table 9. These benchmarks were run on the same GCE virtual as the previous section, however the memory was

Table 7: Results when adding survivability

		Survivability	
Dataset	Objective value	Time (m)	Memory (mB)
polska	60612	3321.37	101231
abilene	34312	2532.59	95095
germany	36128	2021.44	81203
dfngwin	51234	843.21	66812
atlanta	83231	515.91	34106
pdh	39631	808.31	73393

increased to 128GB. The added survivability significantly increased the computational difficulty. Further work could include looking into heuristics for survivability.

5.4.6 Larger problem instances

We have shown in the previous section that it is possible to improve the computational performance of the path-based model, using various ways. In this section we compare Bender's decomposition with column generation, together with a warm-start and strengthed cuts, to the arc-based model, on larger problem instances. Survivability is not covered in this experiment. This larger dataset is summarized by table8.

Table 8: SNDlibs datasets

Dataset	|V|	|E|	|L|	|D|
sun	27	102	51	67
norway	27	51	26	702
india35	35	80	40	595
germany50	50	88	44	662
france	25	45	23	300
nobeleu	28	41	22	378
janosus	26	84	42	650
ta1	24	55	28	396
ta2	65	108	54	1869

These benchmarks were run on the same GCE virtual instance, although with 256GB Memory. The search was stopped after 24 hours (1440 minutes) for each problem instance, if required. The maximum memory usage, total time, as well as the the optimality gap was recorded.

The only problem that was fully solved within the timelimit was the sun instance. The arc-based formulation did well on the france

Table 9: Benders decomposition with column generation on a large dataset

		Benders			Arc	
Dataset	T (m)	Mem (mB)	Gap	T (m)	Mem (mB)	Gap
sun	1312.48	50193	0.0	900.95	60123	0.0
norway	-	92145	45.1	-	128610	40.8
india35	-	40312	22.5	-	44101	23.2
germany50	-	91209	30.7	-	101233	34.4
france	-	41533	4.1	-	69581	1.4
nobeleu	-	66123	5.4	-	78123	7.8
janosus	-	80012	25.2	-	99831	30.2
ta1	-	67521	10.6	-	76234	14.6
ta2	-	140120	80.3	-	224128	79.9

instance and ended with a better optimality gap. A large part of the memory is used for the branch and bound method.

Although not shown, the arc-based formulation finds a better integer solution early on, than the Bender's formulation, however aftewards it progresses much more slowly. A guess would be that we remove unnecessary path variables in the column generation model, and only add paths as necessary. This keeps the size of the LP problem we solve at each node smaller. Comparing the optimality gap the arc-based formulation faires quite well. In general the memory usage of the Bender's with column generation method is better.

Overall there is still work to be done for larger problem instances for integrated multilayer network models.

6 NETWORK MODELS

6.1 SURVIVABLE DWDM

We start by extending the basic model to for WDM network. WDM is inherently a multilayer network. At the bottom we have an optical physical layer, with WDM on top. As usualy, we minimize the total installation cost.

For this network we may choose to install a fiber or not, with each fiber accomodating a maximum number of lightpaths, denoted by B. Modern DWDM systems typically employ 40 or 80 wavelengths per channel, which may carry 10 or 40 Gbit/s per wavelength. The problem of determining which wavelengths should be assigned to each lightpath is not covered here, as it is a difficult problem on its own, but will be covered in a later section.

We proceed under the assumptions that there are an unlimited number of wavelengths available, and that each cross-connect is able to convert wavelengths. Under these assumptions we are able to focus on developing a general survivable WDM network. In the next section we cover a two-part model that solves the network topology and routing and wavelength assignment problem.

In this model a single physical edge represents a single physical fiber, and we solve which modules should be installed on a physical fiber. A module in this context aggregates the costs of the optical cross connects at both endpoints. The capacity of the module is the number of wavelengths to be installed, and we have r capacity (measured in Gbits) available per wavelength. We use B to denote the maximum number of wavelengths to be used in the system; this would also restrict the number of wavelengths per fiber to B. Installing a fiber incurs a cost of c_e. Here we assume that the cost of the physical fiber, as well as the trenching cost is captured by c_e.

Installing fiber on an edge requires work to dig the trench and laying down the fiber $t(l)$, as well as the cost of the fiber itself $f(l)$ as well as any additional costs, j. Here the both trenching and fiber costs can be seen to be dependent on l, where it is plausible to assume that the relation is linear. Thus the actual cost of installing fiber on an edge is $c(l) = t(l) + f(l) + d$. We don't model the actual costs though, and rather rely on our data to provide the actual costs. Hence the cost is obtained from the data, and we can assume that it already takes length and other factors into account.

For our decision variables we have $x_e \in \{0, 1\}$ indicating whether fiber is installed on edge e. This is one way of modelling whether a fiber is to be employed on an edge. With this approach the data set needs to provide multiple parallel edges between two nodes, in order for there to be multiple physical fibers which can be utilized. Another approach is to let x_e be integer, and assume the data allows multiple fibers to exist on a single edge. This alternative approach is used in a later model. y_a^m indicating the number of capacity modules of type m are installed on edge a.

We allow modular capacity modules on the logical layer, for the reason that wavelength channels typically exhibit economies of scale behavior, that is, the cost of large Gbit/s channels is cheaper when compared by capacity per price.

The set P_k contains all logical paths for commodity k. The set P_l contains all logical paths that go over link l. P_a is a set containing all lightpaths that go over edge a. The set M_a contains activatible channels. V is a set containing all the graph nodes. The set E contains all physical edges. The set L contains all logical links. K is the set containing all commodities.

For the decision variables, $x_a \in \{0, 1\}$ indicates whether a fiber line is active on edge e. $y_a^m \in \mathbb{N}^0$ indicates the number of channels of type m on edge a. $f_p^E \in \mathbb{R}$ represents the amount of flow on physical path p. $f_p^L \in \mathbb{R}$ indicates the amount of flow on logical path.

For the problem parameters, we have a demand d_k for each commodity k. B is the maximum number of wavelengths per fiber. c_a is the installation cost of fiber on physical edge a. k_m is the amount of wavelengths given by m. r is the capacity provided per wavelength.

$$\min \quad \sum_{a \in E} c_e x_a + \sum_{l \in L} \sum_{m \in M_a} c_m y_a^m \tag{6.1}$$

s.t.

$$\sum_{p \in P_k} f_p^L = d_k, \qquad \forall k \in K \tag{6.2}$$

The total physical flow realized from a logical link needs to be at least double the flow on that link, for the diversification.

$$2 \sum_{p \in P_l} f_p^L \leqslant \sum_{p \in l} f_p^E, \qquad \forall l \in L \tag{6.3}$$

$$\sum_{p \in P_a} f_p^E \leqslant \sum_{m \in M_a} y_a^m r k_m, \qquad \forall a \in E \tag{6.4}$$

Where k_m is the number of wavelengths that can be installed, r is the capacity per wavelength and y_a^m is a binary decision variable indicating whether module m is installed on fibre a.

Note that the path p in (6.2) is a commodity path, and in (6.4) p is a lightpath that spans multiple fiber links.

When there is a positive number of modules installed on edge e we need to activate the underlying physical optical fiber connection. We do so using *if-then modelling*. We also limit the number of modules on a given fiber, since there can only be a maximum of B wavelengths employed.

$$\sum_{m \in M_a} y_a^m \leqslant Bx_a, \qquad \forall a \in E \tag{6.5}$$

Together with (6.3) we need to ensure we only maximally send half of the required flow over a physical edge. This keeps us from putting all our eggs in one basket; should a physical edge fail, there is sufficient capacity on another path in order to meet the required demand.

$$\sum_{\substack{p \in l \\ :a \in p}} f_p^E \leqslant \sum_{p \in P_l} f_p^L, \qquad \forall l \in L, a \in E \tag{6.6}$$

int

$$x_a \in \{0,1\}, \forall a \in E$$

$$y_a^m \in \{0,1\}, \forall a \in E, m \in M_a$$

The physical and logical layers are intricately connected in this model, due to constraint sets (6.3) and (6.6). Thus the problem will be separated into a master problem and a single subproblem.

The master problem contains the wavelength constraint set and the objective:

min

$$\sum_{a \in E} c_e x_a + \sum_{l \in L} \sum_{m \in M_a} c_m y_a^m \tag{6.7}$$

s.t.

$$\sum_{m \in M_a} y_a^m \leqslant Bx_a, \qquad \forall a \in E \tag{6.8}$$

For the subproblem we introduce the auxiliary α variable, which we minimize in the objective function, as was done in section 4.7, the details are derived mechanically and are omitted here. The dual is more interesting as we obtain the pricing problem and required cut from it.

We obtain the dual subproblem as:

$$\max \qquad \sum_{k \in K} d_k \pi_k + \sum_{a \in E} \sum_{m \in M_a} y_a^{*m} k_m \mu_a \tag{6.9}$$

s.t.

$$\sum_{a \in E} \mu_a = -1 \tag{6.10}$$

$$\pi_k + \sum_{l \in L} \left(2\eta_l - \sum_{a \in E} \gamma_a^l \right) \leqslant 0, \qquad \forall k \in K, p \in P' \tag{6.11}$$

$$-\eta_l + \sum_{a \in E} \left(\mu_a + \gamma_a^l \right) \leqslant 0, \qquad \forall l \in L, p \in P' \tag{6.12}$$

And the Benders cut we need to add is obtained as:

$$0 \geqslant \sum_{k \in K} d_k \pi_k^* + \sum_{a \in E} \sum_{m \in M_a} y_a^m rk_m \mu_a^* \tag{6.13}$$

On the physical layer we have the fixed charge network design problem (FCNDP), with a capacitated multi commodity flow on the logical layer.

Note that a logical link may be more than one lightpath, in which case it can be considered as the aggregate of all constituent lightpaths.

6.2 DWDM RWA

In the previous section we assume there were a unlimited number of wavelengths available and that wavelength conversion took place. Realistically this is not possible. In this section we consider the Routing and Wavelength Assignment (RWA) problem in optical networks employing WDM for which the traffic is known in advance, i.e. the network is assumed to be static.

Each lightpath in the network is assigned a wavelength. A lightpath must use the same wavelength on each physical fibre along its path. Furthermore, all lightpaths traversing the same physical fibre, must be assigned distinct wavelengths.

The overall problem is solved in two steps:

- Solve the minimum cost network topology. The topology and routing of the network is solved and the actual lightpaths to be used is obtained.

- Determine the wavelength for each lightpath, obeying the wavelength constraints.

The problem is separated, as solving both in a single model is not feasible for large networks.

For the first part, the model is similar to the one presented in the previous section. We omit survivability and use a path based formulation in order to focus on the wavelength assignment problem.

DWDM:

$$\min \quad \sum_{a\in E} c_a x_a + \sum_{l\in L} \sum_{m\in M_a} c_m y_a^m \tag{6.14}$$

s.t.

$$\sum_{p\in P_k} f_p^L = d_k, \qquad \forall k \in K \tag{6.15}$$

$$\sum_{p\in P_l} f_p^L \leqslant \sum_{p\in l} f_p^E, \qquad \forall l \in L \tag{6.16}$$

$$\sum_{p\in P_a} f_p^E \leqslant \sum_{m\in M_a} y_a^m rk_m, \qquad \forall a \in E \tag{6.17}$$

$$\sum_{m\in M_a} y_a^m \leqslant Bx_a, \qquad \forall a \in E \tag{6.18}$$

$$\sum_{\substack{p\in l\\ :a\in p}} f_p^E \leqslant \sum_{p\in P_l} f_p^L, \qquad \forall l \in L, a \in E \tag{6.19}$$

with integral bounds

$$x_a \in \{0,1\}, \forall a \in E$$

$$y_a^m \in \{0,1\}, \forall a \in E, m \in M_a$$

For performance this problem is decomposed using Bender's decomposition and only a subset of paths is used, with possible new paths obtained using column generation and proceeds in the same manner as in the previous section. More details of this can be found in the chapter 4.

Routing and Wavelength Assignment:

Subsequently, we utilize the answer of the cost minimization problem in order to solve the wavelength assignment problem. The light-paths are obtained from the solution of the first problem, these are the physical paths. We also use the number of wavelengths used from the first problem.

The set W denotes all possible wavelengths. The set P denotes all lightpaths $\{p \in P_a : \forall a \in E\}$.

The decision variable λ_p^w denotes whether wavelength w is used for the lightpath p. Since the model is formulated in a path based manner, the continuity constraint is already satisfied, as we are assigning a wavelength w to lightpath p.

The wavelength assignment problem is formulated as follows:

$$\min \quad \sum_{w \in W} \sum_{p \in P} \lambda_p^w$$

That is, we are minimizing the total number of wavelengths used over all paths.

s.t.

The number of wavelengths to be installed needs to match the number we solved in the previous problem:

$$\sum_{w \in W} \sum_{p \in P_a} \lambda_p^w = \sum_{m \in M_a} k_m y_a^m, \qquad \forall a \in E \tag{6.20}$$

All wavelengths are required to be distinct over a fiber.

$$\sum_{p \in P_a} \lambda_p^w \leqslant 1, \qquad \forall a \in E, w \in W \tag{6.21}$$

The set P_k contains all logical paths for commodity k. The set P_l contains all ogical paths that go over link l. P_a is a set containing all lightpaths that go over edge a. The set M_a contains activatible channels. V is a set containing all the graph nodes. The set E contains all physical edges. The set L contains all logical links. K is the set containing all commodities.

For the decision variables, $x_a \in \{0, 1\}$ indicates whether a fiber line is active on edge e. $y_a^m \in \mathbb{N}^0$ indicates the number of channels of type m on edge a. $f_p^E \in \mathbb{R}$ represents the amount of flow on physical path p. $f_p^L \in \mathbb{R}$ indicates the amount of flow on logical path. $\lambda_p^w \in \{0, 1\}$ indicates whether wavelength w is used for lightpath p.

For the problem parameters, we have a demand d_k for each commodity k. B is the maximum number of wavelengths per fiber. c_a is the installation cost of fiber on physical edge a. k_m is the amount of wavelengths given by m. r is the capacity provided per wavelength.

6.3 ETHERNET OVER DWDM

The model is based on the multicommodity flow problem and flow variable are utilized in order to determine the appropriate amount of flow to send over a path. Given two graphs, $\mathcal{P} = (N, E)$ and $\mathcal{L} = (N, L)$, the network is considered multi layer and each graph constitutes a layer. Note that the set of nodes is shared. The networks considered here are opaque, that is, E contains the same set of edges as L. We loosely use the term *edge* to refer to a physical edge $e \in E$, and *link* to refer to a logical edge $l \in L$, outside of the usual graph theoretic definitions.

The input graphs, $\mathcal{P}$ and $\mathcal{L}$, are undirected, weighted and non-cyclic. When demand is sent over an edge, we assume the same amount can be sent in the opposite direction. This can be realized practically by installing the same fiber configuration in the opposite direction. The cost of installing a fiber is an aggregation of the forward and backward fibers, and is already taken into account in c_e. The same is true for the lightpaths, with the additional wavelengths installed in the separate backward fiber.

Multiple physical lightpaths may have the same terminal nodes, which results in parallel logical links. In this model these parallel links are aggregated into a single link. Thus the decision variable y_l indicates the number of wavelengths over all possible physical paths which realize the logical link l.

Each lightpath in the network is assigned a wavelength. A lightpath must use the same wavelength on each physical fiber along its path. Furthermore, all lightpaths traversing the same physical fiber, must be assigned distinct wavelengths. The problem of assigning wavelengths to lightpaths is solved separately, due to the high computational capacity that it requires.

The set M contains available routers. At each node $i \in N$ a single router $m \in M$ may be installed. The router needs to be able to switch all incoming and outgoing traffic.

For computational reasons the set of edges E only contains a undirected edge once, and for each $e = (i, j) \in E$ we have that $i \leqslant j$. The same is true for the set of logical links L. When generating flow variables, we omit flow variables on edges which do not exist. Thus

$$\sum_{w \in V} Q^k_{v,w}, \qquad \forall v \in V,$$

can be simplified to

$$\sum_{w \in V : (w,v) \in L} Q^k_{v,w}, \qquad \forall v \in V$$

The same is done for the lightpath flow variables.

For the decision variables we have $z_e \in \mathbb{N}^0$ indicating the number of fibers to be installed on edge e, and y_l indicating the number of wavelengths to be installed on logical link l.

For this model we develop both a path-flow model and an arc-based model. For the path model we apply the usual decompositions.

6.3.1 Path model

$$\min \quad \sum_{i \in V} \sum_{m \in M_i} c_m x_i^m + \sum_{l \in L} \gamma y_l + \sum_{a \in E} c_a z_a \tag{6.22}$$

s.t.

$$\sum_{p \in P_k} f_p^L = d_k, \qquad \forall k \in K \tag{6.23}$$

$$\sum_{p \in P_l} f_p^L \leqslant r\delta y_l, \qquad \forall l \in L \tag{6.24}$$

$$\sum_{p \in l} f_p^E = r\delta y_l, \qquad \forall l \in L \tag{6.25}$$

$$\sum_{l \in L_a} y_l \leqslant B z_a, \qquad \forall a \in E \tag{6.26}$$

$$\sum_{m \in M_i} x_i^m \leqslant 1, \qquad \forall i \in V \tag{6.27}$$

$$\sum_{m \in M_i} k_m x_i^m \geqslant \sum_{l \in L_i} r\delta y_l + \sum_{\substack{(a,b) \in K \\ a=i \vee b=i}} d_{(a,b)}, \qquad \forall i \in V \tag{6.28}$$

6.3.2 Path model Decomposition

Bender's decomposition is applied in order to separate **EWDM** into a reduced master problem, and two subproblems.

For the reduced master problem, the objective function and the routing constraints are obtained as:

RMP

$$\min \quad \sum_{i \in V} \sum_{m \in M_i} c_m x_i^m + \sum_{l \in L} \gamma y_l + \sum_{a \in E} c_a z_a \tag{6.29}$$

s.t.

$$\sum_{m \in M_i} k_m x_i^m \geqslant \sum_{l \in L_i} r\delta y_l + \sum_{\substack{(a,b) \in K \\ a=i \vee b=i}} d_{(a,b)}, \qquad \forall i \in V \tag{6.30}$$

$$\sum_{m \in M_i} x_i^m \leqslant 1, \qquad \forall i \in V \tag{6.31}$$

The subproblems are separated into a logical- and physical subproblem.

The auxiliary variable α is introduced that serves as capacity shortfall. The subproblems are solved when a feasible integer solution is obtained during the branch and bound process. Thus the variables y_l^* and z_l^* denote the current integer solution of the respective decision variables.

For the logical subproblem we have:

LOG:

$$\min \alpha \tag{6.32}$$

s.t.

$$\sum_{p \in P'_k} f_p^L = d_k, \qquad \forall k \in K \tag{6.33}$$

$$\sum_{p \in P'_l} f_p^L - \alpha \leqslant r\delta y_l^*, \qquad \forall l \in L \tag{6.34}$$

For the physical subproblem we have:

PHYS

$$\min \quad \alpha \tag{6.35}$$

s.t.

$$\sum_{p \in l} f_p^E = r\delta y_l^*, \qquad \forall l \in L \tag{6.36}$$

$$\sum_{p \in P'_a} f_p^E - \alpha \leqslant Bz_a^*, \qquad \forall a \in E \tag{6.37}$$

Note that in the above constraints we are using a separate set of paths, $P'_k \subseteq P_k$, $P'_l \subseteq P_l$ and $P'_a \subseteq P_a$. Initially $|P'_k| = 1, \forall k \in K$ and $|p \in l| = 1, \forall l \in L$, and only contain a single shortest path. We only generate new paths as needed using column generation - we try to find new flow variables in order to reduce the objective function by identifying variables with negative reduced cost.

For the logical subproblem, we obtain the dual as:

LOGD

$$\max \quad \sum_{k \in K} \pi_k d_k + \sum_{l \in L} r\delta y_l^* \theta_l \tag{6.38}$$

s.t.

$$\sum_{l \in L} \theta_l = -1 \tag{6.39}$$

$$\pi_k \leqslant \sum_{l \in L_k} -\theta_l, \qquad \forall k \in K, p \in P'_k \tag{6.40}$$

From (6.40), the pricing problem can be more clearly written as (6.41), which can be seen as a shortest path problem.

$$\pi_k = \min_{p \in P_k} \left\{ \sum_{l \in L} -\theta_l \right\} \tag{6.41}$$

For every commodity $k = (a, b)$ we calculate the shortest path from a to b on graph $\mathcal{L}$ given edge weights $-\theta_l, \forall l \in L$.

We need to see if we can find a shorter path length than π_k for commodity k. If this is the case then constraint (6.40) is violated and we need to add the path, and resolve the subproblem in order to see if there are any additional variables to add.

Let $\mathrm{SP}(\mathcal{G}, W, a, b)$ be a function returning the shortest path between a and b on graph $\mathcal{G}$ given edge weights W. Let (d, p) denote the result, where d is the distance and p is the path.

For every commodity k we solve the shortest path using Dijkstra's algorithm, we add the resulting path p to P'_k, if $d < \pi_k$.

When there are no more variables to add, we add the Benders cut (6.42) to the master problem.

$$\sum_{l \in L} r\delta y_l \theta^*_l \leqslant \sum_{k \in K} -\pi^*_k d_k \tag{6.42}$$

Proceeding in the same manner, we obtain the dual of the physical subproblem as:

PHYSD

$$\max \quad \sum_{l \in L} \eta_l r\delta y^*_l + \sum_{a \in E} \mu_a B z^*_a \tag{6.43}$$

s.t.

$$\sum_{a \in E} \mu_a = -1 \tag{6.44}$$

$$\eta_l + \sum_{a \in E_l} \mu_a \leqslant 0 \qquad \forall l \in L, p \in P'_l \tag{6.45}$$

With the Bender's cut obtained as:

$$\sum_{l \in L} \eta_l^* r \delta y_l + \sum_{a \in E} \mu_a^* B z_a \leqslant 0 \tag{6.46}$$

Variables are added as needed using column generation; in the same manner variables are removed from the subproblems after a certain number of iterations has passed and they have have not been used in the basis.

6.3.3 Arc Model

The set P_k contains all logical paths for commodity k. The set P_l contains all ogical paths that go over link l. P_a is a set containing all lightpaths that go over edge a. The set M_a contains activatible channels. V is a set containing all the graph nodes. The set E contains all physical edges. The set L contains all logical links. K is the set containing all commodities. Λ is the set of all wavelengths.

For the decision variables, $X_i^m \in \{0, 1\}$ indicates whether router m installed at node i. $Y_l \in \mathbb{N}^0$ indicates the number of lightpaths on link l. $Z_a \in \mathbb{N}^0$ indicates the number of fibers installed at physical edge e. $Q_l^k \in \mathbb{R}^+$ indicates the amount of flow for commodity k on link l. $W_e^l \in \mathbb{R}^+$ indicates the amount of flow for link l on edge e. $\lambda_p^w \in \{0, 1\}$ inndicates whether wavelength w is installed on path p.

For the problem parameters, we have a demand d_k for each commodity k. b is the maximum number of wavelengths per fiber. c_a is the installation cost of fiber on physical edge a. c_m is the cost of router m. γ_l is the cost of installing a lightpath on link l. k_m is the amount of wavelengths given by m. r is the capacity provided per wavelength. k_m is the capacity provided by router m.

The model **EWDMA** describes the integrated approach.

EWDMA (Ethernet over WDM Arc):

$$\min \quad \sum_{l \in L} \gamma_l Y_l + \sum_{i \in V} \sum_{m \in M} c_m X_i^m + \sum_{a \in E} c_a Z_a \tag{6.47}$$

s.t.

$$\sum_{w \in V} Q_{(v,w)}^k - \sum_{w \in V} Q_{(w,v)}^k = \begin{cases} d_k & v = s \\ -d_k & v = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall k = (s,t) \in K, v \in V \tag{6.48}$$

$$\sum_{k \in K} \left(Q_{(a,b)}^k + Q_{(b,a)}^k \right) \leqslant r Y_l, \qquad \forall l \in L \tag{6.49}$$

$$\sum_{w \in V} W^l_{(v,w)} - \sum_{w \in V} W^l_{(w,v)} = \begin{cases} Y_l & v = s \\ -Y_l & v = t \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in V, l = (s,t) \in L \tag{6.50}$$

$$\left(W^l_{(s,t)} + W^l_{(t,s)}\right) \leqslant \frac{1}{2} Y_l, \qquad \forall (s,t) \in E, l \in L \tag{6.51}$$

$$\sum_{l \in L} \left(W^l_{(s,t)} + W^l_{(t,s)}\right) \leqslant b Z_a, \qquad \forall a = (s,t) \in E \tag{6.52}$$

$$\sum_{m \in M} X^m_i \leqslant 1, \qquad \forall i \in V \tag{6.53}$$

$$\sum_{m \in M} k_m X^m_i \geqslant \sum_{l \in L_i} r Y_l + \sum_{\substack{k=(a,b) \in K \\ a=i \vee b=i}} d_k, \qquad \forall i \in V \tag{6.54}$$

The objective function (6.47) minimizes the total capital expenditure cost of the network, which includes the wavelength costs, fiber installation costs, and routing equipment costs.

Constraints (6.48) require the flow to match the demand of each commodity; each commodity $k = (s,t)$ has a certain bandwidth requirement called the demand, all outgoing edges from node s send d_k flow and all incoming edges to node t absorb d_k flow.

Constraints (6.49) require that a sufficient number of lightpaths are installed on a link in order to satisfy capacity requirements.

Constraints (6.50) require that the physical flow be equal to the logical capacity. The flow is sent in the same manner as for the commodities in constraints (6.48)n.

Constraints (6.51) provide survivability to the model by means of physical diversification. For each $l \in L, e \in E$ we require that the total flow over W^l_e is less than or equal to half of the link capacity. In the event that a physical edge fails, half of the required link capacity can be routed over another physical path. Constraints (6.52) limit the maximum number of wavelengths on an edge to B; additionally, it indicates that a fiber should be installed on edge a if there are wavelengths assigned to it. Constraints (6.53) state that only a single router may be installed at a node. Constraints (6.54) require that the router have sufficient capacity in order to route the traffic through the node.

6.3.4 Top-down model

A top-down approach sequentially solves each layer from the top, downwards. The top-down is similar to the integrated model, however, the problem is separated into two, due to its block-like structure. The solved capacities from the top layer are solved in the model **TD-TOP** and are used to solve for the capacities on the bottom layer in model **TDBOT**. This simulates the design process commonly used by network providers when planning a network sequentially.

The top-most layer is solved first for the lightpath capacities:

TDTOP (Top-down Top):

$$\min \quad \sum_{l \in L} \gamma_l Y_l + \sum_{i \in V} \sum_{m \in M} c_m X_i^m \tag{6.55}$$

s.t.

$$\sum_{w \in V} Q^k_{(v,w)} - \sum_{w \in V} Q^k_{(v,w)} = \begin{cases} d_k & v = s \\ -d_k & v = t \\ 0 & \text{otherwise} \end{cases} \quad \forall k = (s,t) \in K, v \in V \tag{6.56}$$

$$\sum_{k \in K} \left(Q^k_{(a,b)} + Q^k_{(b,a)} \right) \leqslant r Y_l, \qquad \forall l \in L \tag{6.57}$$

$$\sum_{m \in M} X_i^m \leqslant 1, \qquad \forall i \in V \tag{6.58}$$

$$\sum_{m \in M} k_m X_i^m \geqslant \sum_{l \in L_i} r Y_l + \sum_{\substack{k=(a,b) \in K \\ a=i \vee b=i}} d_k, \qquad \forall i \in V \tag{6.59}$$

Since the number of logical wavelengths Y_l is already solved for, it can be used in the subsequent bottom layer problem. The solved fixed value is indicated by Y_l^*.

TDBOT (Top-down bottom):

$$\min \quad \sum_{a \in E} c_a Z_a \tag{6.60}$$

s.t.

$$\sum_{w \in V} W^l_{(v,w)} - \sum_{w \in V} W^l_{(w,v)} = \begin{cases} Y_l^* & v = s \\ -Y_l^* & v = t \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in V, l = (s,t) \in L \tag{6.61}$$

$$\left(W^l_{(s,t)} + W^l_{(t,s)}\right) \leqslant \frac{1}{2} Y^*_l, \qquad \forall (s,t) \in E, l \in L \tag{6.62}$$

$$\sum_{l \in L} \left(W^l_{(s,t)} + W^l_{(t,s)}\right) \leqslant b Z_a, \qquad \forall a = (s,t) \in E \tag{6.63}$$

The overall objective value is then:

$$\sum_{l \in L} \gamma_l Y^*_l + \sum_{i \in V} \sum_{m \in M} c_m X^{*m}_i + \sum_{a \in E} c_a Z^*_a \tag{6.64}$$

The constraints are similar to those from the integrated approach, **EWDMA**.

The top-down model solves the same problem as the integrated approach, however it may be done sequentially. For this reason the top-down solution is used as a warm-start for the integrated model. This decreases the total computational time required.

6.3.5 Results

One focus of this work is to investigate the viability of an integrated multilayer network model over a more traditional top-down model. We evaluate these two approaches by comparing the objective value, as this provides a method to gauge the possible capital expenditure savings for network planners.

Thus we set up an experiment to compare these two approaches, specifically for this Ethernet over WDM problem.

The physical topologies of the network instances were obtained from SNDlib [36]. SNDlib network instances contain modules for each edge. A module provides capacity for a certain cost. The highest cost is taken as the cost of installing the fiber.

The logical layer is generated based on the physical layer. The same nodes and edges are used, resulting in an opaque topology. The cost of installing a wavelength γ_l is taken as the lowest module cost for an edge and scaled down by a factor of ten.

The models were programmatically implemented and solved using IBM ILOG CPLEX. The results were generated on an HP Z1 workstation with 16GB DDR3 Ram and Intel(R) Xeon(R) CPU E3-1245 processor.

The top-down and integrated model were run for each network instance, the top-down result was used as a warm-start for the integrated model. The parameters were kept constant, with $b = 40$ and $r = 10$.

Table 10 displays the list of network instances used to test the integrated and top-down models. The number of logical links |L| and physical edges |E| is equal since an opaque topology is used.

Table 10: SNDlibs datasets

Dataset	$\|V\|$	$\|E\|$	$\|L\|$	$\|D\|$
pdh	11	34	34	24
polska	12	18	18	66
nobelus	14	21	21	91
nobelger	17	26	26	121
atlanta	15	22	22	210
abilene	12	16	16	132
geant	22	36	36	462

Table 11: Comparison between top-down and integrated approach

	Integrated		Top Down		
Dataset	Objective value	Time	Objective value	Time	Π
pdh	1.77×10^8	04:47	2.41×10^8	01:31	1.36
polska	3.34×10^7	01:39	3.66×10^7	00:47	1.09
nobelus	6.11×10^8	05:02	6.43×10^8	01:05	1.05
nobelger	1.43×10^8	09:20	1.53×10^8	07:53	1.07
atlanta	2.16×10^{11}	05:02	2.34×10^{11}	00:07	1.08
abilene	5.85×10^9	00:04	6.75×10^9	00:02	1.15
geant	4.18×10^9	17:36	4.39×10^9	02:50	1.05

Table 11 shows the results obtained from the top-down and integrated approach. For all cases, the integrated approach obtains an improved objective value. The optimality factor is defined as: $\Pi = \frac{\hat{y}}{y}$ where y is the optimal objective value obtained from the integrated approach, and $\hat{y}$ is the objective value from the top-down approach. This factor indicates how far the optimal solution is from the top down approach. The time is given as *mm:ss*. The *Objective value* column refers to the the total capital expenditure cost of the network. From table 11 it is evident that the top-down approach performed poorly when applied to the *pdh*- and *abilene* network instances.

This suggests that network providers may obtain significant cost savings by utilizing an integrated approach for network planning. A top-down approach, when compared to an integrated approach is expected to perform even worse on larger network instances.

7 CONCLUSIONS

In this work we set out to cover the following:

1. Develop a flexible approach to solving multilayer network problems.
2. Apply the developed network model to a Ethernet over WDM network
3. Improve scalability and performance.

For the first point we developed a general mutlilayer network model (Chapter 4). This network model uses modules in order to encapsulate components that provide capacity to the network at a certain cost. These modules can be installed on edges or nodes. Without extending the module to a specific network, these modules allow edges to provide capacity at a certain cost, for example, an Ethernet line. On a node this could be thought of as a router. We also advocate that a integrated multilayer network model reduces the capital expenditure costs compared to the traditional top-down approach (Section 6.3).

For the third point we try to improve the computational performance of the approach. We develop two models, an arc-based model to serve as a reference, and a path-flow model. The path-flow model lends itself to being decomposed, as such we apply Bender's decomposition and column generation. We furthermore develop a primal heuristic to serve as a warm-start for the decomposition model, as well as strengthen the cuts. In Section 5.4 we find that this approach reduces the memory consumption, and provides a better optimality gap for larger problem instances.

Finally we apply the general multilayer network model developed in chapter 4 to different Ethernet over WDM topologies, in Chapter 6. We demonstrate the flexible approach and show that modelling the physical layer and logical layer separately allows the model to easily translate to different network topologies and technologies.

7.1 RESULTS

The results are covered in Section 5.4, and indicate that it is possible to improve the computational performance of the general multilayer network model. In particular by decomposing the problem using Bender's decomposition and column generation the memory usage can be reduced. Furthermore adding a primal heuristic as a warm-start solution with strengthened cuts, reduces the the time taken to fully solve instances. The arc-based model has decent performance as well, and this is due to the many optimization the CPLEX solver is able to perform on this formulation.

When adding survivability to the network model, the memory usage and time taken to fully solve problem instances increase drastically. Furthermore, it is still difficult to solve larger problem instances with this approach.

It should be noted that using the top-down approach as a warm-start, and using the integrated model to solve the problem further will improve the results, even if the computation is stopped before fully solving the model. This is strictly better than only using a top-down solution and we advocate that network planners incorporate such an approach.

Still there is further work warranted to improve the performance of these models.

7.2 FUTURE WORK

There are many areas still left to explore. These are covered below (and is by no means an exhaustive list).

Heuristics are mentioned but not explored. We employ a simple heuristic based on Bender's decomposition and column generation, where we limit the amount of variables generated. This functions quite well as a warm-start for larger solutions, however more traditional local search algorithms can be applied to the problem. Possibile avenues of enquiry would be genetic algorithms, particle swarm optimization or a custom expert heuristic.

Additionally, heuristics could be developed that provide a warm-start solution for the Bender's decomposition subproblems.

Another interesting avenue would be to try and improve the lower bound of the the master problem, as this would reduce the size of the branch and bound tree.

The developed multilayer network model is general enough to be extended to more than two layers. Most of the models covered in this work are 2-layer network models. The general approach is the same for n-layer networks and could be extended to practical $n > 2$ layer networks. This will be difficult to scale computationally, and it may be necessary to explicitely specify paths for upper level edges.

BIBLIOGRAPHY

[1] E. Harstead and R. Sharpe, "Forecasting of access network bandwidth demands for aggregated subscribers using monte carlo methods," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 199–207, 2015.

[2] K. G. Coffman and A. M. Odlyzko, "Internet growth: Is there a moore's law for data traffic?" in *Handbook of massive data sets*. Springer, 2002, pp. 47–93.

[3] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, "A survey on predicting the popularity of web content," *Journal of Internet Services and Applications*, vol. 5, no. 1, p. 8, 2014.

[4] Metro Ethernet Forum, *MEF Technical Specifications*. [Online]. Available: https://www.mef.net/carrier-ethernet/technical-specifications

[5] R. Santitoro, "Metro Ethernet Services,ÄìA Technical Overview," in *Metro Ethernet Forum*, vol. 2006, 2003.

[6] HUAWEI, "Technical White Paper for Multi-Layer Network Planning," University of Zurich, Department of Informatics, Tech. Rep.

[7] M. R. Garey and D. S. Johnson, *Computers and intractability*. wh freeman New York, 2002, vol. 29.

[8] *CPLEX Optimization Studio*. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/

[9] S. Orlowski and R. Wessäly, *An integer programming model for multi-layer network design*. Konrad-Zuse-Zentrum für Informationstechnik, 2004.

[10] S. Orlowski, "Optimal design of survivable multi-layer telecommunication networks," 2009.

[11] J. S. Carson, "Model verification and validation," in *Simulation Conference, 2002. Proceedings of the Winter*, vol. 1. IEEE, 2002, pp. 52–58.

[12] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 37th conference on Winter simulation*. winter simulation conference, 2005, pp. 130–143.

[13] S. Jacholke, M. Grobler, and S. Terblanche, "Development of a multi-layer model for optimal core ethernet resource planning," in *Southern Africa Telecommunication Networks and Applications Conference*, 2016.

[14] ——, "A multilayer approach for solving the ethernet over wdm network design problem," in *Southern Africa Telecommunication Networks and Applications Conference*, 2017.

[15] K. Lai and M. Goemans, "The knapsack problem and fully polynomial time approximation schemes (fptas)," *Retrieved November*, vol. 3, p. 2012, 2006.

[16] S. Aaronson, G. Kuperberg, and C. Granade, "The complexity zoo," 2005.

[17] G. Sierksma, *Linear and integer programming: theory and practice*. CRC Press, 2001.

[18] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[19] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[20] D. G. Luenberger, *Introduction to linear and nonlinear programming*. Addison-Wesley Reading, MA, 1973, vol. 28.

[21] K. G. Murty, "Linear programming," 1983.

[22] G. B. Dantzig, "Maximization of a linear function of variables subject to linear inequalities," *The Basic George B. Dantzig*, pp. 24–32, 2003.

[23] L. G. Khachiyan, "Polynomial algorithms in linear programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72, 1980.

[24] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, pp. 497–520, 1960.

[25] E. Kalvelagen, "Benders decomposition with gams," *web. stanford. edu/class/msande348/papers/bendersingams. pdf*, vol. 7, 2002.

[26] J. Snyman, *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*. Springer Science & Business Media, 2005, vol. 97.

[27] E. H. Aarts and J. H. Korst, "Simulated annealing," *ISSUES*, vol. 1, p. 16, 1988.

[28] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[29] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[30] M. Ali, G. Chiruvolu, and A. Ge, "Traffic engineering in metro Ethernet," vol. 19, no. 2, pp. 10–17, 2005.

[31] S. Keshav, "An engineering approach to computer networking: Atm networks, the internet, and the telephone network," *Reading MA*, vol. 11997, 1997.

[32] A. Zapata, M. Duser, J. Spencer, P. Bayvel, I. de Miguel, D. Breuer, N. Hanik, and A. Gladisch, "Next-generation 100-gigabit metro ethernet (100 gbme) using multiwavelength optical rings," *Journal of lightwave technology*, vol. 22, no. 11, pp. 2420–2434, 2004.

[33] Lucient Technologies, "Carrier Ethernet Defined," 2005.

[34] H. Schmidtke and A. Gibbemeyer, "Five reasons to adopt layer 2 Ethernet switching over DWDM networks now," *Rapport technique, Siemens Networks*, 2006.

[35] G. Rizzelli, A. Morea, M. Tornatore, and O. Rival, "Energy efficient traffic-aware design of on–off multi-layer translucent optical networks," *Computer Networks*, vol. 56, no. 10, pp. 2443–2455, 2012.

[36] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0 survivable network design library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.

[37] S. Orlowski and M. Pioro, "On the complexity of column generation in survivable network design with path-based survivability mechanisms," *Zuse Institute Berlin and Warsaw University of Technology, Tech. Rep*, 2008.

[38] S. Terblanche, R. Wessäly, and J. M. Hattingh, "Survivable network design with demand uncertainty," vol. 210, no. 1, pp. 10–26, 2011.

[39] Y. Lee, Y. Seok, Y. Choi, and C. Kim, "A constrained multipath traffic engineering scheme for mpls networks," in *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 4. IEEE, 2002, pp. 2431–2436.

[40] A. Bley and T. Koch, "Integer programming approaches to access and backbone ip network planning," in *Modeling, Simulation and Optimization of Complex Processes*. Springer, 2008, pp. 87–110.

[41] B. Gendron, T. G. Crainic, and A. Frangioni, "Multicommodity capacitated network design," in *Telecommunications network planning*. Springer, 1999, pp. 1–19.

[42] F. Idzikowski, L. Chiaraviglio, and F. Portoso, "Optimal design of green multi-layer core networks," in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*. IEEE, 2012, pp. 1–9.

[43] T. Engel, A. Autenrieth, and J.-C. Bischoff, "Packet layer topologies of cost optimized transport networks multi-layer netwok optimization," in *Optical Network Design and Modeling, 2009. ONDM 2009. International Conference on*. IEEE, 2009, pp. 1–7.

[44] E. Kubilinskas, P. Nilsson, and M. Pióro, "Design models for robust multi-layer next generation internet core networks carrying elastic traffic," *Journal of Network and Systems Management*, vol. 13, no. 1, pp. 57–76, 2005.

[45] G. Baier, T. Engel, and A. Autenrieth, "Evaluation of survivable ethernet over WDM network architectures in metro ring networks," in *Design and Reliable Communication Networks, 2007. DRCN 2007. 6th International Workshop on*. IEEE, 2007, pp. 1–8.

[46] G. Dahl and M. Stoer, "A cutting plane algorithm for multicommodity survivable network design problems," *INFORMS Journal on Computing*, vol. 10, no. 1, pp. 1–11, 1998.

[47] C. H. Papadimitriou, "On the complexity of integer programming," *Journal of the ACM (JACM)*, vol. 28, no. 4, pp. 765–768, 1981.

[48] L. S. Lasdon, *Optimization theory for large systems*. Courier Corporation, 1970.

[49] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[50] G. L. Nemhauser, "Column generation for linear and integer programming," *Optimization Stories*, vol. 20, p. 64, 2012.

[51] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[52]

[53] L. Ford Jr and D. Fulkerson, "Maximal flow through a network," in *Classic papers in combinatorics*. Springer, 2009, pp. 243–248.

[54] C. Raack, "Capacitated network design-multi-commodity flow formulations, cutting planes, and demand uncertainty," 2012.

[55] S. Chopra, I. Gilboa, and S. T. Sastry, "Source sink flows with capacity installation in batches," *Discrete Applied Mathematics*, vol. 85, no. 3, pp. 165–192, 1998.

[56] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[57] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Foundations of Computer Science, 1975., 16th Annual Symposium on*. IEEE, 1975, pp. 184–193.

[58] M. P. Kleeman, B. A. Seibert, G. B. Lamont, K. M. Hopkinson, and S. R. Graham, "Solving multicommodity capacitated network design problems using multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 449–471, 2012.

[59] A. M. Alvarez, J. L. González-Velarde, and K. De-Alba, "Grasp embedded scatter search for the multicommodity capacitated network design problem," *Journal of Heuristics*, vol. 11, no. 3, pp. 233–257, 2005.

[60] D. S. Johnson, J. K. Lenstra, and A. Kan, "The complexity of the network design problem," *Networks*, vol. 8, no. 4, pp. 279–285, 1978.

[61] R. Hinze, "A simple implementation technique for priority search queues," in *ACM SIGPLAN Notices*, vol. 36, no. 10. ACM, 2001, pp. 110–121.