Disjoint Paths in Directed Networks

with

Length and Distribution Criteria

Marie-Louise Højlund Rasmussen

Supervised by:

Professor Martin P. Bendsøe
Assistant Professor Thomas Britz
Assistant Professor Mathias Stolpe

September 1, 2005, Lyngby, Denmark
Department of Mathematics
Technical University of Denmark

Summary

The main part of this thesis work is to develop a model for solving the problem of finding arc- or node disjoint paths with different length- and distribution criteria in directed networks. As one result, we present a fairly general mathematical model that can incorporate a multitude of criteria, and which is suitable for computations. The model applies to problems that can be modeled as directed networks and in which one wants to transport commodities of certain sizes from sources to terminals in this directed network. The transportation routes constitute connecting paths. Length criteria can be that connecting paths must equal, be less than, or greater than certain lengths, or they must have equal lengths or approximately equal lengths. The distribution criteria include defining where the connecting paths must begin and where they must end, defining forbidden areas, and defining mandatory areas.

Other products of this thesis work are developed algorithms for eliminating occurrences of sub-tours in the solutions and solving large-scale programs. The large-scale programs are decomposed into several sub-programs that are solved separately. There are two kinds of decompositions: using separate regions or overlapping regions. Using overlapping- instead of separate regions gives more flexibility to the solutions. When using overlapping regions, both the number of regions and the size of the overlap decide the behavior of the decomposition algorithm.

A last part of this thesis work is to implement and experiment upon the model and the developed algorithms. The level of difficulty of the problem of finding arc- or node disjoint paths with different length- and distribution criteria in directed networks depends on the posed criteria. Using stricter criteria makes the problem more difficult to solve. However, the decomposition algorithm is able to find solutions in most cases when choosing an appropriate number of regions and size of overlap.

Keywords

Arc disjoint paths, Distribution criteria, Integer multicommodity flow, Large scale problems, Length criteria, Named integer multicommodity flow, Node disjoint paths, Sub-tour elimination

Preface

This thesis is submitted as partial fulfillment of the requirements for the degree of Master of Science in Engineering in Applied Mathematics. The work has been carried out in the period between January 2005 and August 2005 at the Department of Mathematics at the Technical University of Denmark.

The work reported here constitutes the theoretical part of work developed in order to attack an industrial problem and the confidential part of the work is reported in a separate text; the complete thesis work thus involves these two reports. We also remark here that the theoretical work is illustrated by examples from commodity flow in road transport problems; this only resembles the industrial problem in its abstract mathematical form.

It is assumed that the readers of this thesis are familiar with mathematical modeling. However, all elements of the problem are defined such that a reader unfamiliar with the terminology in graph theory still will be able to understand the problem. Some of the discussed problems are best understood if the reader is acquainted with some optimization; however, such knowledge is not necessary. This report is thus also written so as to provide our industrial partner with a background for the material described in the confidential report.

Acknowledgments

My warmest thanks go to my supervisors, Professor Martin P. Bendsøe, Assistant Professor Thomas Britz, and Assistant Professor Mathias Stolpe, for your guidance, time, and motivation. The results from this thesis would not have been the same without your help. Thank you, it has been a pleasure to work with this thesis.

Furthermore, I wish to thank Professor Carsten Thomassen for his explanations of NP-complete problems.

Lyngby, Denmark
September 1, 2005

Marie-Louise Højlund Rasmussen

Definitions

Word	***Definition***
active arc	arc with the entire quantity of a commodity as flow
arc	ordered pair of nodes (i, j)
arc capacity	total amount of flow that can be assigned to an arc
arc cost	price it costs to transport a commodity of quantity 1 along an arc
arc disjoint connecting paths	connecting paths with no common arcs
capacity function	function that assigns an arc capacity to each arc
connecting path	directed path in which the initial node is the source and the final node is the terminal
connecting path length	total cost of all arcs along a connecting path
connections	specifies which sub-sources are connected to which sub-terminals
demand	nonnegative value assigned to the terminal
directed graph	a node set and an arc set on these nodes
directed network	directed graph together with a capacity function
directed path	sequence $(i_1, i_2, \ldots, i_n)$ of nodes such that $(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n)$ are arcs of a directed graph
edge	unordered pair of nodes $\{i, j\}$
Eulerian directed graph	directed graph in which every node has equal in-degree and out-degree
feasible network flow	a flow in a directed network that satisfies flow conservation and does not exceed any arc capacity
fixed-charge network flow problem	the problem of minimizing the total cost of some active arcs while maintaining a feasible network flow
flow	nonnegative value assigned to an arc
flow balance	when the sum of flows along all inward directed arcs of a node equals the sum of flows along all outward directed arcs of that node

flow conservation	when there is flow balance at all nodes except at the source and the terminal, and the total amount of flow assigned to the outward- and inward directed arcs of the source and terminal, respectively, equals the demand
forbidden area	specifies which nodes and arcs some commodities may not have, respectively, a flow through or along
inactive arc	arc that is not active
in-degree	number of inward directed arcs at a given node in a directed graph
inward adjacent node	node that reaches a specific node by any inward directed arc of that node
inward directed arc	arc in which a specific node constitutes the second node in the ordered pair of nodes
mandatory area	specifies which nodes and arcs some commodities must have, respectively, a flow through or along
node	point or vertex
node disjoint connecting paths	connecting paths with no common nodes except for the source, terminal, and possibly sub-sources and sub-terminals
out-degree	number of outward directed arcs at a given node in a directed graph
outward adjacent node	node that is reached by any outward directed arc from a specific node
outward directed arc	arc in which a specific node constitutes the first node in the ordered pair of nodes
planar graph	undirected graph that can be drawn in a plane so that no edges cross each other
quantity	amount of a commodity
source	node with no inward adjacent nodes
sub-source	node with no inward adjacent nodes except for the source
sub-terminal	node with no outward adjacent nodes except for the terminal
terminal	node with no outward adjacent nodes
total cost	combined arc costs of some arcs
undirected graph	a node set and an edge set on these nodes

Symbol	***Definition***
$a_{i,j}$	constraint coefficient of constraint i for integer variable x_j
A	arc set
b_i	right-hand side coefficient of constraint i
C	set of linear constraints

c_i	objective function coefficient for integer variable x_i
$c_{i,j}$	cost of arc (i,j)
d	capacity function
$d_{i,j}$	capacity of arc (i,j)
D	number of smaller sub-programs in decomposition
δ	sub-tour
Δ	set of all sub-tours
E	edge set
f	linear objective function
$\overrightarrow{F}^k$	set of forbidden arcs of commodity k
$\dot{F}^k$	set of forbidden nodes of commodity k
$g_{i,j}$	constraint coefficient of constraint i for real variable y_j
$G=(V,A,d)$	directed network
h_i	objective function coefficient for real variable y_i
i or j	node
I	number of sub-sources
$I(i)$	set of inward adjacent nodes of node i
(i,j)	arc
$\{i,j\}$	edge
J	number of sub-terminals
k	type of commodity
K	number of commodities
L	connecting path length when equal for all commodities
L^k	connecting path length of commodity k
m	number of constraints in a general MILP
$\overrightarrow{M}^k$	set of mandatory arcs of commodity k
$\dot{M}^k$	set of mandatory nodes of commodity k
n	number of integer variables in a general MILP
N	set of actively named commodities
$N_{i,j}$	set of actively named commodities along arc (i,j)
$\overline{N}$	set of inactively named commodities
$\overline{N}_{i,j}$	set of inactively named commodities along arc (i,j)
$O(i)$	set of outward adjacent nodes of node i
p	number of real variables in a general MILP
q^k	quantity of commodity k
$\mathbb{R}^p_+$	set of p-dimensional nonnegative real vectors
ρ	percent of the average connecting path length
S	source
s_i	sub-source i
T	terminal
t_j	sub-terminal j
V	node set
x	nonnegative integer variables

y	nonnegative real variables
$\mathbb{Z}_+^n$	set of n-dimensional nonnegative integer vectors

Contents

Chapter 1

Introduction

Imagine a scenario in which an enterprise must deliver some goods to a customer. The enterprise has a chart that illustrates cities linked together by roads. The goods have a certain size and the roads have a possible load that cannot be exceeded. The transportation of goods must follow these roads. The customer demands that no two goods pass through the same location, perhaps due to security reasons.[1] There must, hence, be as many transportation routes as goods, and no two routes may intersect. The roads also take time to travel, and cost a certain amount. For this specific problem, referred to as the *time problem*, the enterprise wishes the transport to be as cheap as possible. However, the customer demands that the enterprise must deliver the goods within a certain time, so no route may take longer than that time. Furthermore, the customer expects goods to arrive at approximately the same time.

An analog of this problem is often seen in telecommunication. Here the roads correspond to cables and the cities to juncture points. In telecommunication, a data signal is split into several small packages of signals. This reduces the risk of transmission failures. No signal may exceed the possible load capacity of a cable. Furthermore, all signals must arrive at approximately equal times so as not to cause a too large time-delay.

Imagine another scenario with the same enterprise as in the time problem. In this scenario, a customer needs products distributed to its factories. The customer needs as many products as possible under the restriction that these products may not pass through the same location on their way to the factories.[2] Furthermore, the enterprise must again deliver the goods within a certain time, so no route may take longer than that time. This problem is referred to as the *distribution problem*.

The time- and distribution problem belong to a certain group of problems.

[1] It might be dangerous to keep two pieces of goods at the same location. Another reason might be that the customer wants independent transportation routes. This would result in a minimization of the importance of a failure of one transportation link.

[2] This could be for secrecy reasons. For example, it might be possible to determine the end product by seeing more than one component.

These are fixed-charge network flow problems in directed networks. However, the problems in this thesis are special types of fixed-charge network flow problems since they also include finding routes that do not intersect and that also satisfy other criteria. The main part of this thesis is to model these problems. This thesis develops a fairly general mathematical model of these problems in which a multitude of criteria can be incorporated, and which is suitable for computations.

Furthermore, this thesis gives algorithms to solve different challenges that the modeling gives rise to. These are how to eliminate sub-tours and how to decompose the problem into smaller sub-problems. Finally, this thesis implements and experiments upon the modeling and the developed algorithms.

In the following section, the notation that is used is introduced, and the problem is stated using this notation. The last section is about what has already been done within the field of the problem.

1.1 The Problem

This thesis deals with certain types of problems that can be modeled as fixed-charge network flow problems. This section begins by defining such problems and then states the problem that this thesis solves.

A *node* is a point or vertex, and an *arc* is an ordered pair of nodes. A node is usually denoted i or j, and an arc from node i to node j is denoted (i, j). In this thesis, an arrow visualizes an arc in the way seen in Figure 1.1. Here, node i and node j are connected by the arc (i, j).

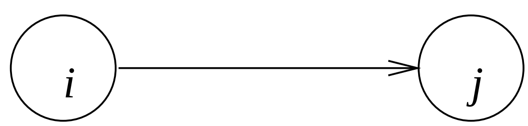

Figure 1.1: Two nodes i and j connected by the arc (i, j).

An arrow pointing both ways between node i and node j is a visualization of the two arcs (i, j) and (j, i).

A *node set* V is a set of nodes and an *arc set* A is a set of arcs. A *directed graph* (V, A) consists of a node set V and an arc set A on these nodes.

Arcs are both inward- and outward directed. *Inward directed* arcs of node j are all arcs of the form (i, j). Similarly, the *outward directed* arcs of node i are all arcs of the form (i, j). Note that arc (i, j) is an outward directed arc of node i and an inward directed arc of node j.

An *outward adjacent node* of node i is a node j for which (i, j) is an arc. Similarly, an *inward adjacent node* of node j is a node i for which (i, j) is an arc. In Figure 1.1, node j is an outward adjacent node of node i, and node i is an inward adjacent node of node j. A *source* is a node with no inward adjacent nodes. A *terminal* is a node with no outward adjacent nodes. The directed networks in this thesis have only one source node and one terminal node, denoted S and T, respectively. Note that these nodes may be fictive. That is, they might not represent a location like the other nodes in the directed network but are introduced to model the problem.

A *directed path* in a directed graph is a sequence $(i_1, i_2, \ldots, i_n)$ of nodes such that $(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n)$ are arcs of the directed graph. A *connecting path* is a directed path in which the initial node is the source and the final node is the terminal. In this thesis, each connecting path models the transportation of a commodity.

The *flow* of an arc is a nonnegative value assigned to this arc. The *demand* of a graph is a nonnegative value assigned to the terminal T. For the demand to be satisfied, the total amount of flow assigned to the inward directed arcs of the terminal node must be greater than or equal to the demand. *Flow balance* at node i is the state in which the sum of flows along all inward directed arcs of node i equals the sum of flows along all outward directed arcs of node i. *Flow conservation* is the state in which there is flow balance at all nodes except at the source S and the terminal T, and the total amount of flow assigned to the outward- and inward directed arcs of the source and terminal, respectively, equals the demand of the graph.

The *arc capacity* of an arc is the total amount of flow that can be assigned to that arc. A *capacity function* d assigns an arc capacity to each arc.

This leads to the definition of a *directed network*. Such network is a directed graph together with a capacity function, and is denoted $G = (V, A, d)$.

Seen in Figure 1.2 is an example of a directed network $G = (V, A, d)$ with the node set

$$V = \{1, 2, 3, 4, S, T\} ,$$

in which S is the source node and T is the terminal node, and the arc set

$$A = \{(1, 2), (3, 4), (1, 3), (4, 2), (1, 4), (S, 1), (S, 3), (2, T), (4, T)\} .$$

The arc capacities are written next to the arcs; they are

$$d = (2, 4, 1, 1, 2, 3, 4, 2, 1) .$$

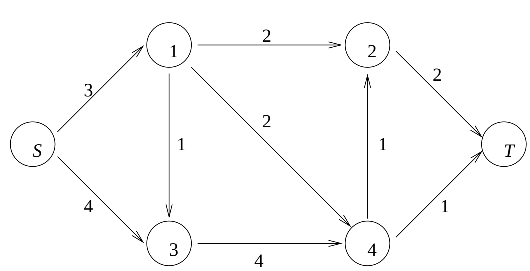

Figure 1.2: A directed network with 6 nodes and 9 arcs.

A given flow is a *feasible network flow* when
- no flow exceeds the corresponding arc capacity, and
- flow conservation is preserved.

In this thesis, we wish to model a transportation of commodities from the source node S to the terminal node T. Each commodity has a *quantity* that is the amount of that commodity. The *cost* of an arc is the price it costs to transport a commodity of quantity 1 along that arc. The *total cost* of some arcs is the combined arc costs of these arcs.

An arc is an *active arc* if it has the entire quantity of a commodity as flow, and an *inactive arc* otherwise. This may be thought of as transportation of commodities along an arc. Hence, there is no transportation along an arc when it is inactive, and there is a transportation when the arc is active.

The *fixed-charge network flow problem* is to minimize the total cost of some active arcs while maintaining a feasible network flow.

The fixed-charge network flow problem does not model routes that do not intersect. The definitions are therefore extended as in the following. *Arc disjoint connecting paths* are connecting paths with no common arcs. *Node disjoint connecting paths* are connecting paths with no common nodes except for the source S and terminal T. Node disjoint connecting paths are arc disjoint but the converse is not true. An example of two arc disjoint connecting paths is seen to the left in Figure 1.3. They are not node disjoint since they share node 4. Two node disjoint paths are seen to the right in Figure 1.3. The connecting paths are the arcs in bold; one is gray, and the other black.

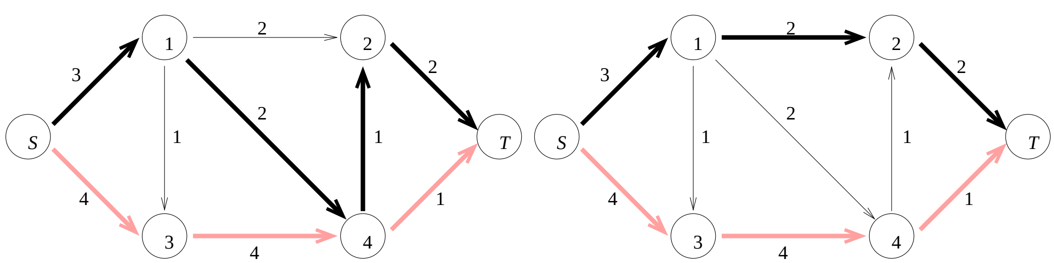

Figure 1.3: Arc- (left) and node disjoint connecting paths (right).

The *connecting path length* is the total cost of all arcs along this connecting path. Length criteria define what these connecting path lengths must be.

A *sub-source* is a node with no inward adjacent nodes except for the source; similarly, a *sub-terminal* is a node with no outward adjacent nodes except for the terminal. Sub-sources and sub-terminals in a directed network may be related to one-another. The *connections* of a directed network specifies which sub-sources are connected to which sub-terminals via some connecting paths. Furthermore, there may be *forbidden-* and *mandatory areas* in a directed network. These specify whether there are some commodities that may not or must have a flow through some nodes or along some arcs. Distribution criteria define the connections and the forbidden- and mandatory areas in a directed network.

It is now possible to state the problem of this thesis. It is to find either a specific demand or a maximal number of connecting paths in a directed network such that the connecting paths satisfy a feasible network flow and are either arc- or node disjoint. Furthermore, different length- and distribution criteria may be imposed on the connecting paths.

1.2 Background

The problem of finding arc- or node disjoint connecting paths has a theoretical and practical history, and has been treated in many articles. This is because it occurs in areas such as telecommunication, transportation, and production. In these fields, it is of great interest to have more than one connecting path. However, the literature typically does not treat length- and distribution criteria.

To explain why it is not possible to use much of this existing literature to solve the thesis problem, we need some more definitions. An *edge* is an unordered pair of nodes $\{i, j\}$. An *undirected graph* (V, E) consists of a node set V and an edge set E on the nodes V.

Networks, paths, and edge- and node disjointness may each be defined as in the directed case. Notice that an undirected network can be thought of as a special type of directed network: the directed network has two arcs, (i, j) and (j, i), for each edge $\{i, j\}$ in the undirected network.

In the existing literature, a common problem is to find a set of edge- or node disjoint connecting paths in undirected network, see e.g. [4, 5, 16]. This theory is not used in this thesis since a larger class of networks are treated here.

However, there are also many articles that treat problems concerning arc- and node disjoint connecting paths in directed networks. An early article on this subject is [12] from 1974; an improved version is [13] from 1984. These articles propose an algorithm for finding the K node disjoint connecting paths with minimal combined connecting path lengths. Several other papers have treated this problem, see e.g. [11, 15]. However, the proposed algorithms give the combined connecting path lengths and not the individual connecting path lengths.

Article [4, Section 8] proposes a way to solve the problem of finding edge disjoint connecting paths in undirected networks. This is to write the problem as a special type of mixed-integer linear program, namely as an integer multicommodity flow. When modeling the problem in this way, it is possible to impose criteria on each individual undirected connecting path. Furthermore, the problem can be optimized by general optimization routines since it is a linear model. Integer multicommodity flow is also applied to directed network, see [2].

For these reasons integer multicommodity flow, which we present in the following chapter, has been chosen as a template for the models in this thesis.

Chapter 2

Theory

2.1 Introduction

This chapter shows how to translate the thesis problem into a mathematical model suitable for the application of computational methods. It presents some general theory and demonstrates how this theory can be expanded to solve the problem of finding arc- or node disjoint connecting paths. The general theory uses mixed-integer linear programming and integer multicommodity flow. This theory is our point of departure and is enhanced in order to solve the problem of finding arc- or node disjoint connecting paths with length- and distribution criteria. One extension we make is naming, which enables criteria to be modeled by extra constraints to the integer multicommodity flow program. Furthermore, we see the influences of having different objective functions. Finally, we discuss the challenges that can arise: sub-tours in solutions and NP-complete large-scale problems.

2.2 Mixed-Integer Linear Programming

The point of departure to model the problems of this thesis is to model them as proposed in [4, Section 8]. The model proposed in [4, Section 8] is a special type of *mixed-integer linear program*, denoted MILP. This section presents MILP. Further discussions are seen in [9, Chapter I.5] and [10, Chapter 13].

A MILP (f, C) is a mathematical model composed of a linear objective function f and a set of linear constraints C. Both the objective function and the set of constraints contain two types of variables, namely nonnegative integers and nonnegative reals. Throughout the following, f is linear in its variables and the constraints C are linear inequalities. If there are n nonnegative integer variables $x_1, \ldots, x_n$ and p nonnegative real variables $y_1, \ldots, y_p$, then set $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_p)$. Note that

$$x \in \mathbb{Z}_+^n \quad \text{and} \quad y \in \mathbb{R}_+^p \ ,$$

where $\mathbb{Z}_+^n$ denotes the set of n-dimensional nonnegative integer vectors, and $\mathbb{R}_+^p$ denotes the set of p-dimensional nonnegative real vectors. In a MILP, there must always be at least one variable x_i or y_i present, so $n+p \geq 1$. If there only are integer variables present in the modeling (i.e., $p = 0$), then this problem is a *pure-integer program*. When only real variables are present (i.e., $n = 0$), the problem is a *linear program*. In this thesis, the problems are modeled as both pure-integer programs and mixed-integer programs.

The linear objective function f may be any linear combination of the integer variables x and the real variables y. That is, f can be written as

$$\sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{p} h_i y_i \ ,$$

for real coefficients c_i and h_i; the *objective function coefficients* for integer variable x_i and real variable y_i, respectively.

The set of linear constraints C can be written as

$$\sum_{j=1}^{n} a_{i,j} x_j + \sum_{j=1}^{p} g_{i,j} y_j \leq b_i \ , \quad \forall i \in \{1, \ldots, m\} \ ,$$

for real coefficients $a_{i,j}$ and $g_{i,j}$; the *constraint coefficients* of constraint i for integer variable x_j and real variable y_j, respectively. Furthermore, the real coefficients b_i constitute the *right-hand side* of the constraints.

In a MILP, the objective function is optimized by either being maximized or minimized.[1] A general MILP can be written in the form

$$\max / \min \quad \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{p} h_i y_i \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{i,j} x_j + \sum_{j=1}^{p} g_{i,j} y_j \leq b_i \ , \quad \forall i \in \{1, \ldots, m\} \ , \tag{2.2}$$

$$x \in \mathbb{Z}_+^n \ , \text{ and } \quad y \in \mathbb{R}_+^p \ . \tag{2.3}$$

The objective of the MILP is expressed in (2.1); the set of linear constraints C is presented in (2.2); and the variables are seen in (2.3).

The *set of feasible solutions* to the MILP (2.1)–(2.3) are defined as those variables $x \in \mathbb{Z}_+^n$ and $y \in \mathbb{R}_+^p$ satisfying all constraints (2.2). A *feasible (infeasible) program* is a program for which the set of feasible solutions is not empty (is empty). An *optimal solution* is a feasible solution which minimizes or maximizes the objective function f from (2.1). For all feasible programs, there is at least one optimal solution.[2]

[1] The maximum of an objective function f can also be found by minimizing $-f$.

[2] There exist only a finite number of feasible solutions, so there must be an optimal solution among them.

2.3 Integer Multicommodity Flow

An integer multicommodity flow is a special type of mixed-integer linear program. It can be used to model the problem in which commodities of certain quantities must share arcs with arc capacities. This section presents the integer multicommodity flow model, as presented in [2]. For further details see also [1] and [4, pp. 159–177]. In the following, integer multicommodity flow is denoted IMCF. Throughout this section, parallels are drawn to the time problem in Chapter 1 when describing the model.

IMCF represents a directed network $G = (V, A, d)$ consisting of a node set V, an arc set A, and a capacity function d. Returning to the enterprise that has a chart with roads linking cities together, the chart corresponds to the directed network; the cities correspond to the node set; the connections between the cities are described by the arc set; and the maximal load of transport along each road corresponds to the capacity function.

The enterprise wants to transport a certain amount of goods to the customer. In IMCF, this is represented by K commodities $k \in \{1, \ldots, K\}$. The goods each have a size which corresponds to commodity k having a quantity q^k.

In the example, the roads take a certain time to travel. This is equivalent to the arcs having a cost $c_{i,j}$. Furthermore, the roads have a maximal transport load that cannot be exceeded. This is represented by the arc capacities $d_{i,j}$.

The above described parameters of IMCF are listed in Table 2.1.

Parameters	***Explanation***
A	arc set
$c_{i,j}$	cost of arc (i, j)
$d_{i,j}$	capacity of arc (i, j)
K	number of commodities
V	node set
q^k	quantity of commodity k

Table 2.1: Parameters of IMCF.

In IMCF, the variables $x^k_{i,j}$ describe the flow of each commodity k along each arc (i, j). This is obtained by representing each arc K times. Hence, we have K variables per arc describing the flow along that arc. Figure 2.1 illustrates arc (i, j) represented K times.

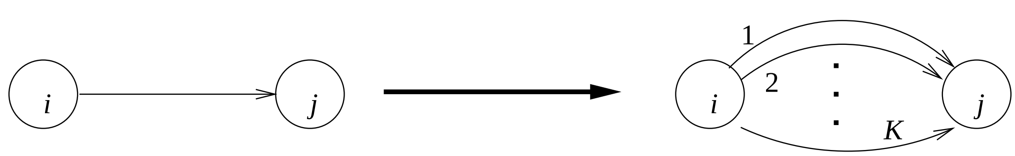

Figure 2.1: Arc (i, j) is represented K times.

The variable $x^k_{i,j}$ is defined as follows:

$$x^k_{i,j} = \begin{cases} 1 \text{ if the entire quantity } q^k \text{ of commodity } k \text{ is assigned to arc } (i,j); \\ 0 \text{ otherwise.} \end{cases}$$

Note that this representation produces large-scale programs, since the number of variables is K larger than the number of arcs in the directed network. For now, we do not consider the complexities of this mathematical model. However, we develop an algorithm in Chapter 4 that compensates for this representation. What is more, we see in Chapter 5 that this representation produces sparse matrices, which also compensates for this representation.

With this representation, it is possible to model feasible flows in the directed network $G = (V, A, d)$. The following constraints must apply:

1. flow conservation,
2. arc capacity limitations, and
3. non-negativity of flows.

Constraint 1 preserves flow balance in all nodes except for the source S and terminal T. We denote the set of outward adjacent nodes $O(i)$, and the set of inward adjacent nodes $I(i)$. In IMCF, flow balance is then written as

$$\sum_{j \in O(i)} x^k_{i,j} - \sum_{j \in I(i)} x^k_{j,i} = 0 \ , \quad \forall i \in V \backslash \{S, T\}, \ k \in \{1, \ldots, K\} \ .$$

However, additional constraints are required to ensure positive flows along the outward- and inward directed arcs of the source and terminal, respectively. This is done by introducing the parameter

$$b^k_i = \begin{cases} 1 & \text{if } i = S; \\ -1 & \text{if } i = T; \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Now the set of constraints that imposes conservation of flow can be written as

$$\sum_{j \in O(i)} x^k_{i,j} - \sum_{j \in I(i)} x^k_{j,i} = b^k_i \ , \quad \forall i \in V, \ k \in \{1, \ldots, K\} \ .$$

Constraint 2 imposes that the commodities k of quantities q^k transported along arc (i, j) do not exceed the arc capacity $d_{i,j}$. This is expressed in IMCF by the inequalities

$$\sum_{k=1}^{K} q^k x^k_{i,j} \leq d_{i,j} \ , \quad \forall (i,j) \in A \ .$$

The variables are defined as $x^k_{i,j} \in \{0, 1\}$, $\forall (i,j) \in A, \ k \in \{1, \ldots, K\}$, hence, constraint 3 is automatically satisfied.

All in all, the following sets of constraints are given:

$$\begin{aligned} \sum_{k=1}^{K} q^k x_{i,j}^k \leq d_{i,j} \ , \qquad & \forall (i,j) \in A \ , \\ \sum_{j \in O(i)} x_{i,j}^k - \sum_{j \in I(i)} x_{j,i}^k = b_i^k \ , \qquad & \forall i \in V, k \in \{1, \ldots, K\} \ , \\ x_{i,j}^k \in \{0,1\} \ , \qquad & \forall (i,j) \in A, k \in \{1, \ldots, K\} \ , \end{aligned} \tag{2.5}$$

where the parameters are as specified in (2.4) and Table 2.1. Throughout this thesis, the sets of constraints (2.5) are known as the *IMCF network model*.

The IMCF network model (2.5) is a feasibility model since with this we are only searching a feasible solution. What is more, also the disjointness of the connecting paths and the length- and distribution criteria can be represented by constraints. The following sections treat therefore only constraints, whereas we treat objective functions afterwards.

2.4 Disjoint Connecting Paths

This section discusses how to model the problem of finding arc- or node disjoint connecting paths in a directed network. This is relevant when no more than one commodity may have a flow along an arc or through a node.

When we model directed networks with sub-sources and/or sub-terminals, then the arc- and node disjointness criteria are somewhat altered. In these cases, arc disjointness criteria do not apply to inward- and outward directed arcs of, respectively, sub-sources and sub-terminals. Similarly, node disjointness does not apply to sub-sources and sub-terminals. This, however, complicates notation, so, in this section, we choose to consider directed networks without sub-sources and sub-terminals. Later in the chapter, we see a case in which we include sub-terminals.

2.4.1 Arc Disjoint Connecting Paths

The IMCF network model is extended, to model arc disjoint connecting paths, by adding the constraint that no more than one commodity may use an arc. In other words,

$$\sum_{k=1}^{K} x_{i,j}^k \leq 1 \ , \quad \forall (i,j) \in A \ . \tag{2.6}$$

2.4.2 Node Disjoint Connecting Paths

It is also possible to extend the IMCF network model to find node disjoint connecting paths. However, the extra constraint has to concern the arcs, since it is the arcs that are represented by the variables in the IMCF network model.

There are two ways of modeling how to find node disjoint connecting paths: representing a node twice and imposing arc disjointness or extending the IMCF network model with another set of constraints. If we would choose to represent every node twice, then the first should contain the inward directed arcs and the second the outward directed arcs. Hereafter, an extra arc is introduced connecting the two representations. See Figure 2.2 for an illustration.

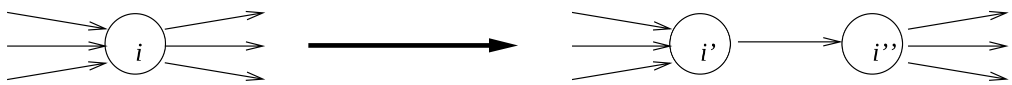

Figure 2.2: One node is represented twice and an extra arc is introduced.

This is done for all nodes except for the source and terminal. It would entail a larger model (an arc per node), so we recommend the following approach instead. The IMCF network model (2.5) is extended with a set of constraints ensuring that no more than one commodity has a flow along the inward- or outward directed arcs of a node except for the source and terminal. These constraints may be expressed as follows:

$$\sum_{k=1}^{K} \sum_{i \in I(j)} x_{i,j}^{k} \leq 1 \ , \qquad \forall j \in V \backslash \{S, T\} \ , \quad \text{and} \tag{2.7}$$

$$\sum_{k=1}^{K} \sum_{j \in O(i)} x_{i,j}^{k} \leq 1 \ , \qquad \forall i \in V \backslash \{S, T\} \ . \tag{2.8}$$

Note that (2.7) (corresponding to inward directed arcs) implies (2.8) (corresponding to outward directed arcs) and vice versa due to flow balance. In this thesis, we model the problem of finding node disjoint connecting paths in a directed network by imposing (2.8).

2.5 Naming

In the IMCF network model it is implicitly assumed that the directed network has a demand which is the same as the number of commodities. However, the directed network might not have this demand; for instance, the customer in the distribution problem in Chapter 1 wishes an unspecified maximal number of goods. Furthermore, the criteria that are imposed on the connecting paths might not apply to all connecting paths. This entails a need for two certain types of *naming* of the connecting paths.

Active naming entails that a certain commodity must have a positive flow along a certain arc. The set of commodities that are actively named along arc (i, j) are denoted $N_{i,j}$. This is modeled by setting

$$x_{i,j}^{k} = 1 \ , \quad \forall (i, j) \in A, \ k \in N_{i,j} \ . \tag{2.9}$$

Inactive naming forces some commodities to be inactive on specific arcs. However, it does not decide which commodities, if any, are active. The set of commodities that are forced to be inactive on arc (i, j) is denoted $\overline{N}_{i,j}$. Hence, inactively naming the commodities $k \in \overline{N}_{i,j}$ to not have a flow along arc (i, j) is imposed by setting

$$x_{i,j}^k = 0 \ , \quad \forall (i, j) \in A, \ k \in \overline{N}_{i,j} \ . \tag{2.10}$$

Note that active naming combined with a disjointness criterion imply that the other commodities are inactively named along that arc.

In the IMCF network model, there must be as many connecting paths as commodities. This is modeled from the flow conservation constraints, which forces positive flow along the outward- and inward directed arcs of the source and terminal, respectively. That is, the flow conservation constraints at the source S and at the terminal T are, respectively,

$$\begin{aligned} \sum_{j \in O(S)} x_{S,j}^k = 1 \ , \qquad \forall k \in \{1, \ldots, K\} \quad \text{and} \\ \sum_{j \in I(T)} x_{j,T}^k = 1 \ , \qquad \forall k \in \{1, \ldots, K\} \ . \end{aligned} \tag{2.11}$$

To avoid forcing a solution to have K connecting paths, the flow conservation constraints are first excluded, and flow balance is introduced in the model. Flow balance must be satisfied at all nodes except for the source and the terminal:

$$\sum_{j \in O(i)} x_{i,j}^k - \sum_{j \in I(i)} x_{j,i}^k = 0 \ , \quad \forall i \in V \backslash \{S, T\}, k \in \{1, \ldots, K\} \ .$$

However, there may be some commodities that must have corresponding connecting paths. This is modeled by using active naming along the outward directed arcs of the source.[3] The set of commodities that are actively named is denoted N, which is the union of all subsets $N_{i,j}$. We force all actively named commodities to have a flow from the source by the set of constraints:

$$\sum_{j \in O(S)} x_{S,j}^k = 1 \ , \quad \forall k \in N \ . \tag{2.12}$$

Not only does (2.12) imply that there must be a connecting path for each commodity $k \in N$, but it ensures that there is precisely one such path. This is important when length- and distribution criteria are imposed on the connecting paths.

[3]Forcing flow from the source together with flow balance imposes flow to the terminal even though (2.11) is left out.

Using active naming, we modify the IMCF network model to what we call the *named IMCF model*:

$$
\begin{aligned}
\sum_{k=1}^{K} q^k x_{i,j}^k \leq d_{i,j} \ , &\qquad \forall (i,j) \in A \ , \\
\sum_{j \in O(i)} x_{i,j}^k - \sum_{j \in I(i)} x_{j,i}^k = 0 \ , &\qquad \forall i \in V \backslash \{S, T\}, k \in \{1, \ldots, K\} \ , \\
\sum_{j \in O(S)} x_{S,j}^k = 1 \ , &\qquad \forall k \in N \ , \\
x_{i,j}^k \in \{0,1\} \ , &\qquad \forall (i,j) \in A, k \in \{1, \ldots, K\} \ . \qquad (2.13)
\end{aligned}
$$

The (non named) IMCF network model can be used when we know that the demand of the directed network is the same as the number of commodities. In the following, however, this may not be true, and therefore we use the named IMCF model. To the best of our knowledge, the named IMCF model is new for this thesis, and will prove substantial to an algorithm presented in Chapter 4. Furthermore, notice that when the objective is to minimize, then using this approach might result in a solution only having the forced connecting paths. Hence, the objective should also reflect the choice of model. This is something we return to in Section 2.8.

2.6 Length Criteria

In the time- and distribution problem given in Chapter 1, the customer demands that the enterprise must deliver the goods within a certain time, so no route may take longer than that time. This corresponds to a length criterion imposed on the connecting paths in the directed network. By adding sets of constraints, we extend the named IMCF model (2.13) in order to include different length criteria.

The connecting path length is the total cost of all arcs in a connecting path with cost $c_{i,j}$ of arc (i,j). That is, the connecting path length of commodity k is:

$$\sum_{(i,j) \in A} c_{i,j} x_{i,j}^k \ .$$

This section gives three examples of length criteria. These ensure that connecting paths have either

1. specified lengths,
2. equal length, or
3. approximately equal lengths.

2.6.1 Specified Lengths

The named IMCF model can be extended to encompass the cases in which goods must be delivered before, at, or after a specific time, during a specified time period, or a mixture of all of these specified time/length cases.

By adding a set of constraints, we extend the named IMCF model to ensure specified lengths of given connecting paths N_{sl}^e. These constraints limit the lengths of each connecting path to equal the specified lengths L^k:

$$\sum_{(i,j)\in A} c_{i,j} x_{i,j}^k \;=\; L^k \;, \quad \forall k \in N_{sl}^e \;.$$

In the example from Chapter 1, the goods should be delivered before a specific delivery time. This is modeled by extending the named IMCF in a slightly different manner:

$$\sum_{(i,j)\in A} c_{i,j} x_{i,j}^k \;\leq\; L^k \;, \quad \forall k \in N_{sl}^l \;,$$

in which N_{sl}^l is the set of commodities with the less than length criteria L^k. The same modeling is found in [3], in which the problem is to find node disjoint paths for the movement of train cars under the restriction that the train cars must arrive before certain time limits.

It is also possible to ensure connecting paths N_{sl}^g of at least certain lengths L^k:

$$\sum_{(i,j)\in A} c_{i,j} x_{i,j}^k \;\geq\; L^k \;, \quad \forall k \in N_{sl}^g \;.$$

If there is no difference between the specification of above lengths L^k, then the constants L^k are substituted by a single constant L in the formulas.

2.6.2 Equal Length

The named IMCF model can be extended to include the length criterion ensuring that the connecting paths corresponding to commodities $k \in N_{el}$ have equal lengths. With this criterion, it is not specified what length the connecting paths should have, only that they must be equal. This gives the set of constraints

$$\sum_{(i,j)\in A} c_{i,j} x_{i,j}^k - \sum_{(i,j)\in A} c_{i,j} x_{i,j}^{k'} \;=\; 0 \;, \quad \forall k, k' \in N_{el} \;.$$

2.6.3 Approximately Equal Lengths

In the time problem in Chapter 1, the customer expected the goods to arrive at approximately equal times. This is another type of length criterion that can be incorporated in an extended version of the named IMCF model. The extension transforms the named IMCF model from a pure-integer program to a mixed-integer program.

That the connecting paths should be of approximately equal lengths means that no connecting path length may be very different from the average connecting path length. In the following, we treat the case in which this average connecting path length is unknown. A final remark is given of how this average connecting path length also can be specified.

When the average connecting path length is unknown, it can be defined as the average over the actively named commodities N_{ael}; namely, the total cost of all connecting path lengths with actively named commodities divided by the number of these paths. This is modeled by introducing a continuous variable y that is to be this average and, hence, satisfy the constraint

$$\sum_{k \in N_{ael}} \sum_{(i,j) \in A} c_{i,j} x_{i,j}^k - |N_{ael}| y = 0 \ , \tag{2.14}$$

where $|N_{ael}|$ is the number of commodities with corresponding connecting paths ensuring approximately equal lengths.

Another average is the total cost of all connecting path lengths divided by the number of connecting paths K. Now, the average y must satisfy the constraint

$$\sum_{k=1}^{K} \sum_{(i,j) \in A} c_{i,j} x_{i,j}^k - K y = 0 \ . \tag{2.15}$$

Knowing the average, it is now possible to constrain the connecting paths to be of lengths within some percentage from each other. The connecting path length of connecting path corresponding to a commodity k is within ρ percent of the average length y if it satisfies

$$(1 - \rho) y \leq \sum_{(i,j) \in A} c_{i,j} x_{i,j}^k \leq (1 + \rho) y \ .$$

Hence, the requirement of connecting paths having approximately equal lengths is modeled by including the sets of constraints

$$\begin{aligned} \sum_{(i,j) \in A} c_{i,j} x_{i,j}^k - (1 + \rho) y \ &\leq \ 0 \ , \quad \forall k \in N_{ael} \\ \sum_{(i,j) \in A} c_{i,j} x_{i,j}^k - (1 - \rho) y \ &\geq \ 0 \ , \quad \forall k \in N_{ael} \ , \end{aligned} \tag{2.16}$$

together with one of the average constraints (2.14) or (2.15).

In the above model, the average y depends on the variables x. However, the average may also be given as input, by specifying the value of y in (2.16) and ignoring criteria (2.14) and (2.15). The connecting path lengths are then within ρ percent of this given value.

2.7 Distribution Criteria

In the distribution problem in Chapter 1, the enterprise has to deliver goods from a producer to factories in different cities. This is a connection distribution criterion.

This section treats the criteria that concern distribution. Distribution criteria arise in problems in which there are differences in the commodities. These could be the origin or destination of the commodity, and also whether a commodity must pass through or is not allowed to pass through some areas. The first type is a connection distribution criterion. The other two types are, respectively, mandatory- and forbidden area distribution criteria. In the following, we describe and model these criteria.

2.7.1 Connections

A way to model connection distribution criteria is proposed in [1, Chapter 2, Section 5]. The IMCF network flow model does then not impose flow balance in all nodes except for the source and terminal, but instead it introduces the vector

$$b_i^k = \begin{cases} 1 & \text{if } i \text{ is a sub-source;} \\ -1 & \text{if } i \text{ is a sub-terminal; and} \\ 0 & \text{otherwise.} \end{cases}$$

The model in [1, Chapter 2, Section 5] is then modeled as the IMCF network flow model with this different definition of the right-hand side in the conservation of flow constraints. This is not the model we choose, since it assumes that we know how much should be transported of each commodity. We propose a more general approach in this section which uses naming.

General Model

Connection distribution criteria are modeled by defining sub-sources and sub-terminals. This is to name the commodities. Throughout this thesis, sub-sources are denoted s_i and sub-terminals t_j, and I and J are the numbers of sub-sources and sub-terminals, respectively. Since the naming might differ for each sub-source and sub-terminal, we introduce the subsets N_{s_i}, $\overline{N}_{s_i}$, N_{t_j}, and $\overline{N}_{t_j}$. The commodities that must or may not leave sub-source s_i are included in N_{s_i} and $\overline{N}_{s_i}$, respectively. Similarly, N_{t_j} includes the commodities that must enter sub-terminal t_j, and $\overline{N}_{t_j}$ those that may not.

We actively and inactively name, respectively, all $k \in N_{s_i}$ and $k \in \overline{N}_{s_i}$ representations of the outward directed arcs of the source. Similarly, the representations of the arcs connecting the sub-terminals with the terminal are actively and inactively named for all $k \in N_{t_j}$ and $k \in \overline{N}_{t_j}$, respectively. The connection distribution criteria can then be written generally as

$$\begin{aligned}
x^k_{S,s_i} &= 1 \ , \qquad \forall i \in \{1,\ldots,I\},\ k \in N_{s_i} \\
x^k_{S,s_i} &= 0 \ , \qquad \forall i \in \{1,\ldots,I\},\ k \in \overline{N}_{s_i} \\
x^k_{t_j,T} &= 1 \ , \qquad \forall j \in \{1,\ldots,J\},\ k \in N_{t_j} \\
x^k_{t_j,T} &= 0 \ , \qquad \forall j \in \{1,\ldots,J\},\ k \in \overline{N}_{t_j} \ .
\end{aligned}$$

Example

In the distribution problem in Chapter 1, the customer wanted products from a producer delivered to some factories. Figure 2.3 is an illustration of a more general case with eight producers and eight factories represented by dots and double triangles, respectively. There are 8 types of commodities, in which we define that commodities 1–3 must have a corresponding connecting path. In this situation, there are some limitations to which commodity may enter which factory; all commodities may enter factories 1–6 but only commodities 4–8 may enter factories 7 and 8.

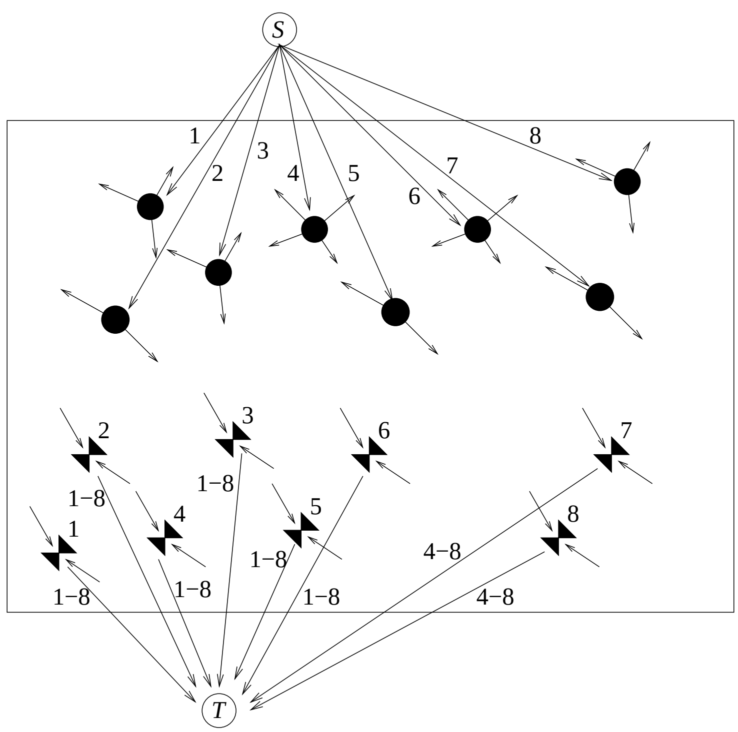

Figure 2.3: Connection distribution criterion.

We define the producers as sub-sources and the factories as sub-terminals, and extend the directed network by two nodes S and T and by the arcs from S and to T. The connection distribution criterion described in Figure 2.3 is modeled as illustrated in Figure 2.4. Here, the dark area symbolizes the part of the directed network that connects the sub-sources with the sub-terminals.

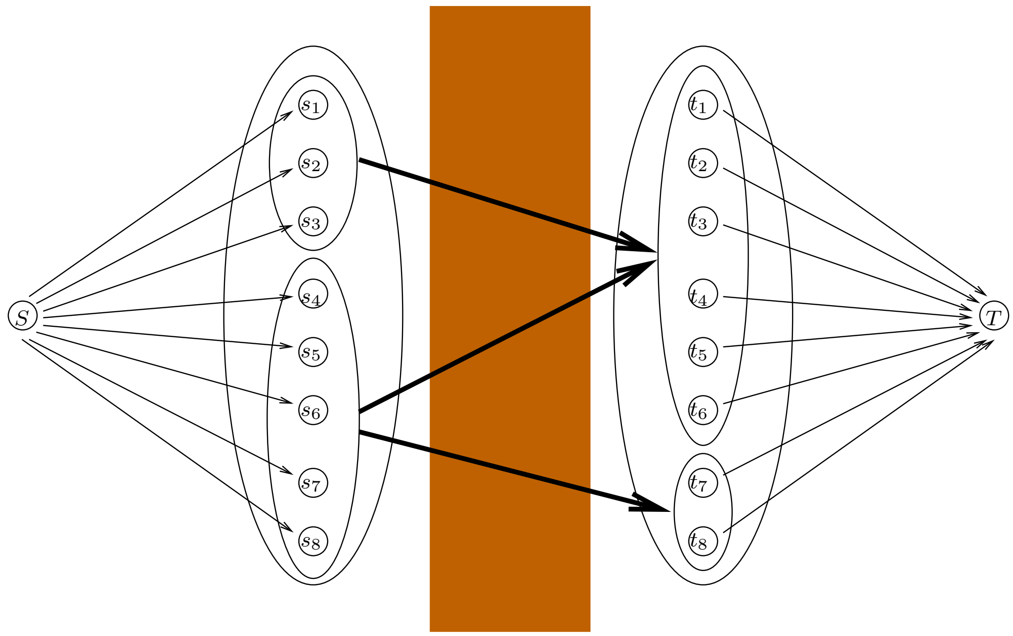

Figure 2.4: Modeling connection distribution criterion from Figure 2.3.

We impose inactive naming of all commodities on all outward directed arcs from the source except for the kth representation of the arcs. Furthermore, we impose active naming of commodities 1–3 along arcs (S, s_1), (S, s_2), and (S, s_3), respectively. Hence,

$$N_{s_i} = \begin{cases} \{i\}, & \text{if } i \in \{1,2,3\} \\ \{\}, & \text{otherwise} \end{cases} \text{ and } \overline{N}_{s_i} = \{1,\ldots,8\} \backslash \{i\} \; .$$

Furthermore, we define where the connecting paths corresponding to each commodity must terminate. Sub-terminals t_1–t_6 are not named since all types of commodities may have a positive flow to them. However, sub-terminals t_7 and t_8 may not receive connecting paths from sub-sources s_1–s_3; hence, t_7 and t_8 have an inactive naming of commodities 1–3. Expressed in the subsets N_{t_j} and $\overline{N}_{t_j}$, then all N_{t_j} are empty, and

$$\overline{N}_{t_j} = \begin{cases} \{\}, & \text{if } j \in \{1,\ldots,6\} \\ \{1,2,3\}, & \text{otherwise} \end{cases} \; .$$

The modeling using this approach is

$$
\begin{aligned}
x^k_{S,s_i} &= 1 \ , \qquad \forall i \in \{1,2,3\},\ k \in N_{s_i} \\
x^k_{S,s_i} &= 0 \ , \qquad \forall i \in \{1,\ldots,8\},\ k \in \overline{N}_{s_i} \\
x^k_{t_j,T} &= 0 \ , \qquad \forall j \in \{7,8\},\ k \in \overline{N}_{t_j} \ .
\end{aligned}
$$

2.7.2 Forbidden Areas

It may be that there are locations through which the goods may not pass, or roads on which the goods may not be transported. These correspond to *forbidden areas* in the directed network. The set of forbidden arcs for commodity k in the directed network is denoted $\overrightarrow{F}^k$, and is imposed by the set of constraints:

$$x^k_{i,j} = 0 \ , \quad \forall k \in \{1,\ldots,K\},\ (i,j) \in \overrightarrow{F}^k \ .$$

When a certain location is forbidden, it is modeled by ensuring that no commodity may enter or exit the corresponding node:

$$x^k_{i,j} = 0 \ , \qquad \forall k \in \{1,\ldots,K\},\ j \in \dot{F}^k,\ i \in I(j) \quad \text{and} \tag{2.17}$$

$$x^k_{i,j} = 0 \ , \qquad \forall k \in \{1,\ldots,K\},\ i \in \dot{F}^k,\ j \in O(i) \ , \tag{2.18}$$

where $\dot{F}^k$ is the collection of nodes that are forbidden for commodity k. Note that, due to flow balance, (2.17) and (2.18) imply one another.

2.7.3 Mandatory Areas

A *mandatory area* distribution criterion includes a directed network in which a commodity must have a flow along some arcs or through some nodes. The mandatory arcs of a certain commodity are modeled by making the corresponding variables active:

$$x^k_{i,j} = 1 \ , \quad \forall k \in \{1,\ldots,K\},\ (i,j) \in \overrightarrow{M}^k \ ,$$

where $\overrightarrow{M}^k$ is the collection of arcs that are mandatory for commodity k.

Each necessary location is modeled by ensuring that the commodity with the imposed criterion must have a flow along one inward- and one outward directed arc of the corresponding node:

$$\sum_{i \in I(j)} x^k_{i,j} = 1 \ , \qquad \forall k \in \{1,\ldots,K\},\ j \in \dot{M}^k \text{ and} \tag{2.19}$$

$$\sum_{j \in O(i)} x^k_{i,j} = 1 \ , \qquad \forall k \in \{1,\ldots,K\},\ i \in \dot{M}^k \ , \tag{2.20}$$

where $\dot{M}^k$ is the collection of nodes that are mandatory for commodity k. Here as above, (2.19) and (2.20) imply one another due to flow balance.

2.8 Combined Criteria

This section presents different objectives, and then gives examples of how we can combine objectives and criteria.

2.8.1 Different Objectives

There are two types of objectives. One is to minimize, say cost, and the other is to maximize, say profit. In the two problems in Chapter 1, i.e., the time- and distribution problem, there are two different objectives. In the following, these are given as examples.

When searching for the maximum number of node disjoint connecting paths, the objective is to maximize the number of such paths. This may be done in three different ways, namely to maximize one of the objective functions:

$$\sum_{k=1}^{K} \sum_{j \in O(S)} x_{S,j}^k \ , \tag{2.21}$$

$$\sum_{k=1}^{K} \sum_{j \in I(T)} x_{j,T}^k \ , \quad \text{or} \tag{2.22}$$

$$\sum_{k=1}^{K} \left(\sum_{j \in O(S)} x_{S,j}^k + \sum_{j \in I(T)} x_{j,T}^k \right) \ . \tag{2.23}$$

Choosing (2.21) corresponds to summing the flow along the outward directed arcs of the source; (2.22) corresponds to the inward directed arcs of the terminal; and (2.23) is to combine both. These are examples of three different objective functions that model the same objective; maximizing the number of connecting paths.

Example of an objective function to minimize is the total cost of all arcs. This gives the objective

$$\min \quad \sum_{k=1}^{K} \sum_{(i,j) \in A} c_{i,j} x_{i,j}^k \ . \tag{2.24}$$

Many other objective functions are possible; these were only some examples.

Note that if we do not impose that there must be connecting paths in the directed network and we use a minimization objective, then the optimal solution is with no connecting paths.

2.8.2 Examples

It is now possible to define a mathematical model for the problems presented in Chapter 1, i.e., the time- and distribution problem.

The Time Problem

We define the chart with cities and their relative connections as the directed network $G = (V, A, d)$. It is constituted by the node set V representing the cities; the arc set A representing their connections; and the capacity function d representing the maximal possible loads for all roads.

The enterprise transports K goods to the customer; hence, K commodities from the source node to the terminal node in the directed network $G = (V, A, d)$.

Here the demand is known, and there are criteria to all connecting paths. Hence, we do not need to formulate the problem as a named IMCF model but can model it as an IMCF network model with the extensions of node disjointness, specified lengths, and approximately equal lengths.

We model node disjoint connecting paths because the customer demands that no two goods pass through the same location:

$$\sum_{k=1}^{K} \sum_{j \in O(i)} x_{i,j}^k \leq 1 , \quad \forall i \in V \backslash \{S, T\} .$$

The customer demands that the enterprise must comply with a delivery time L, which we model as specified lengths criteria:

$$\sum_{(i,j) \in A} c_{i,j} x_{i,j}^k \;\; \leq \;\; L , \quad \forall k \in \{1, \dots, K\} ,$$

in which $c_{i,j}$ is the time it takes to travel the road connecting city i and city j.

Furthermore, the customer expects that the K goods arrive within a fraction ρ from each other. This we model by:

$$\begin{aligned}
\sum_{k=1}^{K} \sum_{(i,j) \in A} x_{i,j}^k - Ky &= 0 , \\
\sum_{(i,j) \in A} x_{i,j}^k - (1 + \rho) y &\leq 0 , \quad \forall k \in \{1, \dots, K\} , \text{ and} \\
\sum_{(i,j) \in A} x_{i,j}^k - (1 - \rho) y &\geq 0 , \quad \forall k \in \{1, \dots, K\} .
\end{aligned}$$

The enterprise has the objective to make transport as cheap as possible, so we minimize all costs $c_{i,j}$ of all commodities k with their quantities q^k. That is,

$$\min \quad \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{i,j} q^k x_{i,j}^k \ .$$

All in all, the modeling of the time problem in Chapter 1 is:

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{i,j} q^k x_{i,j}^k \\
\text{s.t.} \quad & \sum_{j\in O(i)} x_{i,j}^k - \sum_{j\in I(i)} x_{j,i}^k = b_i^k \ , \quad \forall i \in V, k \in \{1,\ldots,K\} \\
& \sum_{k=1}^{K} q^k x_{i,j}^k \le d_{i,j} \ , \quad \forall (i,j) \in A \\
& \sum_{k=1}^{K} \sum_{j\in O(i)} x_{i,j}^k \le 1 \ , \quad \forall i \in V \backslash \{S,T\} \\
& \sum_{(i,j)\in A} c_{i,j} x_{i,j}^k \le L \ , \quad \forall k \in \{1,\ldots,K\} \\
& \sum_{k=1}^{K} \sum_{(i,j)\in A} x_{i,j}^k - Ky = 0 \\
& \sum_{(i,j)\in A} x_{i,j}^k - (1+\rho)y \le 0 \ , \quad \forall k \in \{1,\ldots,K\} \\
& \sum_{(i,j)\in A} x_{i,j}^k - (1-\rho)y \ge 0 \ , \quad \forall k \in \{1,\ldots,K\} \\
& x_{i,j}^k \in \{0,1\} \ , \quad \forall (i,j) \in A, k \in \{1,\ldots,K\} \ ,
\end{aligned}
$$

where the b_i^k are as specified in (2.4), i.e., $b_i^k = 1$ for $i = S$; -1 for $i = T$; and 0 otherwise. This is a mixed-integer linear program since it can be written in the form of (2.1)–(2.3) with $p = 1$.

The Distribution Problem

The distribution problem in Chapter 1 is with the same enterprise. However, the customer needs products distributed to its factories at different locations from a headquarter. The modeling of this alters the definitions of the nodes. The headquarter corresponds to the source, and the locations of the cities in which the J factories are situated correspond to the sub-terminals t_j. The directed network is then changed, so that all outward directed arcs from the sub-terminals are deleted. Furthermore, new arcs from the sub-terminals to a fictive terminal node T are introduced. This gives the altered directed network $G' = (A', V', d')$. Having introduced connection distribution criteria we now model the problem as a named IMCF model with extensions.

The customer demands that its products may not pass through the same location on their way to the factories, which we model by imposing node disjoint connecting paths in all nodes except for the source, terminal, and, now also, sub-terminals. Hence, node disjointness for the distribution problem is modeled by the set of constraints:

$$\sum_{k=1}^{K} \sum_{j \in O(i)} x_{i,j}^k \leq 1 \ , \quad \forall i \in V' \backslash \big\{\{S,T\} \cup \{t_j \ : \ j \in \{1,\ldots,J\}\}\big\} \ .$$

We add specified length criteria to comply with delivery time L. That is,

$$\sum_{(i,j) \in A} c_{i,j} x_{i,j}^k \ \leq \ L \ , \quad \forall k \in \{1,\ldots,K\} \ .$$

The commodities that must or may not enter factory t_j are included in N_{t_j} or $\overline{N}_{t_j}$, respectively. Hence, we model the distribution criteria by including the constraints

$$\begin{aligned} x_{t_j,T}^k = 1 \ , & \quad \forall j \in \{1,\ldots,J\}, \ k \in N_{t_j} \quad \text{and} \\ x_{t_j,T}^k = 0 \ , & \quad \forall j \in \{1,\ldots,J\}, \ k \in \overline{N}_{t_j} \ . \end{aligned}$$

The objective is to maximize number of products.[4] In other words, the objective is to find as many connecting paths as possible:

$$\max \quad \sum_{k=1}^{K} \sum_{j \in O(S)} x_{S,j}^k \ .$$

This leads to the modeling of the distribution problem in Chapter 1 written

[4] We assume that all products give equivalent profit.

as a named IMCF model:

$$
\begin{aligned}
\max \quad & \sum_{k=1}^{K} \sum_{j \in O(S)} x^k_{S,j} \\
\text{s.t.} \quad & \sum_{j \in O(i)} x^k_{i,j} - \sum_{j \in I(i)} x^k_{j,i} = 0 \ , \quad \forall i \in V' \backslash \{S,T\}, k \in \{1,\ldots,K\} \\
& \sum_{k=1}^{K} q^k x^k_{i,j} \leq d_{i,j} \ , \quad \forall (i,j) \in A' \\
& \sum_{k=1}^{K} \sum_{j \in O(i)} x^k_{i,j} \leq 1 \ , \quad \forall j \in V' \backslash \big\{ \{S,T\} \cup \{t_j \ : \ j \in \{1,\ldots,J\}\} \big\} \\
& \sum_{(i,j) \in A} c_{i,j} x^k_{i,j} \leq L \ , \quad \forall k \in \{1,\ldots,K\} \\
& x^k_{t_j,T} = 1 \ , \forall j \in \{1,\ldots,J\}, \ k \in N_{t_j} \ , \text{ and} \\
& x^k_{t_j,T} = 0 \ , \forall j \in \{1,\ldots,J\}, \ k \in \overline{N}_{t_j} \\
& x^k_{i,j} \in \{0,1\} \ , \quad \forall (i,j) \in A, k \in \{1,\ldots,K\} \ .
\end{aligned}
$$

This is a pure-integer program since it can be written in the form of (2.1)–(2.3) with $p = 0$.

Other Possibilities

There are many possible combinations of objectives and criteria. However, models must include a set of constraints that impose flow balance and a set of constraints that ensure that arc capacity limits are not exceeded. This is to sustain a feasible network flow.

Using the named IMCF model, it is possible to impose any combination of the following criteria, as well as many other criteria not mentioned here:

- Disjointness
  - Arc disjoint connecting paths
  - Node disjoint connecting paths
- Length
  - Specified lengths
  - Equal length
  - Approximately equal lengths
- Distribution
  - Connections
  - Forbidden area
  - Mandatory area

2.9 Challenges

Up until now, we have seen how to represent a variety of problems mathematically. However, there are two challenges that arise. These are possible occurrences of sub-tours in solutions and that the problems in the modeling become large-scale programs. These two challenges are discussed in this section.

2.9.1 Sub-tours

A main modeling challenge is that there may arise sub-tours in the solutions. A *sub-tour* is a directed path that is connected to itself, i.e., in which the initial and final nodes are the same in the sequence representing the directed path. This means that the solution may contain a connecting path that connects the source and terminal and has a separated sub-tour somewhere in the directed network. It is still a feasible network flow but not what is sought for since it does not model a solution to the problems of this thesis.

One could argue that the sub-tours could be removed from the solution afterwards. However, a consequence is that criteria no longer are ensured to be satisfied; removing a sub-tour changes the length of the connecting path, and a distribution criterion is no longer certain to be satisfied.

An illustration of a directed graph with two connecting paths is seen in Figure 2.5. The connecting paths are the arcs in bold; one is gray, and the other is black. The two connecting paths are of equal connecting path length since the sub-tour adds to the connecting path length of the gray connecting path. Hence, a length criterion of equal lengths is no longer satisfied if the sub-tour is simply removed. This is a challenge that we solve with an algorithm presented in Chapter 4.

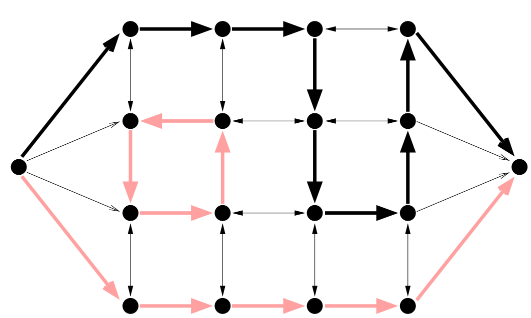

Figure 2.5: Directed graph with two connecting paths.

2.9.2 Large-scale Programs

Another main challenge is that the modeling gives rise to large-scale programs. Furthermore, the problem of finding arc- or node disjoint connecting paths with distribution criteria is NP-complete. It is also an NP-complete problem to find arc- or node disjoint connecting paths with equal- or approximately equal length criteria, and with greater than specified length criteria.[5] Hence, the practical difficulty of solving these problems increases fast with respect to the problem size, and especially with respect to the number of commodities. It is therefore of great interest to decrease the size of the program, i.e., decrease the number of variables. Note that since the problems are NP-complete, the best we can do is to make specific algorithms for solving them. In Chapter 4, we present some algorithms that deal with this issue.

2.10 Summary

The theory that is used in this thesis builds on IMCF, which is a form of MILP. A MILP is a mathematical model written in the form seen in (2.1)–(2.3); a linear model, in which the variables both can be integer and real numbers.

The main characteristic of IMCF is that it makes it possible to keep track of which commodity has a positive flow along which arc. It models a directed network by imposing conservation of flow at every node, and by restricting flows to exceed arc capacities.

The problem of finding arc- or node disjoint connecting paths is formulated as an IMCF network model with an extra constraint imposed. The extra constraint depends on whether the disjoint connecting paths are arc- or node disjoint connecting paths. When they should be arc disjoint connecting paths, the extra constraint states that no more than one type of commodity may share a common arc. However, node disjoint connecting paths, which implies arc disjoint connecting paths, is modeled by imposing that no more than one commodity may enter a node.

We saw the importance of introducing naming. This was when the number of connecting paths with criteria was less than the number of commodities. In these cases, we use the named IMCF model, instead of the IMCF network model, with which we could impose different length- and distribution criteria on the connecting paths.

The length criteria included both specific and comparisons of connecting path lengths. The specific connecting path length criteria included imposing connecting paths to be of less than, equal to, or greater than some connecting path lengths. Furthermore, we are now able to model length criteria imposing connecting paths to be of equal or approximately equal connecting path lengths.

We saw three different types of distribution criteria. These were connection and forbidden- and mandatory area distribution criteria. A connection distribution criterion defines which sub-sources are connected with which sub-

[5] See Appendix A for proofs hereof.

terminals. This is used when the different commodities source from and destine to different locations. The mandatory area criterion defines if some commodities must pass through or along some nodes or arcs. The opposite, forbidden area criterion, defines if some commodities are not allowed to pass through or along some nodes or arcs.

The different objectives and criteria can be combined in a number of ways such that we now can model problems that include disjointness and length- and distribution criteria.

The problem of finding arc- or node disjoint connecting paths with different criteria is NP-complete. This together with the IMCF network model produces large-scale programs is a major difficulty of the modeling. Another is that with this modeling there may arise sub-tours in the solutions. These challenges are dealt with in Chapter 4.

This chapter also modeled the time- and distribution problem from Chapter 1. Before discussing the challenges of sub-tours and large-scale programs, we see different solutions to these two problems in the following chapter.

Chapter 3

Tour de France

3.1 Introduction

Throughout Chapter 2, we referred to the time- and distribution problems. Some examples of these two problems are solved in this chapter to illustrate the appearance of solutions.

The setting takes place in France. The chart used by the enterprise is illustrated in Figure 3.1.[1] The data sets of the cities and roads are listed in Appendix B. The node set V consists of the cities in the intersection points, and the arc set A consists of the roads connecting the cities. There are two oppositely directed arcs for every road; illustrated by the bold lines.

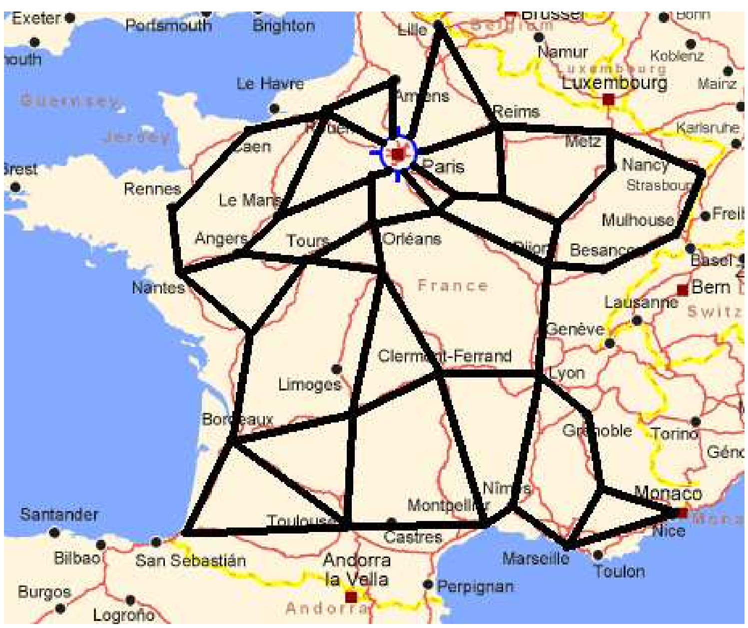

Figure 3.1: Chart of France used by the enterprise.

[1] The chart without the lines in bold is seen in Appendix B.

It is assumed that the goods have a size that is less than any upper limit of the roads. The capacities d of the roads are therefore not important since no two goods may pass through the same location. The time and costs of the roads are listed in Appendix B. These values have been obtained from `http://www.viamichelin.com`, as has the chart.

3.2 Time Problem

Recall that the time problem consisted of transporting goods as cheap as possible under the restriction that the goods arrive within a certain time and at approximately equal times. This problem is solved with the following criteria:

- there are 3 goods;
- Paris is the departure, and Toulouse is the destination;
- the time-limit is either 12 hours or 13 hours; and
- the margin is either 10% or 20%.

The solutions to the problems are seen in Figure 3.2. We see small variations in the node disjoint connecting paths. This is further seen in Table 3.1. The value of the objective function is the cost of using the three connecting paths. We see that the value of the objective function decreases when the criteria are less severe. In other words, when we allow a fluctuation of 20%, the cost is less than with 10%. Furthermore, having specific time-limits of 12 hours give higher costs than when we allow time-limits of 13 hours. We explain these observations by the fact that having stricter criteria decreases the feasible set of solutions, and therefore an optimal solution might not be feasible with some other stricter criteria.

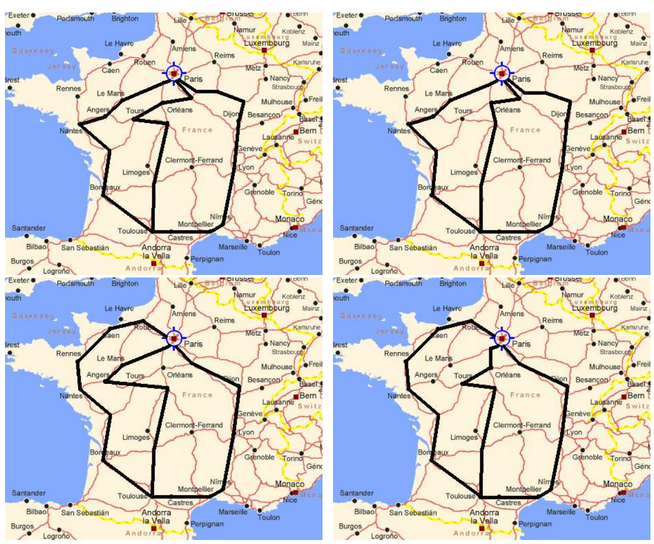

Figure 3.2: Solutions to time problem: for 12h and 10% (upper-left); for 12h and 20% (upper-right); for 13h and 10% (lower-left); and for 13h and 20% (lower-right).

Criteria	*Time 1*	*Time 2*	*Time 3*	*Cost*
12h and 10%	10h23	10h52	12h00	€354.24
12h and 20%	9h08	10h23	12h00	€337.31
13h and 10%	10h48	10h49	12h09	€353.40
13h and 20%	8h55	10h48	12h09	€334.38

Table 3.1: Summary of the solutions to time problem.

3.3 Distribution Problem

In the distribution problem, the goods are distributed to different factories under the restriction that the goods arrive within a certain time.

This problem is solved with the following criteria:

- the departure is Paris, and the factories are in Lille, Montpeiller, Nantes, and Strasbourg; and
- the time-limit is either 5 hours, 11 hours, or no time-limit.

The solutions to the problems are seen in Figure 3.3. We see that with the most severe criteria, i.e., a time-limit of 5 hours, the solution only consists of three node disjoint connecting paths: two going to Lille, and one to Nantes. However, when we have a time-limit of 11 hours, we see that there are 6 node disjoint connecting paths: two to Lille, two to Nantes, one to Strasbourg, and one to Montpeiller. In the last problem, in which we have no time-limit, we see that the connecting path from Paris to Montpeiller takes a detour. This is because we do not model finding a shortest connecting path, but instead model the problem of obtaining the greatest possible number of node disjoint connecting paths.

Figure 3.3: Solutions to distribution problem: for 5h (left); for 11h (middle); and for no time-limit (right).

3.4 Summary

We have seen the solutions to some examples of the time- and distribution problems. Here, we saw that changing the criteria alters the solution. In particular, we saw that with stricter criteria, e.g. a time-limit of 12 hours instead of 13 hours, the objective function values became poorer due to an optimal solution to one problem might not be feasible when we restrict the problem even more.

Chapter 4

Algorithms

4.1 Introduction

In Chapter 2, we saw how to model the problem of finding arc- or node disjoint connecting paths with different length- and distribution criteria. We also discussed some challenges connected with this model. This chapter gives algorithms that deal with these challenges.

One challenge of the modeling proposed in Chapter 2 is that sub-tours may arise in the solution. We give therefore an algorithm that finds solutions without occurrences of sub-tours. This is an essential algorithm since we are only interested in this type of solution. It builds on the same principle as a so-called cutting-plane algorithm.

The models will in Chapter 5 be optimized using a standard routine optimization software. However, large-scale NP-complete problems will in practice take very long time to optimize. We will see in Chapter 5 that finding solutions to a problem without criteria is faster than finding solutions to problems with criteria. This chapter therefore gives an algorithm that first checks whether it is possible to find the desired number of disjoint paths before solving the problem with criteria. Furthermore, we give an algorithm to treat large-scale programs by reducing the size of the program. This algorithm eliminates fixed variables and decomposes the program into smaller sub-programs. We present two ways in which to decompose the program into smaller sub-programs. They are both based on dividing the directed network into regions, which can be seen as geometric regions. The first way in which we decompose the program is with separate regions, and the second is with regions that overlap.

Throughout this chapter, we make references to Figure 4.1. It illustrates the sets of solutions and their mutual relationships. The largest set of solutions are those that are feasible network flows in which sub-tours may occur. Moreover, we see that node disjoint connecting paths without sub-tours with length- and distribution criteria constitute the smallest set of solutions. In this thesis, we are interested in the sets of solutions that are within the black- and gray boxes. These are solutions that are arc- or node disjoint connecting paths without sub-tours satisfying some length- and distribution criteria.

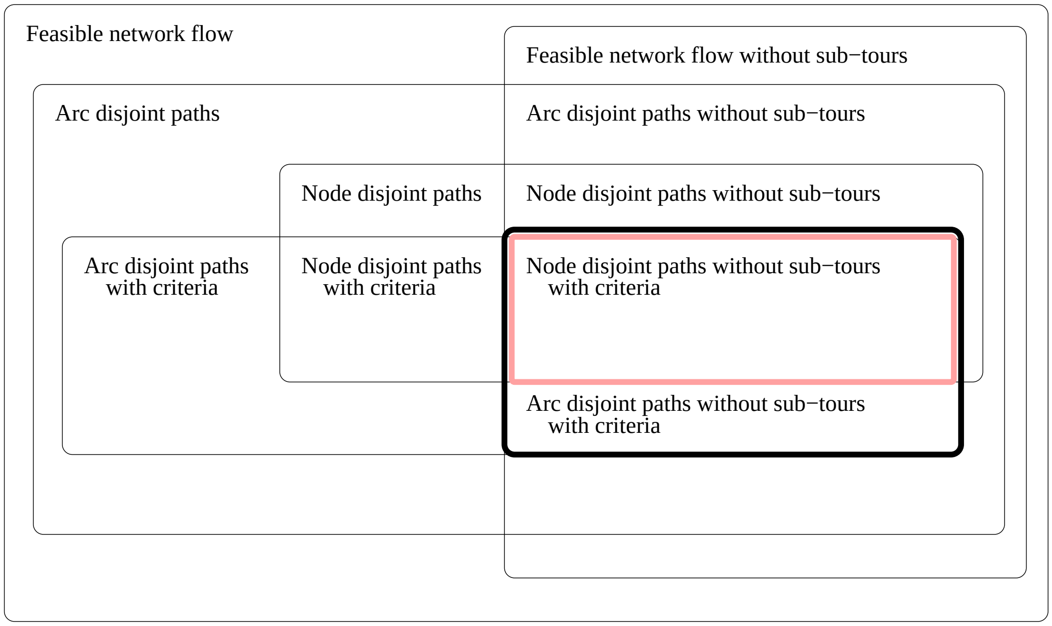

Figure 4.1: Sets of solutions and their mutual relationships.

In this chapter, we only consider finding arc- or node disjoint connecting paths. However, the algorithms can be altered so that they apply to capacitated network flow problems without the extra criterion of arc- or node disjoint connecting paths.

4.2 Disjoint Connecting Paths with Criteria

There is a maximal number of arc- and node disjoint connecting paths in a directed network. This maximal number might be less than the desired number of arc- and node disjoint connecting paths with length- and distribution criteria. As seen in Figure 4.1, the set of feasible network flow solutions with length- and distribution criteria F_{criteria} is contained within the set of feasible network flow solutions F:

$$F_{\text{criteria}} \subseteq F \ .$$

As we will see in Chapter 5, it is faster to find the maximal number of arc- and node disjoint connecting paths than to find arc- and node disjoint connecting paths with length- and distribution criteria. We therefore develop an algorithm that tests whether there are at least as many arc- or node disjoint connecting paths as commodities with criteria. This is done before solving the problem of finding arc- or node disjoint connecting paths with length- and distribution criteria. This algorithm is named DisjointPathsCriteria.

The node set V, the arc set A, and the capacity function d constitute the directed network $G = (V, A, d)$. It is necessary to know these pieces of information to evaluate the maximal number of arc- and node disjoint connecting paths, and they are, hence, given as input to DisjointPathsCriteria.

The maximal number of arc- and node disjoint connecting paths are found by having the objective: maximize the sum of flows along the outward directed arcs from the source. That is,

$$\max \quad \sum_{k=1}^{K} \sum_{j \in O(S)} x_{S,j}^{k} \ .$$

The objective should still be found under the restrictions that the found flow is a feasible network flow with arc- or node disjoint connecting paths.

Knowing the maximal number of arc- or node disjoint connecting paths, DisjointPathsCriteria either stops or continues. It stops if the maximal number is less than the number of connecting paths with criteria, denoted $|N|$, and continues if this number is greater than or equal to $|N|$. If DisjointPathsCriteria continues, it optimizes the original problem, i.e., the problem with the different length- and distribution criteria. To decide whether to stop or to continue, DisjointPathsCriteria also needs $|N|$ as a piece of information. Furthermore, we must state the length- and distribution criteria, and give these as input. DisjointPathsCriteria returns a solution if there exist a solution to the posed problem. The described algorithm, DisjointPathsCriteria, is presented below as Algorithm 4.2.1.

Algorithm 4.2.1: DisjointPathsCriteria($network, criteria, |N|$)

```
main
  maximal = MaxFlow(network)
  if maximal < |N|
    then return (infeasible)
    else { solution = CriteriaFlow(network, criteria)
         { return (solution)
  endif

procedure MaxFlow(network)
  maximal = find maximal number of disjoint connecting paths in network
  return (maximal)
procedure CriteriaFlow(network, criteria)
  solution = find disjoint connecting paths in network satisfying criteria
  if feasible
    then return (solution)
    else return (infeasible)
  endif
```

4.3 Sub-tour Elimination

The proposed modeling finds solutions with possible occurrences of sub-tours, which is not what we wish.[1] Sub-tours in a solution cannot simply be removed since such a removal changes the length of the connecting path, and the solution might no longer satisfy the length- and distribution criteria. Hence, it is necessary to resolve this challenge through another approach that eliminates all sub-tours in the solution while still having connecting paths satisfying the length- and distribution criteria.

A similar challenge is known from the traveling salesman problem; see [9, pp. 9–10]. The solutions to the traveling salesman problem may also include occurrences of sub-tours. The text [9, pp. 9–10] proposes a model with a set of constraints that disallows sub-tours. However, this gives a huge number of constraints. Hence, the same text proposes an iterative algorithm instead in which constraints are added when needed. This is a so-called cutting-plane algorithm; see [9, II.5.2].

The traveling salesman problem differs from the problems of this thesis since the salesman must pass through all nodes in the directed or undirected network. Hence, we cannot use the same sub-tour elimination constraints but we use the same principle with a cutting-plane algorithm, which is described below.

Theoretically, it is possible to impose constraints disallowing all sub-tours. It could be done by imposing that the sum of all variables constituting a potential sub-tour must be less than the number of arcs in that sub-tour. Let Δ denote the set of all sub-tours. Furthermore, let $|\delta|$ be the number of arcs in sub-tour $\delta \in \Delta$. Then sub-tours are disallowed in the directed network $G = (V, A, d)$ by the set of constraints

$$\sum_{k=1}^{K} \sum_{(i,j)\in\delta} x_{i,j}^{k} \leq |\delta| - 1 \ , \quad \forall \delta \in \Delta \ . \tag{4.1}$$

There are many possible sub-tours; we give three examples in Figure 4.2.

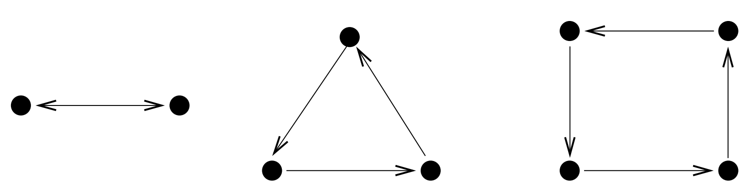

Figure 4.2: Three examples of possible sub-tours in a solution.

To eliminate all sub-tours, there must be as many constraints as sub-tours. However, it is in general not a practical approach. Instead, we restrict the mod-

[1] We defined a sub-tour to be a directed path having the same initial and final node.

eling to find solutions without sub-tours iteratively by adding more constraints to the model. This algorithm is the SubTourElimination algorithm.

Including the sets of sub-tour elimination constraints does not exclude any solutions without sub-tours. This is illustrated in Figure 4.1, in which the set of feasible network flow solutions F contains the set of feasible network flow solutions without sub-tours $F_{\text{w.o. sub-tours}}$. That is,

$$F_{\text{w.o. sub-tours}} \subseteq F \ .$$

SubTourElimination begins by finding all sub-tours in a given solution. To do so, a solution must be given as input. The algorithm returns the same solution as given in input if there does not exist sub-tours in the solution. However, if there are sub-tours, SubTourElimination imposes sub-tour elimination constraints disallowing these sub-tours. Let δ denote a found sub-tour, and $|\delta|$ the number of arcs in that sub-tour. The constraint

$$\sum_{k=1}^{K} \sum_{(i,j)\in\delta} x_{i,j}^{k} \leq |\delta| - 1 \tag{4.2}$$

disallows this sub-tour. In SubTourElimination, constraint (4.2) is imposed for all found sub-tours, and the extended model is then re-solved using the same procedure CriteriaFlow as in Algorithm 4.2.1. In order to do this, SubTourElimination also takes the input a directed network $G = (V, A, d)$ and the criteria to the connecting paths.

This gives the SubTourElimination algorithm presented as Algorithm 4.3.1.

Algorithm 4.3.1: SubTourElimination(*network*, *criteria*, *solution*)

main
  subTours = find sub-tours in *solution*
  while *subTours* exist
  do {
    constraints = SubTourConstraints(*solution*)
    add *constraints* to *criteria*
    solution = CriteriaFlow(*network*, *criteria*)
    subTours = find sub-tours in *solution*
  }
  return (*solution*)

procedure SubTourConstraints(*solution*)
  subTours = find sub-tours in *solution*
  for $i = 1$ **to** #(*subTours*)
  do {
    constraints(i) =
    $\sum$(variables in *subTours*(i)) $\leq$ (length of *subTours*(i)) $- 1$
  }
  return (*constraints*)

A variation of this algorithm is to include sets of sub-tour elimination constraints for some types of sub-tours, such as the sub-tours constituted by the smallest numbers of arcs. It is our assumption that these sub-tours are those that occur the most frequently. This variation is used when we assess that certain types of sub-tours are very likely to occur if they are not prohibited.

4.4 Large-scale Programs

In this thesis, we wish not only to model problems but also to optimize and find solutions. There is no general optimal algorithm for solving large-scale NP-complete problems. This section therefore gives two types of algorithms used to solve large-scale programs: one preprocessing algorithm and one decomposing algorithm. The decomposing algorithm is especially designed to directed networks that can be thought of as geometric areas. It decomposes the directed network into several smaller regions, and solves them separately.

There are existing algorithms for solving large-scale programs without decomposing, such as the Augmented Lagrangian Algorithm and Column Generation; see [8] and [2, 6], respectively. However, we propose a new algorithm since in many network problems it is possible to exploit some of its structure to divide the directed network into smaller regions. Furthermore, as discussed in Chapter 2, the problem of finding arc- or node disjoint connecting paths is NP-complete. Hence, even though there exist algorithms to solve large-scale programs, it might in practice be faster to solve several smaller sub-programs than one large-scale NP-complete problem.

The preprocessing algorithm eliminates all fixed variables from the program. These are the variables that describe either a mandatory- or forbidden area.

The decomposition algorithm decomposes the program into several smaller sub-programs. There are many possible ways to do so, and many considerations to make. First we discuss how to handle criteria when decomposing the program, and then we give two algorithms depending on the type of division of the directed network.

4.4.1 Preprocessing

When there are mandatory- or forbidden area distribution criteria in the directed network, these variables are fixed and can therefore be removed from the program. In the following, we discuss how to make the program smaller using this information.

Forbidden distribution criteria entail that the corresponding variables must be inactive. With preprocessing, we eliminate these fixed variables instead of adding constraints to the program. We saw in Chapter 2 that imposing that some arcs are inactive gives the set of constraints

$$x^k_{i,j} = 0 \ , \quad \forall k \in \{1, \dots, K\},\ (i,j) \in \overrightarrow{F}^k \ ,$$

where $\overrightarrow{F}^k$ is the set of all forbidden arcs for commodity k. Instead of imposing this set of constraints, we remove the variables $\{x^k_{i,j} \ : \ k \in \{1, \dots, K\},\ (i,j) \in \overrightarrow{F}^k\}$.

Similarly, when there are forbidden area distribution criteria on nodes, the inward- and outward directed arcs for the specified commodity k can also be removed. The set of nodes forbidden for commodity k are denoted $\dot{F}^k$. Instead of imposing the constraints

$$\begin{aligned} x^k_{i,j} &= 0 \ , \qquad \forall k \in \{1, \dots, K\},\ j \in \dot{F}^k,\ i \in I(j) \quad \text{and} \\ x^k_{i,j} &= 0 \ , \qquad \forall k \in \{1, \dots, K\},\ i \in \dot{F}^k,\ j \in O(i) \ , \end{aligned}$$

we remove the variables $\{x^k_{i,j} \ : \ k \in \{1, \dots, K\},\ j \in \dot{F}^k,\ i \in I(j)\}$ and $\{x^k_{i,j} \ : \ k \in \{1, \dots, K\},\ i \in \dot{F}^k,\ j \in O(i)\}$.

Preprocessing by using information about mandatory distribution criteria can only be used when these criteria are imposed on the arcs. Here, we eliminate the variables by altering the sets of equations. Let $\overrightarrow{M}^k$ be the set of mandatory arcs for commodity k. We saw in Chapter 2 that imposing that the variables corresponding to these arcs are active gives the set of constraints

$$x^k_{i,j} = 1 \ , \quad \forall k \in \{1, \dots, K\},\ (i,j) \in \overrightarrow{M}^k \ .$$

To eliminate these variables, we first subtract the variables from the right-hand sides of the constraints containing these variables. It is then possible to eliminate the variables from the program. However, it is important to remember to include these variables in the solution afterwards, since they are part of it.

This leads to the Preprocessing algorithm. It takes as input some of the same pieces of information as DisjointPathsCriteria. These are the structure of the directed network and the imposed criteria. Furthermore, it gives the altered network and active variables as output. The Preprocessing algorithm is presented as Algorithm 4.4.1.

Algorithm 4.4.1: PREPROCESSING($network, criteria$)

main
$alteredNetwork =$ FORBIDDENAREA($network, criteria$)
($alteredNetwork, activeVariables$) =
MANDATORYAREA($alteredNetwork, criteria$)
return ($alteredNetwork, activeVariables$)

procedure FORBIDDENAREA($network, criteria$)
$inactiveVariables =$ find inactive variables from $criteria$
$alteredNetwork =$ remove $inactiveVariables$ from $network$
return ($alteredNetwork$)
procedure MANDATORYAREA($network, criteria$)
$activeVariables =$ find active variables from $criteria$
$alteredNetwork =$ subtract and remove $activeVariables$ from $network$
return ($alteredNetwork, activeVariables$)

4.4.2 Decomposition into Smaller Sub-programs

There are several aspects to consider when decomposing a program into smaller sub-programs, such as how to handle criteria, how to choose a division, and how to obtain one feasible solution.

The decomposition algorithms vary with respect to how the directed network is divided into regions, which should be thought of as geometric regions. We present two variations of divisions: one with separate regions and one with overlapping regions. One feasible solution is obtained differently depending on the type of division but criteria are handled in the same manner for the two cases. We therefore explain how to handle the criteria first, discuss the type of regions, and finally we give the two algorithms that decompose the program into smaller sub-programs.

How to Handle Criteria

The main issue when dividing the directed network into smaller geometric regions is that we might exclude possible solutions. Hence, when there are mandatory area- or connection distribution criteria, it is important to construct divisions that do not violate these distribution criteria.

The set of criteria must be altered when we impose equal length- or approximately equal lengths criteria. With equal length criterion, we impose this in the first region to be solved, and then use specified equal length criteria in the other regions with the found length from the first. Similarly, with approximately equal lengths criteria, the found average connecting path length from the first region is given as input for the average connecting path length in the other regions.

Hence, we note that not only is the division important for the feasibility of the problem but, when we include equal- or approximately equal length criteria, it is also important in which order we solve the different regions.

Types of Regions

There are two types of division: one where the regions *separate* and another where the regions *overlap*. With separate regions no arcs are shared by any two regions, whereas this is the case with overlapping regions. Figure 4.3 illustrates the two types of divisions. To the left, we see a directed graph divided into two separate regions. The directed graph to the right is divided into two overlapping regions. Note that the middle arcs in the directed graph to the left are not included in any region.

A region from a division with overlapping regions contains a *main part* and an *overlapping part*. The main part of a region is what is not shared by any other region, whereas the overlapping part are the arcs in the overlapping region. In Figure 4.3, the black arcs are in the main part of Region 1; the dark gray arcs are in the main part of Region 2; and the overlapping part is for both Region 1 and Region 2 the light gray arcs.

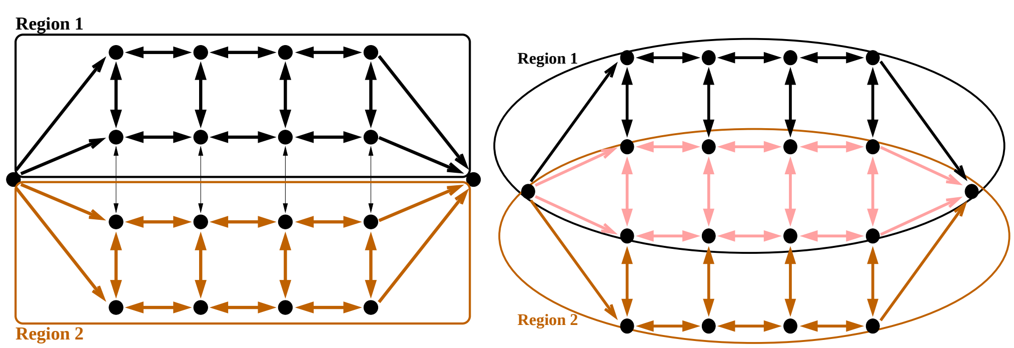

Figure 4.3: Separate regions (left) and overlapping regions (right).

Again, we note that with overlapping regions, the order in which we solve the regions is important for the feasibility of the problem.

Decomposition Algorithms

The outline of a decomposition algorithm depends on the type of region division. We propose an algorithm that uses overlapping regions. However, this algorithm incorporates many details, so first we present the algorithm as it is when regions are separate, since this algorithm gives a general outline of how to decompose the program into smaller sub-programs. Hereafter, we present the decomposition algorithm that we use in this thesis.

The DECOMPOSITION algorithm divides the directed network into separate regions. It begins by preprocessing the program before dividing it into several smaller sub-programs. DECOMPOSITION then solves each sub-program separately. This is done by reusing DISJOINTPATHSCRITERIA and SUBTOURELIMINATION. A new sub-program is solved as long as feasible solutions exist. The algorithm stops if there are no feasible solutions to one of the smaller sub-programs. DECOMPOSITION is presented below as Algorithm 4.4.2. Again, this algorithm needs the same pieces of information about the directed network and the criteria as DISJOINTPATHSCRITERIA and, besides this, also how many regions D it should generate.

Algorithm 4.4.2: DECOMPOSITION(*network*, *criteria*, D)

```
main
  (network, activeVariables) = PREPROCESSING(network, criteria)
  (regions, criteriaRegions, |N|) = DIVISION(network, criteria, D)
  for i = 1 to D
      do { solutions(i) =
              DISJOINTPATHSCRITERIA(regions(i), criteriaRegions(i))
           solutions(i) =
              SUBTOURELIMINATION(regions(i), criteriaRegions(i), solutions(i))
           if solutions(i) is infeasible
             then return (infeasible), STOP
           endif
  solutions = add activeVariables to solutions
  return (solutions)

procedure DIVISION(network, criteria, D)
  regions = divide network into D regions
  criteriaRegions = divide criteria so that they correspond to all D regions
  |N| = calculate number of criteria in criteriaRegions
  return (regions, criteriaRegions, |N|)
```

Note that the DECOMPOSITION algorithm also must take into account the issues there are about the length- and distribution criteria.

The decomposition algorithm proposed in this thesis includes some more details than DECOMPOSITION. In particular, we use overlapping regions, and we also make a choice as to which arc- or node disjoint paths we add to the final solution. This algorithm is the DECOMPOSITIONOVERLAP algorithm (Algorithm 4.4.3). The description of DECOMPOSITIONOVERLAP follows directly after the algorithm.

Algorithm 4.4.3: DecompositionOverlap(*network*, *criteria*, D, *overlap*)

```
main
 (network, activeVariables) = Preprocessing(network, criteria)
 (regions, criteriaRegions, |N|) = Division(network, criteria, D, overlap)
 for i = 1 to D
        ⎧ solutions(i) =
        ⎪   DisjointPathsCriteria(regions(i), criteriaRegions(i))
        ⎪ solutions(i) =
        ⎪   SubTourElimination(regions(i), criteriaRegions(i), solutions(i))
        ⎪ if solutions(i) is infeasible
        ⎪   then return (infeasible), STOP
        ⎪ endif
        ⎪ if i = 1
   do   ⎨   then solutions(1) = keep solutions(1) that do not begin in overlap(1)
        ⎪   else solutions(i) = keep solutions(i) that do not begin in overlap(i)
        ⎪                 or that do begin in overlap(1) ... overlap(i − 1)
        ⎪ endif
        ⎪ if solutions(i) is within overlap(i)
        ⎪   then ⎧ add overlapping solutions(i) as forbidden area to
        ⎪        ⎩ criteriaRegions for regions sharing overlap(i)
        ⎩ endif
 solutions = add activeVariables to solutions
 return (solutions)

procedure Division(network, criteria, D, overlap)
 regions = divide network into D regions with overlap
 criteriaRegions = divide criteria so that they correspond to all D regions
 |N| = calculate number of criteria in criteriaRegions
 return (regions, criteriaRegions, |N|)
```

DecompositionOverlap needs the same pieces of information as Decomposition, and it also needs information about the overlap for each region. The only new part in DecompositionOverlap is the second half of the for-loop. Here, DecompositionOverlap determines which part of the found solution to keep. We keep all arc- or node disjoint connecting paths beginning in the main part, and also we keep the arc- or node disjoint connecting paths that begin in an overlapping part if this part belongs to a region that we have already solved. This means that for the directed graph to the right in Figure 4.3, we would solve for Region 1 and though we could find three arc- or node disjoint connecting paths we would only keep one. However, we would keep all the arc- or node disjoint connecting paths that we find in Region 2 since its overlapping part is the same as Region 1: an already solved region.

When we know which arc- or node disjoint connecting paths to keep, we check to see whether some of these paths use arcs in the overlapping part. If this is the case, then this becomes a forbidden area in all other regions sharing this overlapping part. This is because the collection of connecting paths from the different regions otherwise might neither be arc- nor node disjoint.

The reason that we can use overlapping regions in the decomposition is because we model the problem as a named IMCF model and not as an IMCF network flow model. Recall that with the IMCF network flow model, we assumed to know which commodities must have a corresponding arc- or node disjoint connecting path. With overlapping regions, we might in one region have fewer arc- or node disjoint connecting paths than corresponding commodities entering in this region. This is why we only impose that the commodities beginning in the main part or the overlapping parts of regions already solved must have corresponding arc- or node disjoint connecting paths. Hence, these commodities are actively named, whereas the rest are inactively named.

Furthermore, we use the distribution criterion of forbidden area when a given solution in one region uses arcs in the overlapping part. This added criterion depends on whether we want arc- or node disjoint connecting paths. When the problem is to find arc disjoint connecting paths, then the already used arcs in the overlapping region must be disallowed. Let $\overrightarrow{F}$ denote the already used arcs in the overlapping region, and K the number of commodities, the already used arcs $(i,j) \in \overrightarrow{F}$ are then disallowed for all commodities by imposing the criteria

$$x^k_{i,j} = 0 \ , \quad \forall k \in \{1, \ldots, K\}, \ (i,j) \in \overrightarrow{F} \ .$$

In the other situation, in which the problem is to find node disjoint connecting paths, the already used nodes in the overlapping region must be disallowed. That is, all inward directed arcs to these forbidden nodes are disallowed. This is done by imposing the criteria

$$\sum_{i \in I(j)} x^k_{i,j} = 0 \ , \quad \forall k \in \{1, \ldots, K\}, \ j \in \dot{F} \ ,$$

where $\dot{F}$ is the collection of nodes that are already used.

However, as we mentioned in the first part of this section, we decide how to decompose into regions. With this decision, we might exclude some possible solutions. We denote the set of possible solutions that may be found with a given decomposition $F_{\text{decomposition}}$. This is a subset of the set of feasible solutions to the problem of finding arc- or node disjoint connecting paths without sub-tours with criteria $F_{\text{w.o. sub-tours w. criteria}}$. That is,

$$F_{\text{decomposition}} \subseteq F_{\text{w.o. sub-tours w. criteria}} \ .$$

Since the two subsets $F_{\text{decomposition}}$ and $F_{\text{w.o. sub-tours w. criteria}}$ are not necessarily equal, there may be a solution to the problem even though there be no solution in some region. So once more, the division must be chosen carefully.

4.5 Summary

The problem of finding connecting paths with different length- and distribution criteria is solved by using DisjointPathsCriteria. This algorithm first checks whether there is the desired number of arc- or node disjoint connecting paths in the directed network before solving the problem with criteria.

Furthermore, we have seen how to resolve the challenge of eliminating sub-tours in a solution. The elimination of sub-tours was resolved by using a cutting-plane algorithm that added constraints wherever there appeared sub-tours. These constraints prohibited the same sub-tour from re-occurring. The procedure was continued until no more sub-tours appeared. We noted that this does not eliminate any of the solutions we want, i.e., solutions without sub-tours.

A challenge with the modeling of the problems in this thesis is that we obtain large-scale programs. We gave a decomposition algorithm that decomposes the large-scale program into several smaller sub-programs. This is done by dividing the directed network into separate or overlapping regions, where we presented an algorithm using overlapping regions. We proposed the decomposition algorithms because it might be possible to exploit some of the structure of the directed network, and because it might speed up the optimization of the large-scale NP-complete problems. These algorithms intend to reduce the size of the program. However, when the program is decomposed into sub-programs, independent of which type of region separation we choose, it is possible that we have excluded some feasible solutions by our choice.

Chapter 5

Implementation and Numerical Experiments

5.1 Introduction

This chapter discusses some issues of the implementation and includes the results of some numerical experiments. The given models from Chapter 2 and the algorithms from Chapter 4 are implemented using the commercial software MatLab.[1] These models are optimized using the commercial branch & cut optimizer CPLEX.[2] We use an interface that writes the matrices in MatLab into the format that CPLEX uses. This interface is CPLEX MEX INTERFACE version 2.1.[3] The numerical experiments were executed on a shared server using the default settings in CPLEX. We are allotted 1 CPU at the server consisting of 48 processors UltraSparc-IIICu 900 MHz CPUs with 144 GB memory.[4]

The models are written in matrix-form. The reader familiar with MatLab knows that this software is not suitable for large-scale programs. However, MatLab suffices to make a proof of concept that both the modeling is correct and that the algorithms work. We first discuss this representation, and then a property of the matrix-form, namely sparsity.

The numerical experiments presented in this chapter are aimed at investigating the difficulty of the problems and the property of the DecompositionOverlap algorithm. We use a test set of 7 tests (test problem a – test problem g) presented in Appendix C. In short, test problem a is the largest test problem followed by test problem d, test problem g is the smallest, and the rest are of approximately equal sizes.

[1] MatLab version 7.0.1 (R14); see `http://www.mathworks.com/` for more details.

[2] ILOG CPLEX version 9.0.0; see `http://www.ilog.com/` for more details.

[3] It is free software written by Mato Baotic and Fabio D. Torrisi. It can be downloaded from The Hybrid System's Group university home page: `http://control.ee.ethz.ch/~hybrid/cplexint.php`. It has been slightly changed, so that it returns feasible solutions that are not necessarily optimal.

[4] This server works slower but has more memory than most personal computers.

The difficulties of the problems are investigated by comparing relaxed-, feasible-, and optimal solutions. A relaxed solution is when we let integer variables be real variables; it enables us to analyze the impact the criteria of integer variables have on the solution. We make these comparisons for the problem of finding node disjoint connecting paths while either maximizing the number of connecting paths, in which we do not add criteria, or minimizing the connecting path lengths, in which we add one of the following criteria:

- approximately equal lengths,
- greater than lengths, and
- less than lengths.

The DecompositionOverlap algorithm is tested with different number of regions and sizes of overlap. These experiments are described by listing calculation time, number of iterations, and number of times the problem is re-solved due to sub-tours.

In short, in both investigations, we say that the less the time and the fewer the iterations, the easier the problem was to solve.

5.2 Matrix Products and Structures

A MILP consists of linear models and can therefore be written as matrix products. In particular, a MILP can be written in the form:

$$\begin{aligned} \max / \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathbf{h}^{\mathrm{T}}\mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{G}\mathbf{y} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}_+^n \quad \text{and} \quad \mathbf{y} \in \mathbb{R}_+^p \ . \end{aligned} \tag{5.1}$$

The matrices $\mathbf{A}$ and $\mathbf{G}$ are the *constraint matrices* in which each row corresponds to a constraint, and each column to a variable. Having m constraints, n non-negative integer variables, and p non-negative real variables, the matrices $\mathbf{c}$, $\mathbf{h}$, $\mathbf{A}$, $\mathbf{G}$, and $\mathbf{b}$ are of sizes $n \times 1$, $p \times 1$, $m \times n$, $m \times p$, and $m \times 1$, respectively, with $p + n \geq 1$. Furthermore, all matrices have real coefficients.

There are many possible representations of variables and matrices. To facilitate the writing of the constraint matrices in MatLab, we represent all K variables corresponding to one arc jointly. This enables the use of kronecker products, which is an operator that repeats a matrix in another. Denoting the kronecker product $\otimes$, an example is:

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \otimes \mathbf{A} = \begin{pmatrix} a_{1,1}\mathbf{A} & a_{1,2}\mathbf{A} \\ a_{2,1}\mathbf{A} & a_{2,2}\mathbf{A} \end{pmatrix} \ .$$

The kronecker product is an operator that we use extensively throughout the implementation since many of the constraints are similar for all K representations of an arc.

5.3 Sparsity

One challenge with writing the model as an IMCF network model or as a named IMCF model is that this gives large-scale programs. However, a reason that we are able to both write the models as matrices and then solve the programs is because the constraint matrices are sparse, where sparse means that only few of the elements in the matrices are non-zero. A way to appreciate how sparse the constraint matrix is, is by looking at the percentage of non-zero elements: the *sparsity* of the matrix. We do this on the IMCF network model by first estimating the number of non-zero elements and then calculating the sparsity.

The sets of constraints that impose a feasible network flow are the sets of constraints from arc capacity limitations and from flow conservation. In the following, K is still the number of commodities, and the directed network consists of the arc set A, the node set V, and the capacity function d. We denote the number of arcs and nodes $|A|$ and $|V|$, respectively.

Arc capacity limitations must hold for all arcs. Hence, the number of constraints modeling arc capacity limitations $\#_{Ac}$ is the number of arcs in the directed network:

$$\#_{Ac} = |A| \ .$$

Furthermore, the number of non-zero elements in these constraints $\#_{Ae}$ is the number of commodities times the number of arcs:

$$\#_{Ae} = K|A| \ .$$

To have a feasible network flow, flow conservation must hold for all nodes. Hence, there are

$$\#_{Fc} = |V|$$

constraints to model flow conservation.

The number of non-zero elements in each constraint depends on the number of inward- and outward directed arcs. We simplify the estimation by assuming that they are equal, and denote this number $|O(\cdot)|$. Hence, the number of non-zero elements from the set of constraints modeling flow conservation $\#_{Fe}$ is the number of commodities times the number of inward- and outward directed arcs times the number of nodes. That is,

$$\#_{Fe} = 2K|O(\cdot)||V| \ .$$

Furthermore, to estimate the sparsity we need to estimate the number of variables $\#_{var}$, which is the number of commodities times the number of arcs:

$$\#_{var} = K|A| \ .$$

We have assumed that the number of inward- and outward directed arcs are equal, so we can represent the number of arcs by the number of nodes. For every node there are $|O(\cdot)|$ arcs, giving

$$|A| = |V||O(\cdot)| \ .$$

Now, the sparsity of the constraint matrix modeling a feasible network flow is the fraction of the sum of non-zero elements divided by the total number of elements. The total number of elements is the number of variables times the number of constraints:

$$\begin{aligned}\frac{\#_{Ae} + \#_{Fe}}{(\#_{Ac} + \#_{Fc})\#_{var}} &= \frac{|A|K + 2K|O(\cdot)||V|}{(|A| + |V|) \cdot K|A|} \\ &= \frac{|A| + 2|A|}{(|A| + |V|) \cdot |A|} \\ &= \frac{3}{|A| + |V|} \; .\end{aligned}$$

Hence, directed networks with $|A| + |V| > 300$ have constraint matrices modeling feasible network flow with sparsity of less than 1%. This is a very low sparsity. Matrices modeling disjointness and distribution criteria also show low sparsity, but matrices modeling length criteria do not. Figure 5.1 illustrates a matrix modeling node disjoint connecting paths in a directed network with approximately equal lengths and connection- and forbidden area distribution criteria. The non-zero elements are illustrated by dots. Few elements are non-zero except for the last few rows that model the length criterion. There are 56,256 non-zero elements out of 33,141,760 elements giving 0.17% sparsity.

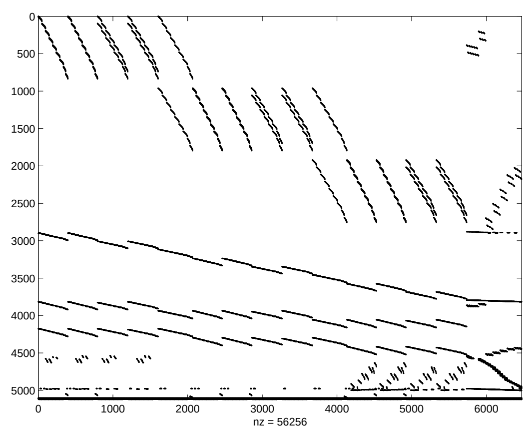

Figure 5.1: Example of a sparsity pattern for a constraint matrix.

Throughout the following, we define the matrices as being sparse, and allocate the necessary amount of memory to the matrices before constructing them. With this we are able to define matrices of sizes of approximately $15,000 \times 60,000$. Table 5.1 gives some results of how large constraint matrices we can construct.

$\|V\|$	$\|A\|$	K	*Rows*	*Columns*	*Possible*
600	2660	20	12600	53200	Yes
900	4020	15	14400	60300	Yes
900	4030	20	18900	80600	No

Table 5.1: Sizes of constraint matrices modeling a feasible network flow.

We see that the size of the directed network that we can model is limited; the first two constraint matrices can be written, whereas the last cannot since MatLab runs out of memory. Furthermore, it does not only depend on the size of the directed network, i.e., how many nodes $|V|$ and arcs $|A|$ there are in the directed network, but also on the number of commodities K. We restate that MatLab is used because we make a proof of concept. These sizes suffice for illustrating the algorithms from Chapter 4.

5.4 Experimenting with Criteria

Theoretically, the difficulty of the problem depends on the imposed criteria; the problem with less than or equal to specified lengths criteria should be an easier problem than problems with other length criteria.[5] In the following, we investigate whether this also holds numerically by comparing problems with different criteria. The tests performed are given in Table 5.2.

Test	*Explanation*
1	*Objective:* maximize number of connecting paths *Criteria:* node disjointness *Constants:* none
2	*Objective:* minimize average connecting path length *Criteria:* node disjointness + approximately equal lengths *Constants:* $\rho = 0.2$, $\rho = 0.5$, and $\rho = 0.8$
3	*Objective:* minimize connecting path lengths *Criteria:* node disjointness + greater than specified lengths *Constants:* $L = 3$ and $L = 6$
4	*Objective:* minimize connecting path lengths *Criteria:* node disjointness + less than specified lengths *Constants:* $L = 6$ and $L = 10$

Table 5.2: Tests with different criteria.

The results from Tests 1–4 are given in Tables 5.4–5.7 in Section 5.7.1. We use the following notation:

[5] See Appendix A for details.

- empty fields are missing values;[6] and
- () and * mark, respectively, optimization time before exiting due to memory limitations and corresponding values that can therefore not be obtained.

The values listed in Tables 5.4–5.7 are:

$T^r(s)$	:	optimization time to calculate relaxed solution,
$T^*(s)$	:	optimization time to calculate feasible solution,
$T(s)$	:	optimization time to calculate optimal solution,
f^r	:	function value of the relaxed solution,
f^*	:	function value of the feasible solution,
f	:	function value of the optimal solution,
it^r	:	iterations to calculate relaxed solution,
it^*	:	iterations to calculate feasible solution,
it	:	iterations to calculate optimal solution,
$\notin \mathbb{Z}$	:	percentage non-integers in relaxed solution, and
δf	:	difference in percentage between function values of relaxed- and optimal solutions.

Furthermore, ρ and L denote the values of the margin and specified lengths, respectively.

The results of Test 1 are characterized by having short optimization times, equal function values for relaxed-, feasible-, and optimal solution, and few non-integer values in the relaxed solution. We therefore say that finding the maximal number of node disjoint connecting paths is an easy problem to solve. This was what motivated us to create the DisjointPathsCriteria algorithm.

However, in Tests 2–4, we see completely different results. Starting with Test 2, we see some very big differences between the relaxed- and feasible- /optimal- solutions. Finding feasible solutions has now become very time consuming (it can take up to 13 hours), and we have only been able to find the optimal solution in the smallest test problem (test problem g) for approximately equal lengths with a margin of 80% even though there evidently must be optimal solutions in all cases.[7] However, finding relaxed solutions is still fast. Furthermore, the optimization of the relaxed solutions is not affected by the value of the margin (neither optimization times nor function values) but we see a big increase in optimization times with narrower margins for the feasible solutions. In Test 2, there are few non-integers in the relaxed solution. However, these few variables have great impact on the solution; for test problem g with a margin of $\rho = 0.8$, it takes 755 seconds to find an optimal solution

[6] We experienced problems in the optimization such as the server being down for a week. That, combined with some results taking 5 days, made it impossible to obtain all values within the deadline of the report.

[7] We find feasible solutions in all cases, and since there only is a finite number of feasible solutions, there must be an optimal solution among them.

compared to 1 second for a relaxed solution, and there is a 21% difference in the function values. All in all, we can say that Test 2 with small margin values is a difficult problem to solve, and, independently of the margin values, requesting an integer solution makes the problem drastically more difficult.

Tests 3 and 4 behave in likewise manners. We see that the results from finding greater than specified lengths of 3 are almost the same as the results from finding less than specified lengths of 10; optimization times are very fast, and the function values are identical except for test problem g in which there is a slight difference. What is more, the few non-integers in the relaxed solution do not affect the function values with much. Hence, we conclude that these two problems are easy problems to solve. However, with specified lengths of 6, the optimization times suddenly take long, and, in the cases in which we have been able to find an optimal solution, there are now quite large differences in the function values of the relaxed- and optimal solutions. Hence, we see how the problems suddenly have become difficult to solve.

From Tests 1–4 we conclude that there are few non-integers in the relaxed solution, which is an easy solution to find. However, these few variables alter the difficulty of the problem drastically. In particular, when the feasible set of solutions is small, the increase in difficulty is at its largest.

5.5 Experimenting with Decomposition

The DecompositionOverlap algorithm presented in Chapter 4 is intended to make the problems easier to solve. In the following, we compare different parameter choices of the number of regions D and the size of the overlap O. The tests (numbered Tests I–IV) are listed in Table 5.3. The size of the overlap denotes how wide the overlapping part is measured in nodes.

Test	*Regions*	*Overlap*
I	2	3
II	2	6
II	4	3
IV	4	6

Table 5.3: Tests with different parameter choices.

These settings are tested on Test 2 from Table 5.2, i.e., the problem of finding the minimal average connecting path length with node disjoint connecting paths of approximately equal lengths $\rho = 0.2$, $\rho = 0.5$, and $\rho = 0.8$. In all tests, we re-solve the problem until there are no more sub-tours in the solution. We use the following notation:

- empty fields are missing values; and
- \- marks an infeasible solution.

Furthermore, the following values are registered:

$T(s)$	:	optimization time to calculate solution,
it	:	iterations to calculate solution,
$\|\Delta\|$	:	times problem was re-solved, and
f	:	function value of solution.

We give the values for each different region and the sum for all regions. The function value is the value obtained in the first region. All results are listed in Appendix C. In this chapter, we have extracted some results that give a good comprehension of the DecompositionOverlap algorithm. The results are listed in Tables 5.8–5.10 in Section 5.7.2.

Table 5.8 gives an overview of whether a parameter choice entails feasible- or infeasible solutions. In the cases with an infeasible solution, we have marked in which region infeasibility appeared. Furthermore, when the problem was feasible, we have denoted the optimization time. The last column marks in which test cases we were able to find a feasible solution: 'x' marks feasible and '-' marks infeasible.

In all cases in which we find feasible solutions with an overlap of 3, we find feasible solutions with an overlap of 6 but it will usually take longer with an overlap of 6. Hence, with an increased problem size, the optimization will be slower. There are also cases in which no feasible solution is found with an overlap of 3 but a feasible solution is found with an overlap of 6. Hence, we increase the feasible set of solutions by increasing the overlap.

Furthermore, we see that the optimization times are usually the same for a decomposition into 2 or 4 regions, when using an overlap of 3 nodes. However, with an overlap of 6 nodes it is very much faster to use 4 regions instead of 2. If we compare solution times with Test 2 without decomposition, we see that for the largest test problem (test problem a), the decomposition algorithm shows impressive optimization times for both margins of 50% and 80%. Recall that these times also include sub-tour eliminations. Hence, we conclude that decomposition improves optimization times but it does not imply that optimization is faster in smaller regions.

We can conclude that increasing the feasible set of solutions by increasing the size of the problem (having an overlap of 6 and not of 3) will slow the optimization but also give better chances of finding a solution. Furthermore, using 4 instead of 2 regions made optimization times faster using an overlap of 6 nodes but did not change optimization times using an overlap of 3 nodes. This illustrates that the behavior of the DecompositionOverlap algorithm is a complex interaction of the two parameters number of regions and size of overlap.

The results listed in Tables 5.9 and 5.10 include the details of different decompositions of Test 2 with a margin of 50%. It is interesting to notice that the number of times the problem is re-solved due to sub-tours are many more in Region 1 than in the following regions. Hence, it might be a good idea to impose more sub-tour constraints from the start in Region 1. In Tests 1–4 there was a correlation between the optimization time and the number of iterations. However, we see that the optimization times of the different regions in these tests (Tests I–IV) is not related to the number of iterations, which implies that some iterations take longer than others. In particular, the iterations in Region 1 take the longest to solve. This could imply that finding approximately equal lengths in which the average connecting path length is not given is more difficult than given an average connecting path length.[8]

5.6 Summary

We use the commercial software MatLab to make a proof of concept that the models from Chapter 2 and the algorithms from Chapter 4 work as supposed. The models are constructed as sparse matrices with sparsity of less than 1%.

We saw that finding a maximal number of arc- or node disjoint connecting paths without criteria was an easy problem to solve. Test 2 showed that stricter criteria (decreasing the feasible set of solutions) entail more difficult problem, and optimization times increased drastically when imposing approximately equal lengths using margins of 20% instead of 80%. Furthermore, the level of difficulty did, in practice, not depend on whether the imposed criteria were less than or greater than specified length criteria but on the value of the specified lengths. Finding node disjoint connecting paths of length greater than 3 was as easy a problem as finding such paths with length less than 10, and the problems were equally difficult with specified lengths of 6. The relaxed problem was an easy problem in all tests. Hence, we concluded that when the feasible set of solutions is small the importance of the few non-integers in the relaxed solution have drastic influence on the difficulty of the problem.

Furthermore, we saw that the DecompositionOverlap algorithm showed different behavior depending on the number of regions and of the overlap. Using a larger overlap increased the feasible set of solutions but also the optimization times. We saw that the number of regions was a more complex parameter showing different results depending on the overlap; there were no big differences in using 2 or 4 regions with an overlap of 3 but big differences with an overlap of 6, in which decomposition into 4 sub-programs was considerably faster to solve. What is more, the behavior of the algorithm, depended also largely on the test problem. When we solved test problem a with node disjoint connecting paths of approximately equal lengths with an 80% margin, we saw that it was almost 30 times faster to use the DecompositionOverlap algorithm than to solve the problem in one piece. What is more, with

[8] Recall that when we use approximately equal lengths, the average connecting path length found in Region 1 is given as average connecting path length to the following regions.

the DecompositionOverlap algorithm, we are guaranteed that there be no sub-tours in the solution.

5.7 Tables

5.7.1 Results from Tests 1–4

#	$T^r(s)$	$T^*(s)$	$T(s)$	f^r	f^*	f	it^r	it^*	it	$\notin \mathbb{Z}$	δf
1_a	2	9	9	15	15	15	1,869	2,818	2,818	0.1%	0%
1_b	1	1	1	7	7	7	600	615	615	1%	0%
1_c	2	4	4	8	8	8	1,351	926	926	1%	0%
1_d	2	4	4	15	15	15	1,537	1,597	1,597	1%	0%
1_e	1	1	1	10	10	10	710	428	428	1%	0%
1_f	1	3	3	10	10	10	324	955	955	0%	0%
1_g	1	1	1	7	7	7	489	216	216	2%	0%

Table 5.4: Test 1. Maximize without length criteria.

#	ρ	$T^r(s)$	$T^*(s)$	$T(s)$	f^r	f^*	f	it^r	it^*	it	$\notin \mathbb{Z}$	δf
2_a	0.2	13	47,243	(518,400)	4.83	7.40	*	4,306	4,603,300	*	0.5%	*
2_b	0.2	3	913	(75,600)	2.00	7.62	*	1,402	244,059	*	1%	*
2_c	0.2	3	346	(75,600)	2.96	6.53	*	1,650	59,230	*	1%	*
2_d	0.2	8	42,983		3.47	6.70		2,519	3,731,460		0.5%	
2_e	0.2	3	17,951		0.51	6.35		1,744	4,300,896		1%	
2_f	0.2	5	213	(518,400)	0.19	4.85	*	3,149	13,392	*	1%	*
2_g	0.2	1	85		3.68	6.64		1,182	15,815		2%	
2_a	0.5	18	15,976	(68,400)	4.83	5.40	*	4,330	21,398	*	0.4%	*
2_b	0.5	2	108		2.00	5.36		1,325	11,607		1%	
2_c	0.5	3	117		2.96	5.75		1,712	3,894		1%	
2_d	0.5	6	1,496		3.47	4.69		2,330	69,142		0.3%	
2_e	0.5	3	143		0.51	3.81		1,822	16,681		1%	
2_f	0.5	6	10		0.19	5.48		3,157	3,757		1%	
2_g	0.5	1	13		3.66	4.48		865	2,165		1%	
2_a	0.8	13	501	(75,600)	4.83	6.09	*	4,243	7,821	*	0.2%	*
2_b	0.8	3	31		2.00	5.63		1,320	4,138		1%	
2_c	0.8	3	4		2.96	5.94		1,623	1,984		0.6%	
2_d	0.8	6	38		3.47	4.36		2,333	4,532		0.3%	
2_e	0.8	3	55	(424,800)	0.51	3.80	*	1,823	4,851	*	1%	*
2_f	0.8	7	11	(180,000)	0.19	4.83	*	3,156	3,852	*	1%	*
2_g	0.8	1	3	755	3.66	5.04	4.44	759	1,157	726,101	1%	21%

Table 5.5: Test 2. Minimize with approximately equal lengths criteria.

#	L	$T^r(s)$	$T^*(s)$	$T(s)$	f^r	f^*	f	it^r	it^*	it	$\notin \mathbb{Z}$	δf
3_a	3	5	7	7	96.0	96.0	96.0	3,231	3,141	3,141	0%	0%
3_b	3	2	2	2	43.0	43.0	43.0	1,910	2,219	2,219	0.2%	0%
3_c	3	1	2	2	54.0	54.0	54.0	1,469	1,511	1,511	0.5%	0%
3_d	3	2	4	4	77.0	77.0	77.0	1,910	2,219	2,219	0.1%	0%
3_e	3	1	1	1	51.0	51.0	51.0	1,102	975	975	0.3%	0%
3_f	3	1	2	2	61.0	61.0	61.0	1,303	1,223	1,223	0.1%	0%
3_g	3	1	1	1	38.1	39.0	39.0	640	587	587	0.9%	2%
3_a	6	17	834	3,106	98.1	131	102	8,357	48,185	347,347	0.3%	4%
3_b	6	2	21	1,235	44.6	57.0	48.0	2,939	3,737	346,431	0.8%	8%
3_c	6	3	6	855	54.0	72.0	56.0	3,364	3,028	293,850	1%	4%
3_d	6	23	1,495		91.9	126		9,310	80,873		1%	
3_e	6	4	477		61.1	100		4,398	30,155		1%	
3_f	6	5	1,253	(75,600)	64.2	87.0	*	4,528	32,392	*	1%	*
3_g	6	2	214	(75,600)	45.1	67.0	*	2,537	18,221	*	2%	*

Table 5.6: Test 3. Minimize with greater than length criteria.

#	L	$T^r(s)$	$T^*(s)$	$T(s)$	f^r	f^*	f	it^r	it^*	it	$\notin \mathbb{Z}$	δf
4_a	6	13	(518,400)	(518,400)	4.83	*	*	4,306	*	*	0.5%	*
4_b	6	1	341	570	43.7	62.0	52.0	1,289	107,009	239,589	0.9%	19%
4_c	6	3	124		56.0	66.0		2,605	14,985		1%	
4_d	6	2	81		77.0	89.0		1,952	4,757		0.2%	
4_e	6	1	2		51.0	59.0		895	1,087		0.4%	
4_f	6	2	24		61.8	78.0		1,898	2,891		0.5%	
4_g	6	1	13,277	13,788	38.0	42.0	42.0	543	14,795,520	14,795,520	0.6%	11%
4_a	10	4	15	15	96.0	96.0	96.0	3,315	3,373	3,373	1%	0%
4_b	10	1	2	2	43.0	43.0	43.0	946	914	914	0.6%	0%
4_c	10	1	2	2	54.0	54.0	54.0	1,378	1,629	1,629	0%	0%
4_d	10	2	3	3	77.0	77.0	77.0	2,052	1,967	1,967	0.2%	0%
4_e	10	1	1	1	51.0	51.0	51.0	822	928	928	0.2%	0%
4_f	10	1	1	1	61.0	61.0	61.0	1,220	1,148	1,148	0.2%	0%
4_g	10	1	1	1	38.0	38.0	38.0	495	501	501	0%	0%

Table 5.7: Test 4. Minimize with less than length criteria.

5.7.2 Extracts of Results from Tests I–IV

		$D = 2$ and $O = 3$		$D = 2$ and $O = 6$		$D = 4$ and $O = 3$		$D = 4$ and $O = 6$		
#	ρ	infeasible	feasible	infeasible	feasible	infeasible	feasible	infeasible	feasible	Feasible?
2_a	0.2	Region 2				Region 4				`- -`
2_b	0.2		6,599		3,861	Region 3			3,475	`xx-x`
2_c	0.2	Region 2		Region 2		Region 3		Region 3		`----`
2_d	0.2	Region 2				Region 2		Region 2		`- --`
2_e	0.2	Region 2				Region 2		Region 2		`- --`
2_f	0.2	Region 2				Region 2				`- -`
2_g	0.2		102		24,657	Region 2		Region 2		`xx--`
2_a	0.5		52		6,517		808			`xxx`
2_b	0.5		180		575		3		29	`xxxx`
2_c	0.5	Region 2			3,401		4		15	`-xxx`
2_d	0.5		583				31			`x x`
2_e	0.5	Region 2			46,622	Region 2				`-x-`
2_f	0.5	Region 2			304,870	Region 3				`-x-`
2_g	0.5		2		1,533		48		21	`xxxx`
2_a	0.8		18		471		10			`xxx`
2_b	0.8		4		151		3			`xxx`
2_c	0.8		9		3,008		5			`xxx`
2_d	0.8		697		2,490	Region 3				`xx-`
2_e	0.8		199		70,491	Region 3				`xx-`
2_f	0.8		9		20,051	Region 3				`xx-`
2_g	0.8		4		281		2		57	`xxxx`

Table 5.8: Test 2. Minimize with approximately equal lengths criteria.

		Region 1			Region 2			Sum			
#	O	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	f	$T(s)$	it	$\|\Delta\|$
2_a	3	24	1,617	5	27	15,091	0	6.44	52	16,708	5
2_b	3	180	717	64	1	82	0	6.41	180	799	64
2_c	3	3	1,862	0	-	-	-	6.21	47	2,034	0
2_d	3	578	3,261	7	6	1,416	2	7.40	583	4,677	9
2_e	3	193	4,986	16	-	-	-	4.79	193	5,204	16
2_f	3	95	2,381	17	-	-	-	5.79	285	2,470	17
2_g	3	1	929	0	1	1,192	0	5.13	2	2,121	0
2_a	6	6,081	2,324	59	436	64,607	2	6.44	6,517	66,931	61
2_b	6	574	3,709	109	1	218	0	6.37	575	3,927	109
2_c	6	3,387	5,809	69	14	4,194	2	6.35	3,401	10,003	71
2_d	6										
2_e	6	46,621	15,842	96	1	594	0	6.12	46,622	16,436	96
2_f	6	304,870	9,302	210	1	239	0	8.09	304,870	9,541	210
2_g	6	1,533	3,156	83	1	177	0	5.39	1,533	3,333	83

Table 5.9: Test 2. Minimize with approximately equal lengths of 50%.

		Region 1			Region 2			Region 3			Region 4			Sum			
#	O	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	f	$T(s)$	it	$\|\Delta\|$
2_a	3	3	844	3	2	807	1	1	378	0	801	1,238,553	0	6.60	808	1,240,582	4
2_b	3	2	272	3	1	149	1	1	117	0	1	92	0	8.08	3	630	4
2_c	3	1	187	0	3	321	5	1	474	0	1	402	0	7.00	4	1,384	5
2_d	3	19	5,780	0	1	953	0	6	667	4	5	2,837	0	6.00	31	10,237	4
2_e	3	2	1,198	0	-	-	-	-	-	-	-	-	-	4.26	6,536	1,305	0
2_f	3	5	522	3	2	1,242	0	-	-	-	-	-	-	7.06	9	2,045	3
2_g	3	45	1,248	13	1	270	1	1	558	2	1	22	0	6.11	48	2,098	16
2_a	6																
2_b	6	6	740	3	12	1,130	5	11	427	8	1	87	0	10.7	29	2,384	16
2_c	6	7	500	3	3	821	1	5	1,205	2	1	581	0	7.81	15	3,107	6
2_d	6																
2_e	6																
2_f	6																
2_g	6	14	164	2	6	1,605	4	1	1,445	1	1	57	0	7.17	21	3,271	7

Table 5.10: Test 2. Minimize with approximately equal lengths of 50%.

Chapter 6

Conclusion

In this thesis, we develop a specific type of the IMCF network model: the named IMCF model. It is to the best of our knowledge a new model used to model the problem of finding arc- or node disjoint connecting paths with length- and distribution criteria. Finding arc- or node disjoint connecting paths with length- and distribution criteria are problems within areas such as telecommunication, transportation, and production. As an example, a solution to a transportation problem could be as seen in Figure 6.1. It illustrates the three cheapest roads connecting Paris with Toulouse under the criteria that no two roads intersect; no road takes more than 13 hours; and all roads are within 20% from their average length. This is only one example of the multitude of criteria we can incorporate in the named IMCF model (or the IMCF network model).

Figure 6.1: Solution to a transportation problem.

There are some challenges with the modeling such as sub-tours in the solutions and NP-complete large-scale problems. Algorithms to overcome these challenges are the DisjointPathsCriteria which checks whether there are sufficiently many arc- or node disjoint connecting paths before solving the problem with length- and distribution criteria; the SubTourElimination (a cutting-plane algorithm) that enforces constraints where there arise sub-tours; and the Decomposition and DecompositionOverlap which overcome the challenge of large-scale problems. We recommend the DecompositionOverlap algorithm since it is the most flexible of the two. It represents the directed network in smaller regions thought of as geometric regions. Each region constitutes the directed network of a separately solved sub-problem. The regions overlap which entails that some parts of the large directed network are solved more than once. However, this is also that which gives flexibility to the algorithm. Having overlapping regions is made possible by using the named IMCF model.

Using the DecompositionOverlap algorithm, we cannot guarantee an optimal solution since choices are made. In particular, the more sub-programs, the more limitations to the feasible set of solutions. However, it is rarely possible to achieve optimality due to a limited amount of memory; hence, this gives more motivation to use the DecompositionOverlap algorithm.

The level of difficulty depends in theory on the imposed criteria. It should be an easier problem to find arc- or node disjoint connecting paths with less than or equal to specified length criteria than to find such paths with e.g. approximately equal to or greater than specified length criteria. However, in practice this does not hold. Finding node disjoint connecting paths of connecting path lengths greater than 3 is as easy a problem to solve as that of connecting path lengths smaller than 10. It seems that it depends more on the size of the feasible set of solutions. For instance, the problem of finding node disjoint connecting paths of connecting path lengths greater than (or smaller than) 6 has feasible solutions as sub-sets to the feasible solutions of the above mentioned problems, and they are two difficult problems. Another test result that confirmed this is finding node disjoint connecting paths with approximately equal lengths. Here, imposing a narrow margin (of 20%) is a much more difficult problem than with a wider margin (of 80%). Again, the feasible solutions to the difficult problem (with a margin of 20%) is a sub-set of the feasible solutions to the less difficult problem (with a margin of 80%).

Finding a maximal number of arc- or node disjoint connecting paths without criteria proves to be an easy problem to solve. Hence, it is better to check whether there is a sufficient number of connecting paths in the directed network before solving a problem including length criteria. This will not increase optimization time by much but it might instead prevent unnecessary work. The SubTourElimination algorithm works as supposed. It re-solves the problem prohibiting the found sub-tours in the proceeding solution. This sometimes involves many iterations but is always a finite process.

Furthermore, the DecompositionOverlap algorithm gives different results depending on the number of regions, on the size of the overlap, and also largely on the test problem. We have an example in which it is almost 30 times faster to use the DecompositionOverlap algorithm, and in which there are guaranteed to be no sub-tours in the solution. This was for the problem of finding node disjoint connecting paths of approximately equal lengths with a margin of 80%. There are examples of when the DecompositionOverlap algorithm failed to find a feasible solution even though there existed such solutions. However, by altering either the number of regions or the size of the overlap, it is possible to find solutions in most cases. It is therefore our belief that the DecompositionOverlap algorithm is a very flexible and useful algorithm which is able to solve the named IMCF model, a model that can incorporate a great multitude of criteria.

Future Work

The optimization software CPLEX has a variety of parameters. We have always used the default settings, except when finding only a feasible solution. It would be very interesting to see whether changing any of these parameters would improve the solution times.

Furthermore, it would be interesting to test the DecompositionOverlap algorithm on a larger group of test problems, to see how flexible an algorithm it is. These test problems could be from application areas such as VLSI layout, telecommunication, designing train traffic, and possibly other areas.

Appendix A

Existing and Developed Theory

We present some existing theory, and explain why this does not apply to the problems of this thesis. Hereafter, we develop theory that does apply.

A.1 Existing Theory

As mentioned in Chapter 1, we use the model presented in [4, Section 8] as our point of departure. What is more, this paper discusses some properties of the problem of finding arc- and edge disjoint connecting paths in, respectively, directed and undirected networks. These properties include the level of difficulty of finding solutions, and also necessary and sufficient criteria for the existence of a solution. The properties apply to some special types of networks, which are first defined and hereafter discussed.

The number of inward directed arcs at a given node in a directed graph is called the *in-degree*. The *out-degree* is the number of outward directed arcs at a given node in a directed graph. A directed graph is *Eulerian* if and only if it is connected and every node has equal in-degree and out-degree.

An undirected graph is *planar* if it can be drawn in a plane without edges crossing. An undirected graph is planar if and only if it does not contain within it any graph that is a graph expansion of the complete graph K_5 or the utility graph $K_{3,3}$. Figure A.1 illustrates, to the left, the complete graph K_5 and, to the right, the utility graph $K_{3,3}$. Here, a line represents an edge.

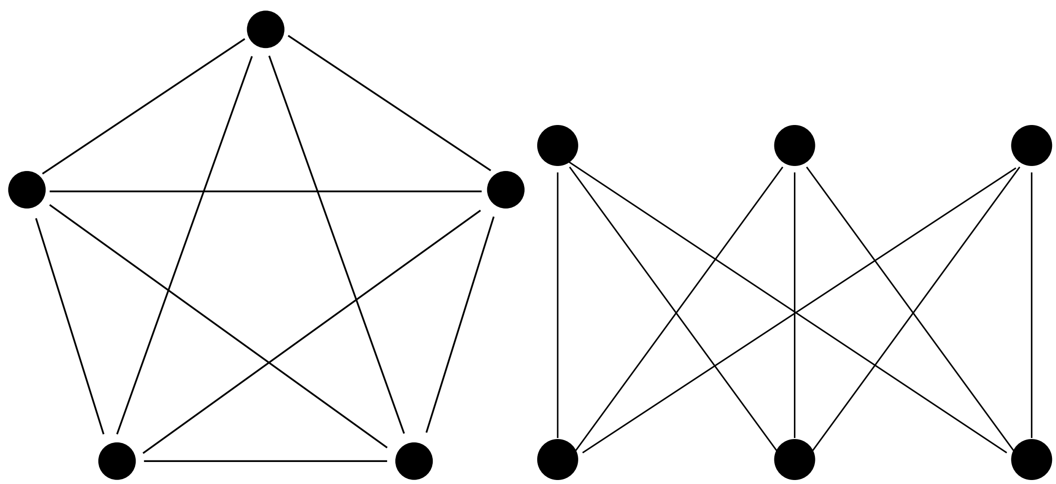

Figure A.1: The complete graph K_5 (left), and the utility graph $K_{3,3}$ (right).

Two operations are allowed when determining whether a graph contains within it an undirected graph that is a graph expansion of K_5 or $K_{3,3}$. These are to delete edges, and merge nodes that are connected by an edge.

An *acyclic directed graph* is a directed graph containing no directed cycles. That is, it cannot contain a directed path in which the initial node equals the final node. Every acyclic directed graph has at least one node of out-degree 0.

With the graphs defined, it is now possible to look at their properties. There are two main categories of problems when discussing their difficulties: one includes problems that in general can be solved in polynomial time; and the other are NP-complete problems that cannot at present. Problems from this second class are considered computationally difficult to solve. For further details, see [9, Section I.5.6].

Two theorems in [4, Section 8] discuss necessary and sufficient criteria for finding arc disjoint connecting paths. However, they make some assumptions of the type of graph: Theorem 8.30 [4] assumes that the directed graph is Eulerian; and Theorem 8.31 [4] assumes that the directed graph is planar and acyclic. Two other interesting theorems are Theorem 8.32 [4] and Theorem 8.35 [4]. The first states that finding arc disjoint connecting paths in an acyclic directed graph can be solved in polynomial time if the number of arc disjoint connecting paths sought for is fixed. The second states what maximum number of arc disjoint connecting paths connecting sub-sources and sub-terminals is possible to find. However, it is only applicable for an Eulerian directed graph.

The graphs of this thesis are not limited to graphs that are either Eulerian, acyclic, or planar. Hence, we cannot apply the described theorems.

A.2 Developed Theory

Theorem 8.29 in [4] states that the arc disjoint connecting paths problem with or without distribution criteria is NP-complete for $K = 2$. Based on this theorem, we prove that the arc disjoint connecting paths problem with or without distribution criteria is NP-complete for any value of K; NP-complete for some length criteria; and that also the node disjoint connecting paths problem is NP-complete; see [14].

Theorem 1. *The arc disjoint connecting paths problem with or without distribution criteria is NP-complete for any fixed value of K, with $K \geq 2$.*

Proof. Supposing we can solve the arc disjoint connecting paths problem with or without distribution criteria for some value of K by some algorithm . This algorithm can then be used to also solve the arc disjoint connecting paths problem with or without distribution criteria in the directed graph (V, A) for $K = 2$ by expanding (V, A) with $K-2$ arcs connecting the source and terminal. As finding arc disjoint connecting paths problem with or without distribution criteria is NP-complete for $K = 2$, it follows that the above problem is NP-complete too, because it is at least as difficult. $\square$

The following theorem proves that it is still NP-complete to find arc disjoint paths of equal- and approximately equal lengths, or greater than specified lengths. In the proof, we assume that all arc costs are 1. However, we can still apply this proof to directed graphs with different arc costs.

Theorem 2. *The arc disjoint connecting paths problem with or without distribution criteria and with equal- and approximately equal length criteria, or greater than specified length criteria is NP-complete.*

Proof. Supposing we can solve the arc disjoint connecting paths problem with the above criteria by some algorithm. This algorithm can then be used to also solve the arc disjoint connecting paths problem with or without distribution criteria in the directed graph (V, A) for $K = 2$ by altering (V, A) such that for each arc, we add arcs of length $2, 3, \ldots$. Additionally to the internal representations at a node, every node only has either one inward- or one outward adjacent node such that if merging all representations of every node, then we would again have the original graph. This transformation is polynomial in the number of arcs. As finding arc disjoint connecting paths problem with or without distribution criteria is NP-complete for $K = 2$, it follows that the above problem is NP-complete too, because it is at least as difficult. $\square$

Theorem 3. *The arc disjoint connecting paths problem with or without distribution criteria and with equal to or less than specified length criteria L^k is not NP-complete when K is fixed.*

Proof. An algorithm to solve this problem is to make a set of all directed paths of length L^k with any sub-source as its initial node. Then these directed paths are compared to see if there are K arc disjoint connecting paths. This algorithm is polynomial in time. □

Article [7] proves that finding a maximum number of vertex disjoint paths of length L is NP-complete for equal to specified lengths larger than 3, and for less than specified lengths larger than 4. Hence, fixing K alters the difficulty of the problem drastically.

Theorem 4. *The node disjoint connecting paths problem with or without distribution criteria is NP-complete for $K = 2$.*

Proof. Supposing we can solve the node disjoint connecting paths problem with or without distribution criteria for $K = 2$ by some algorithm. This algorithm can then be used to also solve the arc disjoint connecting paths problem with or without distribution criteria in the directed graph (V, A) for $K = 2$ by altering (V, A) such that each node is represented as many times as the sum of the in- and out-degree. All representations of node i are connected with all other representations of node i. This transformation is polynomial in the number of arcs. As finding arc disjoint connecting paths problem with or without distribution criteria is NP-complete for $K = 2$, it follows that the above problem is NP-complete too, because it is at least as difficult. □

All in all, depending on the length criteria this problem is more or less difficult to solve.

Appendix B

Data of France

Table B.1 presents the numbering of cities. The chart of France is seen in Figure B.1. The numbers on the chart correspond to the numbering of the cities in Table B.1. Table B.2 presents the costs, time to travel, distances of the roads of the chart, and corresponding arcs. We only list one direction between two cities because the other includes the same data. All information is obtained from `http://www.viamichelin.com`.

Node	***City***	***Node***	***City***
1	Ablis	20	Montpeiller
2	Amiens	21	Mulhouse
3	Angers	22	Nancy
4	Auxerre	23	Nantes
5	Bayonne	24	Nice
6	Besançon	25	Nîmes
7	Bordeaux	26	Niort
8	Brest	27	Orléans
9	Brive-la-Gaillarde	28	Paris
10	Caen	29	Reims
11	Clermont-Ferrand	30	Rennes
12	Dijon	31	Rouen
13	Grenoble	32	Sens
14	Langres	33	Sisteron
15	Le Mans	34	Strasbourg
16	Lille	35	Toulouse
17	Lyon	36	Tours
18	Marseilles	37	Troyes
19	Metz	38	Vierzon

Table B.1: Numbering of cities.

Figure B.1: Chart of France that the enterprise has.

Connection	*Cost*	*Time*	*Distance*	*Arc*
Ablis→Le Mans:	€20.08	1h25	150 km	(1,15)
Ablis→Orléans:	€9.37	0h50	76 km	(1,27)
Ablis→Paris:	€6.79	0h53	66 km	(1,28)
Amiens→Paris:	€14.45	1h37	133 km	(2,28)
Amiens→Rouen:	€12.22	1h14	124 km	(2,31)
Angers→Le Mans:	€11.94	0h58	96 km	(3,15)
Angers→Nantes:	€12.63	1h03	87 km	(3,23)
Angers→Tours:	€12.89	1h32	126 km	(3,36)
Auxerre→Dijon:	€15.97	1h26	150 km	(4,12)
Auxerre→Orléans:	€12.29	1h49	153 km	(4,27)
Auxerre→Paris:	€17.77	1h51	169 km	(4,28)
Auxerre→Sens:	€8.02	0h52	74 km	(4,32)
Bayonne→Bordeaux:	€14.11	2h00	185 km	(5,7)
Bayonne→Toulouse:	€33.54	2h51	297 km	(5,35)
Besançon→Dijon:	€11.30	1h04	98 km	(6,12)
Besançon→Mulhouse:	€16.85	1h31	139 km	(6,21)
Bordeaux→Brive-la-Gaillarde:	€23.06	2h15	196 km	(7,9)
Bordeaux→Niort:	€23.30	2h01	187 km	(7,26)
Bordeaux→Toulouse:	€29.88	2h26	245 km	(7,35)

Brive-la-Gaillarde→Clermont-Ferrand:	€16.58	2h01	166 km	(9,11)
Brive-la-Gaillarde→Toulouse:	€24.43	2h01	200 km	(9,35)
Brive-la-Gaillarde→Vierzon:	€16.35	2h32	272 km	(9,38)
Caen→Rennes:	€11.05	1h50	184 km	(10,30)
Caen→Rouen:	€14.80	1h19	128 km	(10,31)
Clermont-Ferrand→Lyon:	€18.00	2h05	177 km	(11,17)
Clermont-Ferrand→Montpeiller:	€24.95	3h15	334 km	(11,20)
Clermont-Ferrand→Vierzon:	€27.37	2h04	216 km	(11,38)
Dijon→Langres:	€7.82	0h55	79 km	(12,14)
Dijon→Lyon:	€23.50	1h57	195 km	(12,17)
Grenoble→Lyon:	€15.10	1h18	107 km	(13,17)
Grenoble→Sisteron:	€10.28	2h01	143 km	(13,33)
Langres→Nancy:	€14.46	1h32	141 km	(14,22)
Langres→Troyes:	€13.95	1h17	123 km	(14,37)
Le Mans→Rouen:	€15.03	2h41	204 km	(15,31)
Lille→Paris:	€25.85	2h21	221 km	(16,28)
Lille→Reims:	€24.59	2h03	200 km	(16,29)
Lyon→Nîmes:	€32.58	2h29	258 km	(17,25)
Marseilles→Nice:	€25.02	2h14	194 km	(18,24)
Marseilles→Nîmes:	€12.27	1h22	123 km	(18,25)
Marseilles→Sisteron:	€15.45	1h23	132 km	(18,33)
Metz→Nancy:	€3.46	0h42	58 km	(19,22)
Metz→Reims:	€23.10	1h48	192 km	(19,29)
Metz→Strasbourg:	€20.43	1h37	164 km	(19,34)
Montpeiller→Nîmes:	€5.74	0h43	54 km	(20,25)
Montpeiller→Toulouse:	€32.33	2h22	249 km	(20,35)
Mulhouse→Strasbourg:	€6.95	1h15	116 km	(21,34)
Nantes→Niort:	€16.48	1h37	145 km	(23,26)
Nantes→Rennes:	€6.57	1h21	110 km	(23,30)
Nice→Sisteron:	€12.02	2h45	180 km	(24,33)
Niort→Tours:	€24.54	1h43	174 km	(26,36)
Orléans→Tours:	€16.52	1h11	117 km	(27,36)
Orléans→Vierzon:	€11.32	0h55	87 km	(27,38)
Paris→Reims:	€17.31	1h28	143 km	(28,29)
Paris→Rouen:	€19.22	1h35	132 km	(28,31)
Paris→Sens:	€12.23	1h24	125 km	(28,32)
Reims→Troyes:	€15.39	1h21	125 km	(29,37)
Sens→Troyes:	€5.90	0h53	67 km	(32,37)
Tours→Vierzon:	€11.73	1h28	124 km	(36,38)

Table B.2: Costs, time to travel, distances, and arcs.

Appendix C

Test Problems and Results

C.1 Test Problems

The test problems used in this report are listed in Table C.1. *Rows* and *Columns* are the number of rows and columns, respectively, in the constraint matrix used to model the problem of finding a maximum of node disjoint connecting paths.

#	$\|V\|$	$\|A\|$	K	*Rows*	*Columns*
a	596	2,628	15	12,211	40,770
b	365	1,576	7	4,317	11,326
c	365	1,578	8	4,725	13,008
d	332	1,444	15	7,193	23,010
e	266	1,138	10	4,156	11,980
f	354	1,540	10	5,457	16,000
g	218	920	7	2,729	6,734

Table C.1: The test problems used in this report.

We see that test problem a is the largest test problem followed by test problem d, whereas test problem g is the smallest.

C.2 Results from Tests I-IV

Tables C.2–C.5 gives the results from testing the DecompositionOverlap algorithm.

			Region 1			Region 2			Sum			
#	ρ	O	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	f	$T(s)$	it	$\|\Delta\|$
2_a	0.2	3	3,467	6,509	3	-	-	-	6.11	3,519	7,404	3
2_b	0.2	3	6,599	1,448	122	1	55	0	6.91	6,599	1,503	122
2_c	0.2	3	5	1,864	1	-	-	-	6.04	6	2,043	1
2_d	0.2	3	1,603	9,254	2	-	-	-	5.90	26,891	12,477	2
2_e	0.2	3	2,524	170,795	5	-	-	-	5.89	2,525	170,937	5
2_f	0.2	3	8,377	54,816	11	-	-	-	6.16	8,378	55,074	11
2_g	0.2	3	101	5,561	3	1	515	0	6.80	102	6,076	3
2_a	0.5	3	24	1,617	5	27	15,091	0	6.44	52	16,708	5
2_b	0.5	3	180	717	64	1	82	0	6.41	180	799	64
2_c	0.5	3	3	1,862	0	-	-	-	6.21	47	2,034	0
2_d	0.5	3	578	3,261	7	6	1,416	2	7.40	583	4,677	9
2_e	0.5	3	193	4,986	16	-	-	-	4.79	193	5,204	16
2_f	0.5	3	95	2,381	17	-	-	-	5.79	285	2,470	17
2_g	0.5	3	1	929	0	1	1,192	0	5.13	2	2,121	0
2_a	0.8	3	10	1,472	4	7	3,125	0	6.00	18	4,597	4
2_b	0.8	3	4	818	2	1	134	0	6.87	4	952	2
2_c	0.8	3	8	2,052	3	1	885	0	5.87	9	2,937	3
2_d	0.8	3	690	3,631	21	7	5,119	0	5.60	697	8,750	21
2_e	0.8	3	198	5,115	52	1	543	0	5.27	199	5,658	52
2_f	0.8	3	9	2,321	1	1	266	0	5.79	9	2,587	1
2_g	0.8	3	4	945	3	1	220	0	5.13	4	1,165	3

Table C.2: Test 2; minimizing with approximately equal lengths.

			Region 1			Region 2			Region 3			Region 4			Sum			
#	ρ	O	$T(s)$	it	$\lvert\Delta\rvert$	$T(s)$	it	$\lvert\Delta\rvert$	$T(s)$	it	$\lvert\Delta\rvert$	$T(s)$	it	$\lvert\Delta\rvert$	f	$T(s)$	it	$\lvert\Delta\rvert$
2_a	0.2	3	2	1,178	0	1	579	0	22	21,942	1	-	-	-	6.00	27	24,521	1
2_b	0.2	3	3	269	4	1	956	1	-	-	-	-	-	-	6.15	5	1,281	5
2_c	0.2	3	2	140	3	83	4,184	10	-	-	-	-	-	-	5.67	85	4,402	13
2_d	0.2	3	37	1,721	8	-	-	-	-	-	-	-	-	-	4.40	38	1,771	8
2_e	0.2	3	78	4,578	4	-	-	-	-	-	-	-	-	-	4.43	79	4,775	4
2_f	0.2	3	125	3,524	6	-	-	-	-	-	-	-	-	-	5.43	156	4,108	6
2_g	0.2	3	145	127	17	-	-	-	-	-	-	-	-	-	5.44	146	180	17
2_a	0.5	3	3	844	3	2	807	1	1	378	0	801	1,238,553	0	6.60	808	1,240,582	4
2_b	0.5	3	2	272	3	1	149	1	1	117	0	1	92	0	8.08	3	630	4
2_c	0.5	3	1	187	0	3	321	5	1	474	0	1	402	0	7.00	4	1,384	5
2_d	0.5	3	19	5,780	0	1	953	0	6	667	4	5	2,837	0	6.00	31	10,237	4
2_e	0.5	3	2	1,198	0	-	-	-	-	-	-	-	-	-	4.26	6,536	1,305	0
2_f	0.5	3	5	522	3	2	1,242	0	-	-	-	-	-	-	7.06	9	2,045	3
2_g	0.5	3	45	1,248	13	1	270	1	1	558	2	1	22	0	6.11	48	2,098	16
2_a	0.8	3	6	860	5	1	512	1	2	573	1	1	855	0	6.00	10	2,800	7
2_b	0.8	3	1	360	0	2	141	5	2	117	0	1	92	0	9.15	3	710	5
2_c	0.8	3	2	114	6	1	271	0	2	749	3	1	478	0	7.67	5	1,612	9
2_d	0.8	3	42	1,421	8	1	578	0	-	-	-	-	-	-	11.7	43	1,999	8
2_e	0.8	3	43	2,589	5	17	752	1	-	-	-	-	-	-	6.85	45	3,868	6
2_f	0.8	3	238	1,803	17	5	1,403	1	-	-	-	-	-	-	7.35	243	3,877	18
2_g	0.8	3	1	230	0	1	204	0	1	146	0	1	27	0	8.44	2	607	0

Table C.3: Test 2; minimize with approximately equal lengths criteria.

Table C.4: Test 2; minimizing with approximately equal lengths.

			Region 1			Region 2			Sum			
#	ρ	O	$T(s)$	it	$\lvert\Delta\rvert$	$T(s)$	it	$\lvert\Delta\rvert$	f	$T(s)$	it	$\lvert\Delta\rvert$
2_a	0.2	6										
2_b	0.2	6	3,861	5,588	59	1	458	0	6.04	3,861	6,046	59
2_c	0.2	6	30,051	599,505	27	-	-	-	6.21	30,052	599,745	27
2_d	0.2	6										
2_e	0.2	6										
2_f	0.2	6										
2_g	0.2	6	24,657	5,036	87	1	184	0	6.78	24,657	5,220	87
2_a	0.5	6	6,081	2,324	59	436	64,607	2	6.44	6,517	66,931	61
2_b	0.5	6	574	3,709	109	1	218	0	6.37	575	3,927	109
2_c	0.5	6	3,387	5,809	69	14	4,194	2	6.35	3,401	10,003	71
2_d	0.5	6										
2_e	0.5	6	46,621	15,842	96	1	594	0	6.12	46,622	16,436	96
2_f	0.5	6	304,870	9,302	210	1	239	0	8.09	304,870	9,541	210
2_g	0.5	6	1,533	3,156	83	1	177	0	5.39	1,533	3,333	83
2_a	0.8	6	391	2,605	21	80	32,950	1	6.04	471	35,555	22
2_b	0.8	6	151	3,969	45	1	227	0	6.87	151	4,196	45
2_c	0.8	6	3,005	6,062	79	3	570	1	8.27	3,008	6,632	80
2_d	0.8	6	2,486	26,919	10	4	1,518	0	5.83	2,490	28,437	10
2_e	0.8	6	70,490	14,852	169	1	497	0	8.12	70,491	15,349	169
2_f	0.8	6	20,050	7,539	69	1	186	0	7.87	20,051	7,725	69
2_g	0.8	6	281	3,387	29	1	178	0	6.89	281	3,565	29

			Region 1			Region 2			Region 3			Region 4			Sum			
#	ρ	O	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	$T(s)$	it	$\|\Delta\|$	f	$T(s)$	it	$\|\Delta\|$
2_a	0.2	6																
2_b	0.2	6	3,308	4,305	52	159	7,753	18	8	4,522	2	1	102	0	6.12	3,475	16,682	72
2_c	0.2	6	151	596	37	14,835	3,022	53	-	-	-	-	-	-	5.62	14,987	4,058	90
2_d	0.2	6	35,023	99,683	108	-	-	-	-	-		-	-	-	4.43	35,025	100,072	108
2_e	0.2	6	38,390	2,047	72	-	-	-	-	-	-	-	-	-	4.35	38,393	2,907	72
2_f	0.2	6																
2_g	0.2	6	1,546	6,353	42	-	-	-	-	-	-	-	-	-	5.00	1,602	6,631	42
2_a	0.5	6																
2_b	0.5	6	6	740	3	12	1,130	5	11	427	8	1	87	0	10.7	29	2,384	16
2_c	0.5	6	7	500	3	3	821	1	5	1,205	2	1	581	0	7.81	15	3,107	6
2_d	0.5	6																
2_e	0.5	6																
2_f	0.5	6																
2_g	0.5	6	14	164	2	6	1,605	4	1	1,445	1	1	57	0	7.17	21	3,271	7
2_a	0.8	6																
2_b	0.8	6																
2_c	0.8	6																
2_d	0.8	6																
2_e	0.8	6																
2_f	0.8	6																
2_g	0.8	6	9	291	6	48	553	21	1	282	1	1	84	0	6.17	57	1,210	28

Table C.5: Test 2; minimize with approximately equal lengths criteria.

Bibliography

[1] M. O. Ball, T. L. Magnanti, and C. L. Monma, editors. *Network models*, volume 7 of *Handbooks in Operations Research and Management Science*. North-Holland Publishing Co., Amsterdam, 1995.

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

[3] Ricardo Fukasawa, Marcus V. Poggi de Aragao, Oscar Porto, and Eduardo Uchoa. Solving the freight car flow problem to optimality. In *Electronic notes in theoretical computer science, 66*, volume 66 of *Electron. Notes Theor. Comput. Sci.*, page 11 pp. (electronic). Elsevier, Amsterdam, 2002.

[4] R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of combinatorics. Vol. 1, 2.* Elsevier Science B.V., Amsterdam, 1995.

[5] E. Hadjiconstantinou and N. Christofides. An efficient implementation of an algorithm for finding K shortest simple paths. *Networks*, 34(2):88–101, 1999.

[6] Kaj Holmberg and Di Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS J. Comput.*, 15(1):42–57, 2003.

[7] A. Itai, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3):277–286, 1982.

[8] Torbjörn Larsson and Di Yuan. An augmented Lagrangian algorithm for large scale multicommodity routing. *Comput. Optim. Appl.*, 27(2):187–215, 2004.

[9] George Nemhauser and Laurence Wolsey. *Integer and combinatorial optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., New York, 1999. Reprint of the 1988 original, A Wiley-Interscience Publication.

[10] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Dover Publications Inc., Mineola, NY, 1998. Corrected reprint of the 1982 original.

[11] Deepinder Sidhu, Raj Nair, and Shukri Abdallah. Finding disjoint path in networks. Department of Computer Science and Institute for Advanced Computer Science, University of Maryland, Baltimore.

[12] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[13] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest

pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

[14] Carsten Thomassen. Personal communication, June 2005.

[15] Donald M. Topkis. A k shortest path algorithm for adaptive routing in communications networks. *IEEE Trans. Comm.*, 36(7):855–859, 1988.

[16] Spyros Tragoudas and Yaakov L. Varol. Computing disjoint paths with length constraints. In *Graph-theoretic concepts in computer science (Cadenabbia, 1996)*, volume 1197 of *Lecture Notes in Comput. Sci.*, pages 375–389. Springer, Berlin, 1997.

Index