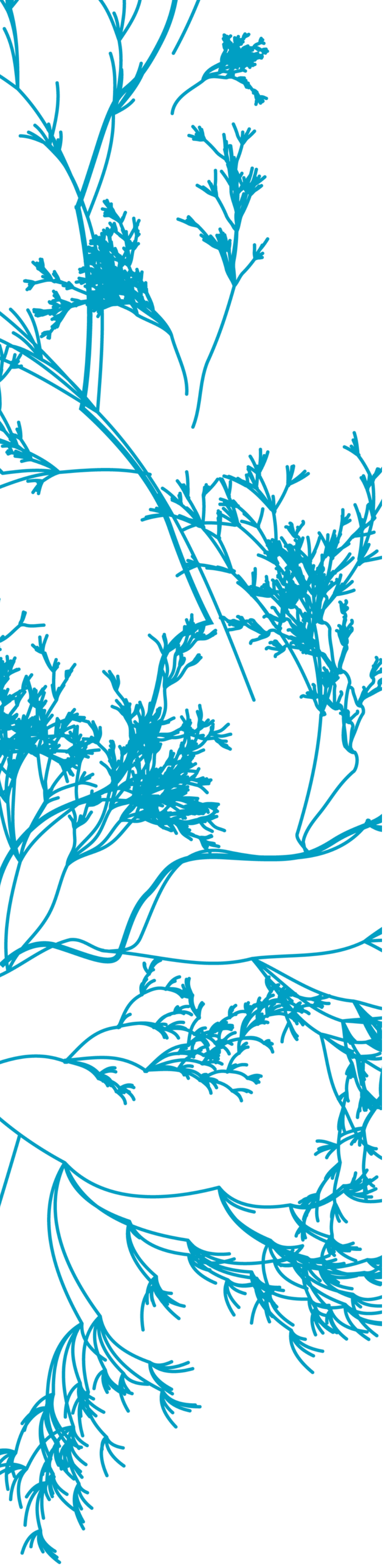
MSc Thesis Applied Mathematics



A 3-stage formulation for solving single modular capacitated network design problems

Cas Sitvast

Graduation committee:
Dr. Ir. W.R.W. Scheinhardt
Prof. Dr. M.J. Uetz
Dr. M. Walter

October 20, 2023

Discrete Mathematics and Mathematical Programming (DMMP)
Mathematics of Operations Research (MOR)
Faculty of EEMCS
University of Twente

UNIVERSITY OF TWENTE.

Abstract

The need for telecommunication and transportation networks is ever increasing. New networks have to be created in order to meet the increasing demand. Due to the complexity of the problem, the methodologies used for developing new networks are slow and inefficient. This thesis aims to construct an efficient method for solving a subset of the network design problems. This method combines a Benders' decomposition with column generation and combinatorial algorithms to construct a formulation of the problem that is easier to solve using a solver like Gurobi. Easier in comparison to a more straightforward formulation. Both formulation are compared on two different sets of example problems. The results show that the new formulation may reduce computation time for some instances.

Keywords- Network design, Benders' decomposition, column generation

Contents

1 Introduction

Networks serve as the backbone of our interconnected world, facilitating the flow of data, resources, and services. Telecommunication and transportation networks as well as supply chains are just a few of the examples in which network design pops up. As the complexity of these networks continue to grow, so do the challenges in their design. This results in optimisation of these systems to become harder and harder, but also more relevant. Therefore the call for fast algorithms solving these kind of problems is also increasing.
Network design problems are not a new concept and have been studied extensively over the past century, with a first introduction by Ford and Fulkerson in 1958 [11]. They introduced the Multi Commodity Network Flow problem (MCNF) in their paper "A suggested computation for maximal multicommodity network flows" [6], of which network design problems are a subset. Within network design problems there are again many different categories, among them is the modular Capacitated Network Design Problem (CNDP), which will also be studied in this thesis. In a CNDP the designer is given a graph and is asked to produce a set of arcs on which capacity has to be installed to route a set of commodities between origin and destination. The goal of the designer is to meet all demands and also minimise the cost. Bienstock et al. [2] however, showed that these problems are NP-hard. Thus finding an algorithm that solves the CNDP in polynomial time is unlikely, unless P=NP. Therefore many researchers instead opt to construct and study algorithms for special cases of the CNDP.
In a standard CNDP, the designer is given a directed graph $G = (V, A)$ in which A are all possible arcs to open. The designer is also given a set of commodities K that have to be transported over the network. Each commodity $k \in K$ is a tuple $(s, t, d) \in V \times V \times \mathbb{R}_{>0}$ of a source node s and target node t between which a certain demand d has to be transported. Opening an arc induces a cost and, depending on the problem, installs a certain capacity on the arc. In the modular case, the designer could also choose between a set of modules, that install additional capacity on an arc against a certain extra cost. Finally sending one unit of a commodity over an arc also induces a routing cost. The goal of the network designer is to construct the network that minimises the total design cost and ensures all demands are satisfied. This thesis will only consider the special case of the CNDP in which the designer can only choose between one module for extra capacity, but an infinite amount of it. This is called the Single Modular Network Design (SMND).
A natural way of solving instances of an SMND is by formulating it as a Mixed Integer Program (MIP), however the resulting program could be extremely large and would take a long time to solve using the standard methods for solving MIPs. Therefore in this thesis a different MIP formulation is posed that also correctly formulates the SMND. In solving this different formulation extra algorithms are used to improve computation time, for which this formulation in particular lends itself. The extra algorithms are combinatorial algorithms, such as Dijkstra's [5] and preflow push-relabel [7], a Benders' decomposition [1] for row generation and column generation [4].

The outline of this thesis will be as follows. First in Section 2 the model formulation is given, followed by two reformulations. In Section 3 the 3-stage formulation is explained. In Section 4 experimental results of the 3-stage formulation are shown. The experiments are done on instance examples of two different libraries. These results are discussed and concluded in Section 5.

2 Model formulation

In this section the two formulations most commonly used in the literature [11] for an SMND will be discussed and analysed. First the compact and straightforward node-arc formulation is given in Section 2.1, followed by a path-based formulation in Section 2.2.

2.1 Compact formulation

An SMND problem can be mathematically modelled as a MIP and a compact formulation is given below, in Table 1 an overview of all symbols and their meaning is given, in Table 2 an overview of variables is given.

Table 1: Overview of symbols in a SMND and their meaning

Symbol	Meaning
V	Set of all vertices/nodes
A	Set of all possible arcs to open
K	Set of all commodities
$c_a \in \mathbb{R}^+$	Cost for opening an arc a
$c_a^m \in \mathbb{R}^+$	Cost for installing a module on arc a
$r_a^k \in \mathbb{R}^+$	Cost for sending one unit of commodity k over arc a
$u_a \in \mathbb{R}^+$	Installed capacity upon opening arc a
$u_a^m \in \mathbb{R}^+$	Modular capacity on arc a
$M_a = \lceil \sum_{k\in K} d_k/u_a^m \rceil$	Maximum number of modules necessary to allow all demand over arc a.
$\mathbf{d} \in \mathbb{R}^{V\times K}$	Demand vector

Table 2: Variables associated with the compact formulation of a SMND.

Variable	Representation
$x_a \in \{0,1\}$	decision variable whether or not to open arc a.
$y_a \in \mathbb{Z}_+$	variable for number of installed modules on arc a.
$f_a^k \geq 0$	the amount of commodity k sent along arc a.

$$\begin{aligned}
\min \quad & \sum_{k\in K}\sum_{a\in A} r_a^k \cdot f_a^k + \sum_{a\in A} c_a \cdot x_a + \sum_{a\in A} c_a^m \cdot y_a & & & \text{(1a)}\\
\text{s.t.} \quad & \sum_{a\in\delta^{\text{in}}(v)} f_a^k - \sum_{a\in\delta^{\text{out}}(v)} f_a^k = \mathbf{d} & & \forall v \in V, \forall k \in K & \text{(1b)}\\
& \sum_{k\in K} f_a^k - u_a \cdot x_a - u_a^m \cdot y_a \leq 0 & & \forall a \in A & \text{(1c)}\\
& M_a \cdot x_a - y_a \leq 0 & & \forall a \in A & \text{(1d)}\\
& f_a^k \geq 0 & & \forall a \in A, \forall k \in K & \text{(1e)}\\
& x_a \in \{0,1\} & & \forall a \in A & \text{(1f)}\\
& y_a \in \mathbb{Z}^+ & & \forall a \in A & \text{(1g)}
\end{aligned}$$

The demand vector is for a tuple (v,k) defined as $-d$ if $v = s_k$, d if $v = t_k$ and zero otherwise. Then Constraints (1b) can be recognised as the flow conservation equations, which ensure that all demands are satisfied. Constraint (1c) ensures that the total flow on an arc can not exceed the capacity of the arc, Constraint (1d) ensures that no modules can be installed on an arc that has not been opened, Constraint (1e) ensures that all flow is non-negative and Constraints (1f) and (1g) ensure the necessary domain for the x and y variables. Hence this formulation correctly describes the SMND.
This formulation has $(2+|K|)|A|$ variables and (not counting Constraints (1e), (1f) and (1g)) $2|A|+|V||K|$ many constraints. Intuitively however most of the f_a^k are zero in an (optimal) basic solution. The reader may realise this, since in practical cases a commodity will probably not be sent over the entire network, but rather over a small part of it. In fact Lemma 1 provides an upper-bound on the number of positive flow variables in any basic solution. Define $N(x) := |\{i : x_i > 0\}|$ to be the number of positive entries of a vector.

Lemma 1. *Given a (directed) graph $D = (V, A)$ and a SMND in compact form defined on this graph. If a feasible solution to (1) exists, then for every basic (feasible) solution f to (1) it holds that $N(f) \leq |V||K| + |A|$.*

Proof. Recall that if a feasible solution exist, then a basic solution also exists. Let (f^*, x^*) be a basic solution to (1). Also U_1 be the matrix such that $U_1 f = \mathbf{d}$ represent the flow conservation constraints (1b). Let U_2 and C be the matrices such that $U_2 f = -C\mathbf{x}^*$ represent Constraints (1c), (1d) and let $U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$. Introduce slack variables $\sigma \geq 0$ for Constraints (1c), (1d) such that they become equality constraints. This allows for a formulation in standard form (2).

$$\min \ \mathbf{r}^T f \tag{2a}$$

$$\text{s.t.} \ U \begin{bmatrix} f \\ \sigma \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ -C\mathbf{x}^* \end{bmatrix} \tag{2b}$$

For some $\hat{\sigma}$ and some (Simplex) basis B of U, $\begin{bmatrix} f^* \\ \hat{\sigma} \end{bmatrix}$ is a basic solution to (2). The number of positive variables in $\begin{bmatrix} f^* \\ \hat{\sigma} \end{bmatrix}$ depends on the rank of B, since it is bounded from above by the number of basic variables. Since B is a basis of U, $\text{rank}(B) \leq \text{rank}(U) \leq |V||K| + |A|$, the number of constraints in formulation (2). Finally $N(f^*) \leq N(\begin{bmatrix} f^* \\ \hat{\sigma} \end{bmatrix}) \leq |V||K| + |A|$. □

This result gives insight into the ratio (3) of nonzero variables in an optimal solution by considering cases 1-4.

$$R := \frac{|V||K| + |A|}{(2 + |K|)|A|} \tag{3}$$

1. $|A| \in \mathrm{O}(|V|)$, $|V| \to \infty$, $R \to \dfrac{|K| + 1}{|K| + 2}$
2. $|A| \in \mathrm{O}(|V|)$, $|K| \to \infty$, $R \to 1$
3. $|A| \in \mathrm{O}(|V|^2)$, $|V| \to \infty$, $R \to \dfrac{1}{|K| + 2}$
4. $|A| \in \mathrm{O}(|V|^2)$, $|K| \to \infty$, $R \to \dfrac{|V|}{(|V| - 1)^2}$

This shows that if there are relatively few arcs in comparison to the number of vertices or commodities, most arcs also participate in a basis for an optimal solution. While on the other hand when there are many arcs, only few of the arcs are used.

2.2 Path-based formulation

The result obtained in Lemma 1 leads to another observation. If the number of positive flow variables is low then the number of distinct paths over which flow is sent for a commodity is also low. This is expressed in Theorem 1, which gives an upper bound on the total number of positive flow-paths. A formulation given the right paths could then result in a MIP with fewer variables and constraints.
Let $\mathcal{P}$ denote the set of all simple paths in the graph, that is, all paths that use an arc at most ones. Also let $\mathcal{D}(f) \in \mathbb{R}^{\mathcal{P}}$ denote a flow decomposition vector of f, i.e. every index uniquely corresponds to a path within the graph and the sum of the entries equals the total flow of f.

Theorem 1. *If formulation (1) is feasible, then a basic solution f^* and a decomposition $\mathcal{D}(f^*)$ of this flow exist such that $N(\mathcal{D}(f^*)) \leq |V||K| + |A|$.*

Proof. Using Lemma 1, a basic solution f to formulation (1) exists with $N(f) \leq |V||K| + |A|$. Let $A_k := \{a \in A | f_a^k > 0\}$, $D^k = (V, A_k)$ and $f_{A_k} := \{f_a^k | a \in A_k\}$. Then the flow decomposition theorem [12] states that on D^k there exists a decomposition $\mathcal{D}_k(f_{A_k})$ such that $N(\mathcal{D}_k(f_{A_k})) \leq |A_k|$. In words this means that the total flow can be separated in less than $|A_k|$ distinct smaller flows. Clearly $\sum_{k \in K} \mathcal{D}_k(f_{A_k}) \leq \sum_{k \in K} |A_k| = N(f) \leq |V||K| + |A|$. . □

In the path-based formulation the flow variables per arc are replaced with flow variables per path the meaning of the various symbols and variables are depicted in Table 3 and Table 4 respectively.

Table 3: Overview of symbols in a path-based SMND and their meaning

Symbol	Meaning
V	Set of all vertices/nodes
A	Set of all possible arcs on which lines can be opened
K	Set of all commodities
$\mathcal{P}_k$	Set of all possible source-target paths of commodity k
$c_a \in \mathbb{R}^+$	cost for opening an arc a
$c_a^m \in \mathbb{R}^+$	cost for installing a module on arc a
$r_a^k \in \mathbb{R}^+$	cost for sending one unit of commodity k over arc a
$r_P^k = \sum_{a \in P} r_a^k$	total cost for sending one unit of commodity k over path P
$u_a \in \mathbb{R}^+$	installed capacity upon opening arc a
$u_a^m \in \mathbb{R}^+$	modular capacity on arc a

Table 4: Variables associated in a path-based formulation of a SMND

Variable	Representation
$x_a \in \{0,1\}$	decision variable whether or not to open arc a.
$y_a \in \mathbb{Z}_+$	variable for number of installed modules on arc a.
$\lambda_P^k \geq 0$	the amount of commodity k sent over path $P \in \mathcal{P}^k$.

$$\begin{aligned}
\min \quad & \sum_{k \in K} \sum_{P \in \mathcal{P}_k} r_P^k \cdot \lambda_P^k + \sum_{a \in A} c_a \cdot x_a + c_a^m \cdot y_a && && \text{(4a)}\\
\text{s.t.} \quad & \sum_{p \in \mathcal{P}_k} \lambda_P^k = d_k && \forall k \in K && \text{(4b)}\\
& \sum_{k \in K} \sum_{p \in \mathcal{P}_k | a \in P} \lambda_P^k - u_a^i \cdot x_a - u_a^m \cdot y_a \leq 0 && \forall a \in A && \text{(4c)}\\
& M_a \cdot x_a - y_a \geq 0 && \forall a \in A && \text{(4d)}\\
& \lambda_P^k \geq 0 && \forall k \in K, \forall P \in \mathcal{P}_k && \text{(4e)}\\
& x_a \in \{0,1\} && \forall a \in A && \text{(4f)}\\
& y_a \in \mathbb{Z}^+ && \forall a \in A && \text{(4g)}
\end{aligned}$$

In this formulation, Constraints (4b) ensure that all demands are satisfied and Constraints (4c) ensure that the total flow over an arc does not exceed the available capacity on that arc. Constraints (4d) are again the linking constraints and Constraints (4e)-(4g) ensure the necessary requirements for the variables. Hence this formulation is also a correct formulation for the SMND.

2.3 Benders' reformulation

A common approach for solving MIPs and for (sometimes) improving computation time is by doing a Benders' decomposition [1]. In a Benders' decomposition the integer variables are separated from the continuous variables by projection. By solving the resulting projected integer program (IP) and iteratively cutting off solutions that result in infeasible solutions to the continuous LP an optimal solution can also be found. In this section a Benders' decomposition is done on formulation (4), which is a good candidate due to its intuitive separation of integer and continuous variables. The separation results in two mathematical programs, one IP and one LP. The IP will decide the capacity installed on the arcs and the LP decides how much flow is sent over the resulting paths.
For the Benders' reformulation a new variable $w_k \in \mathbb{R}$ is introduced for every commodity $k \in K$. This variable represents the total contribution of a commodity on the objective. Changing the objective to (5a) and adding Constraints (5e) yields an equally correct formulation of the problem.

$$
\begin{aligned}
\min \quad & \sum_{k\in K} w_k + \sum_{a\in A}(c_a \cdot x_a + c^m_a \cdot y_a) && && \text{(5a)}\\
\text{s.t.} \quad & \sum_{p\in\mathcal{P}_k} \lambda^k_P = d_k && \forall k \in K && \text{(5b)}\\
& \sum_{k\in K}\sum_{p\in\mathcal{P}_k | a\in P} \lambda^k_P - u^i_a \cdot x_a - u^m_a \cdot y_a \leq 0 && \forall a \in A && \text{(5c)}\\
& M_a \cdot x_a - y_a \geq 0 && \forall a \in A && \text{(5d)}\\
& w_k - \sum_{P\in\mathcal{P}_k} (\sum_{a\in P} r^k_a)\lambda^k_P \geq 0 && \forall k \in K && \text{(5e)}\\
& \lambda^k_P \geq 0 && \forall k \in K, \forall P \in \mathcal{P}_k && \text{(5f)}\\
& x_a \in \{0,1\} && \forall a \in A && \text{(5g)}\\
& y_a \in \mathbb{Z}^+ && \forall a \in A && \text{(5h)}
\end{aligned}
$$

Next the Benders' projection is done. The variables λ^k_P are projected onto the variables x_a, y_a, w_k. Let $\sigma_k, -\Pi_a, \mu_k$ be dual multipliers for Constraints (5b), (5c), (5e) respectively. This results in the "Master LP" (6)

$$
\begin{aligned}
\min \quad & \sum_{k\in[K]} w_k + \sum_{a\in A} c_a \cdot x_a + \sum_{a\in A} c^m_a \cdot y_a && && \text{(6a)}\\
\text{s.t.} \quad & \sum_{k} \mu_k w_k + \sum_{a\in A} \Pi_a u^i_a \cdot x_a + \sum_{a\in A} \Pi_a u^m_a \cdot y_a \geq \sum_k d_k \sigma_k && \forall (\sigma_k, \Pi_a, \mu_k) \in \mathcal{C} && \text{(6b)}\\
& M_a \cdot x_a - y_a \geq 0 && \forall a \in A && \text{(6c)}\\
& w_k \geq 0 && \forall k \in K && \text{(6d)}\\
& x_a \in \{0,1\} && \forall a \in A && \text{(6e)}\\
& y_a \in \mathbb{Z}^+ && \forall a \in A && \text{(6f)}
\end{aligned}
$$

$\mathcal{C}$ is the projection cone defined by the Benders' reformulation and is defined as below:

$$
\begin{aligned}
\mathcal{C} \quad & \sigma_k - \sum_{a\in P} \left(r^k_a \mu_k + \Pi_a\right) \leq 0 && \forall P \in \mathcal{P}_k, \forall k \in K && \text{(7a)}\\
& \Pi_a \geq 0 && \forall a \in A && \text{(7b)}\\
& \sigma_k, \mu_k \geq 0 && \forall k \in K && \text{(7c)}
\end{aligned}
$$

An optimal solution to the Master LP also yields an optimal solution to the compact formulation. The Master LP has infinitely many constraints of (6b). It can be shown that only extreme rays yield non-redundant inequalities, which implies that only finitely many inequalities are needed to describe the feasible solution space of the Master LP. However, since the number of inequalities would still be too large, the restricted master problem (RMP) is considered instead. In the RMP only a subset of the total constraints are present.
Feasibility of a solution to the RMP $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{w}})$ can then be computed by solving the "Lifting LP" (8). Optimality of the Master LP is achieved when the objective value (8a) is less or equal to the value of the projected variables $\sum_{k\in K} \hat{w}_k$.

$$
\begin{aligned}
\min \quad & \sum_{k\in K}\sum_{P\in\mathcal{P}_k} r^k_P \cdot \lambda^k_P && && \text{(8a)}\\
\text{s.t.} \quad & \sum_{k\in K}\sum_{P\in\mathcal{P}_k} \lambda^k_P = d_k && \forall k \in K && \text{(8b)}\\
& \sum_{k\in K}\sum_{P\in\mathcal{P}_k | a\in P} \lambda^k_P \leq u^i_a \cdot \hat{x}_a + u^m_a \cdot \hat{y}_a && \forall a \in A && \text{(8c)}\\
& \lambda^k_P \geq 0 && \forall P \in \mathcal{P}_k, \forall k \in K && \text{(8d)}
\end{aligned}
$$

As mentioned finding all constraints that satisfy (6b) will take a significant amount of time. So instead these constraints are generated iteratively. Using dual multipliers of the Lifting LP (10), two constraints can be generated: feasibility and optimality constraints. In a feasibility constraint the dual multipliers of the substitute variable w_k are 0, and are generated only when the Lifting LP is infeasible, while in an optimality constraint these multipliers are 1.
The observant reader may have noticed that the formulation of the Lifting LP uses all paths between a source and target node. As there usually are many paths connecting two nodes in a graph, this formulation is often much larger compared to the compact formulation in terms of model size. However, this is where column generation [4] will play a role. Instead of the Lifting LP with all paths, the Restricted Lifting Problem (RLP) with a subset of the paths is considered. In Section 3.2 this will be explained in more detail.
This Benders' decomposition already gives rise to an intuitive algorithm, the generation of the constraints. A subset of these constraints can also be obtained by using combinatorial algorithms as shown in [3]. Instead of solving the Lifting LP with all paths/variables this can instead be done using column generation . The combination of these methods result in a 3-stage formulation shown in the next section.

3 3-stage formulation

This section explains the details of the 3-stage algorithm. The algorithm consists of an initialisation step explained in Section 3.1 followed by a loop in which two mathematical programs are solved consecutively. These are the RMP and RLP which are derived by doing a Benders' decomposition explained in Section 2.3. For the RLP a column generation step is done to find relevant paths, which is explained in Section 3.2.

Algorithm 1 The 3 stage algorithm

```
$\mathcal{C}$ := Set of constraints for the Master LP
$\overline{P_k}$ := Set of paths found for commodity $k$ thus far

  Initialisation:                                                        ▷ Section 3.1
  for $k \in K$ do $\overline{P_k} \leftarrow$ {shortest $(s_k, t_k)$-path}
  $\mathcal{C} \leftarrow$ Min-Cuts$(\mathbf{0}, \mathbf{0})$                                    ▷ Algorithm 2

  Main Algorithm:
  while True do
      $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{w}} \leftarrow$ Solve RMP with constraints $\mathcal{C}$              ▷ Formulation (6)
      while True do
          $\lambda \leftarrow$ Solve RLP with $\overline{P_1}, \cdots, \overline{P_{|K|}}$                      ▷ Formulation (8)
          if Lifting LP is infeasible then
              if $\exists$ path $P$ with arc lengths $\Pi_a$ s.t. Constraint (7a) is violated then
                  $\overline{P_k} \leftarrow \overline{P_k} \cup P$                                       ▷ Section 3.2
              else
                  Augment $\mathcal{C}$ with feasibility constraint
                  break
          else
              if $\exists$ path $P$ with arc lengths $\Pi_a + r_a^k$ s.t. Constraint (7a) is violated then
                  $\overline{P_k} \leftarrow \overline{P_k} \cup P$                                       ▷ Section 3.2
              else if $\exists k$ s.t. $\sum_{P \in \mathcal{P}_k} r_P^k \cdot \lambda_P^k > \hat{w}_k$ then
                  Augment $\mathcal{C}$ with optimality constraint
                  break
              else
                  STOP
```

3.1 Initialisation

During initialisation of the 3-stage formulation some combinatorial algorithms are used to improve the total computation time of the RMP and RLP. The RMPs computation time is improved by initialising it with pre-determined Benders' constraints as explained in Section 3.1.1. The RLP is explained in Section 3.1.2.

3.1.1 Initialisation of the RMP

Upon initialisation of the RMP, there are no Benders' constraints in the model. To improve computation time of the RMP a subset of these constraints can already be obtained before solving the RMP once. In [3], it was shown that cuts in the graph correspond to a subset of the Benders' constraints. They also introduced a method to generate Benders' constraints based on a shortest path and a capacity cut. This method is used in the initialisation to "warm" start the RMP.
Given a set of vertices S, denote $\overline{S}$ to be the complement of the vertices $V\backslash S$. The cut $A(S,\overline{S})$ are all arcs that start in S and end in $\overline{S}$. The method works as follows. For every commodity the shortest path between its source-target pair is calculated with respect to number of arcs, i.e. every arc has length 1. Then for a cut $A(S,\overline{S})$ the number of arcs that are used in the shortest path for a commodity k is counted, call this $c_k^{A(S,\overline{S})}$. The Benders' constraint generated with this method is then defined by (9).

$$\sum_{a\in A(S,\overline{S})} u_a x_a + u_a^m y_a \geq \sum_{k\in K} c_k^{A(S,\overline{S})} d_k \tag{9}$$

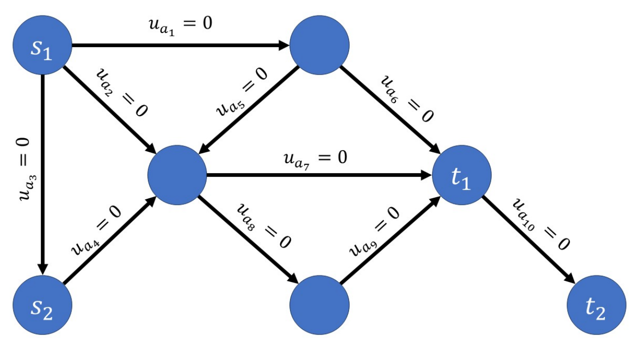
Since the goal is to allow enough flow between the source and target node of every commodity a minimal capacity cut is an obvious candidate to generate useful cuts. By assigning a "virtual" capacity to the arcs a minimal capacity cut can be obtained between a source-target pair. The method above can then be used to generate a Benders' constraint. Assigning $\sum_{k\in K} c_k^{A(S,\overline{S})} d_k$ as the new virtual capacity of the arcs in the cut allows for a new generation of a minimal capacity cut. Repeat this process until the maximum flow between the source and target node is greater or equal to the demand of the commodity. Which happens at some point by min-cut/max-flow theorem. The pseudo code for this algorithm is shown in Algorithm 2 and an example on the process is show in Section 3.1.1. In the initialisation this algorithm is called with the zero vectors for x and y.

Algorithm 2 Min-Cuts(x, y)

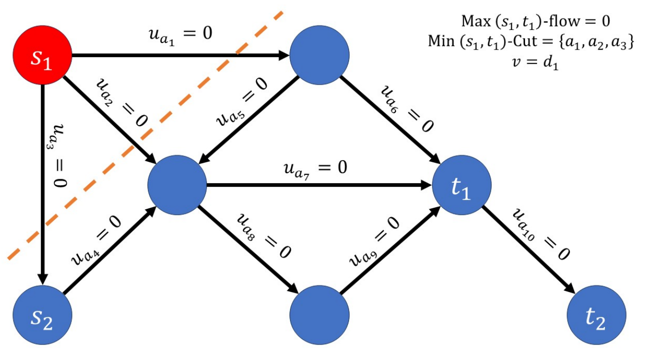
```
$\mathcal{C}_{\text{new}} := \emptyset$
  $U_a \leftarrow x_a \cdot u_a + y_a \cdot u_a^m \quad \forall a \in A$        ▷ $U_a$ is the "virtual" capacity of arc $a$
  for $k \in K$ do
      while True do
          $A(S,\overline{S}), f_{\max} \leftarrow$ Min-Cut/Max-flow($s_k, t_k$) algorithm
          $v \leftarrow 0$
          for $k \in K$ do
              for $a \in A(S,\overline{S})$ do
                  if $a$ in shortest $s_k, t_k$-path then
                      $v \leftarrow v + d_k$
          $\mathcal{C} \leftarrow$ constraint generated with $A(S,\overline{S}) \geq v$        ▷ Eq. (9)
          $\mathcal{C}_{\text{new}} \leftarrow \mathcal{C}_{\text{new}} \cup \mathcal{C}$
          for $a \in A(S,\overline{S})$ do
              $U_a \leftarrow v$
          if $f_{\max} \geq d_k$ then
              break
  return $\mathcal{C}_{\text{new}}$
```

3.1.2 Initialisation of the RLP

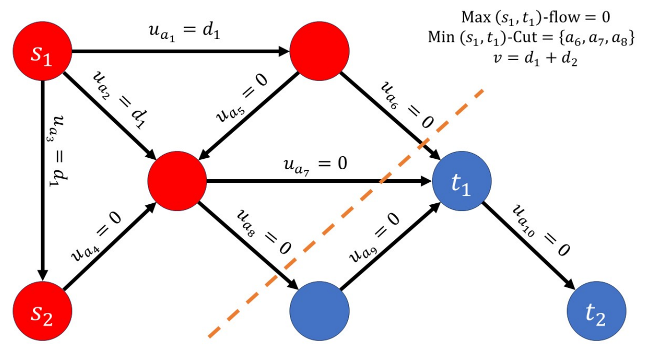
In an optimal solution usually not all paths between two source-target nodes are needed, it would be a waste of time to actually generate all these paths. So instead a subset is generated in which paths are added according to column generation which will be explained in Section 3.2. To this end it is not necessary to initialise commodity-path-sets, however in practical cases the shortest path will be (part of) the optimal solution for at least some of the commodities. Therefore initialising the commodity-path-sets with the shortest path will usually improve computation time.



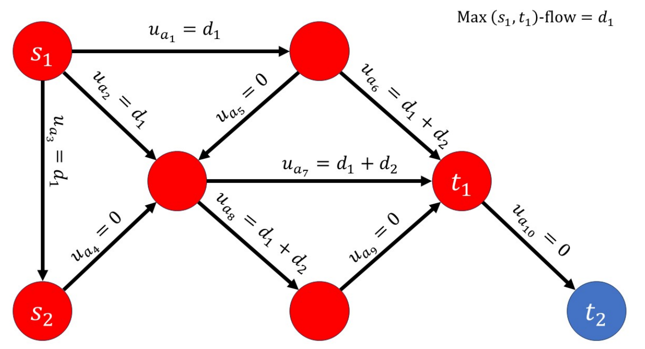
(a) The capacity on the arcs are initialised with zero for all arcs



(b) A max-flow/min-cut computation is done for the first commodity. The max-flow from s_1 to t_1 is 0 and a min-cut is $\{a_1, a_2, a_3\}$. The total flow-value v that has to be sent over this cut is d_1. The capacities of the arcs in the min-cut are increased with d_1 and the constraint that is generated at this step is: $u_{a_1}x_{a_1} + u^m_{a_1}y_{a_1} + u_{a_2}x_{a_2} + u^m_{a_2}y_{a_2} + u_{a_3}x_{a_3} + u^m_{a_3}y_{a_3} \geq d_1$.

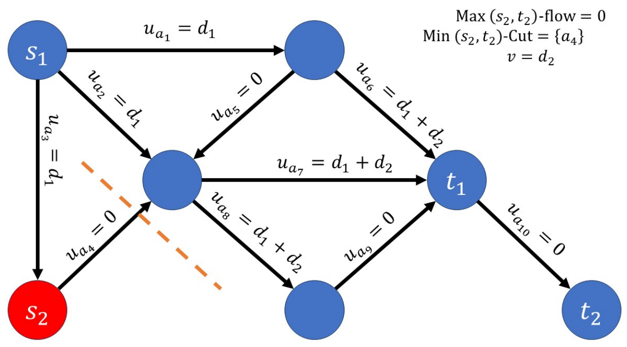


(c) In this step the total flow-value $v = d_1 + d_2$ since the shortest paths for both k_1 and k_2 intersect the minimal cut. The capacities of the arcs in the min-cut are increased with $d_1 + d_2$ and the constraint that is generated at this step is: $u_{a_6}x_{a_6} + u^m_{a_6}y_{a_6} + \cdots + u_{a_8}x_{a_8} + u^m_{a_8}y_{a_8} \geq d_1 + d_2$.

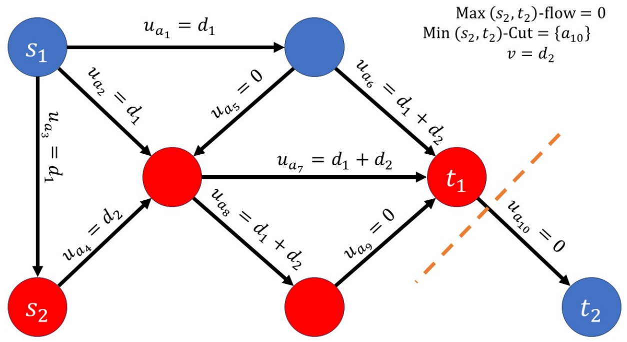


(d) t_1 is now reachable from s_1 and the max-flow is equal to d_1 so the algorithm stops for k_1 and moves on to k_2.

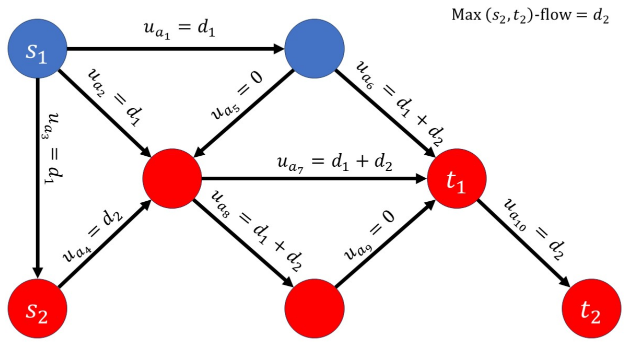
Figure 1: Example on how the Min-Cuts(x, y) algorithm works for an instance with seven nodes, ten arcs and two commodities. The commodities are $k_1 = (s_1, t_1, d_1)$ and $k_2 = (s_2, t_2, d_2)$. The red circles indicate the reachable nodes from the considered source node and the dashed orange line is the minimal cut.



(e) The steps are repeated only this time for commodity k_2. Notice that the direction of the arc matters and thus that arc a_3 is not part of the min-cut. Generated constraint: $u_{a_3}x_{a_3} + u^m_{a_3}y_{a_3} \geq d_2$



(f) The total flow is d_2. The generated constraint is $u_{a_{10}}x_{a_{10}} + u^m_{a_{10}}y_{a_{10}} \geq d_2$



(g) t_2 is now reachable from s_2 and the max-flow is equal to d_2 so the algorithm stops for k_2 and since all commodities have been considered the algorithm stops.

Figure 1: Example continuation of Algorithm 2

3.2 Column generation

In most practical cases there are few paths over which a positive flow is sent in an optimal solution. Also Theorem 1 presents an upper bound of the total number of paths needed in a instance solution. This gives rise to gradually generate commodity-flow-paths instead of working with a set of all possible paths. One can easily see that the Lifting LP using all paths would yield the same solution as the Lifting LP using a subset of the paths, provided that this subset contains paths of an optimal solution. Since a commodity-flow-path corresponds to a variable in the Lifting LP using column generation [4] to find the right paths, and working with the restricted Lifting problem (RLP) seems an obvious choice.
The reduced cost of a path can be found by considering the dual of the Lifting LP (10). Let $\sigma_k, -\Pi_a$ be the dual multipliers for Constraints (8b), (8c) respectively.

$$\max \quad \sum_{k\in K} d_k \cdot \sigma_k - \sum_{a\in A} (u_a \cdot \hat{x}_a + u^m_a \cdot \hat{y}^m_a) \cdot \Pi_a \tag{10a}$$

$$\text{s.t.} \quad \sigma_k - \sum_{a\in P} \Pi_a \leq r^k_P \qquad \forall P \in \mathcal{P}_k \tag{10b}$$

$$\sigma_k \geq 0 \qquad \forall k \in K \tag{10c}$$

$$\Pi_a \geq 0 \qquad \forall a \in A \tag{10d}$$

The 3-stage formulation works with a subset of the $\mathcal{P}_k$. This means that some paths, that are not in this set, might violate Constraint (10b), that corresponds to Constraint (7a) of the Benders' reformulation. If such a path exists it implies that the reduced cost corresponding to this path is negative. This means that considering this path in the RLP results in an equal or lower objective value of the RLP.
The goal is thus to find such paths. Luckily this boils down to a shortest-path problem for each commodity. By assigning the dual multiplier $\Pi_a + r_a \cdot \mu_k$ as arc lengths, a shortest path algorithm can be applied to each source-target pair of the commodities.
In order to find a feasible solution Inequality (11a) must hold for every path and for it to be optimal Inequality (11b) must hold which correspond to μ_k being 0 or 1. After every solve of the RLP the shortest path, with respect to these dual arc lengths, from source to target node for each commodity k is calculated. If the length of a path violates the constraint the path is added to the respective path-set.

$$\sigma_k - \sum_{a \in P} \Pi_a \leq 0 \tag{11a}$$

$$\sigma_k - \sum_{a \in P} \Pi_a + r_a \leq 0 \tag{11b}$$

4 Experimental results

The 3-stage-algorithm is tested against the compact formulation using Gurobi version 10.0.1 [8], running one thread on a Intel(R) Xeon(R) Gold CPU. The instances on which they were tested are divided in two sets and come from two different sources. The first set of instances are from the SND-library [10] and are taken from real-life problems. The second set are from the CommaLab of the University of Pisa [9] and are randomly generated. The different experimental results are show in Section 4.1 and in Section 4.2 for the SND-libary and CommaLab instances respectively.

4.1 SND-library instances

This library contains instances that are "provided by practitioners who are professionally involved in network design, as well as by researchers working on the theory or practice of network design" [10]. Therefore this set is a good source for testing how the algorithm would perform in real-life problems. Basic information about these instances is shown in Table 5.
For both formulations the total computation time was measured, which is shown in Fig. 2. The total time for the 3-stage formulation includes, shortest-path computations, max-flow/min-cut computations and solving the RMP and RLP. So all time necessary for solving the instance excluding reading and loading of the instance. The total time for the compact formulation includes all constraint generating functions embedded within Gurobi as well as solving the branch-and-bound algorithm employed by Gurobi. From the figure it is clear that in seven instances the 3-stage formulation outperforms the compact formulation. Instances such as *abilene* and *atlanta*. In four instances the compact formulation outperforms the 3-stage formulation (*di-yuan*, *new-york*, *nobel-us* and *polska*). Finally in four instances (*cost266*, *dfn-bwin*, *geant* and *sun*) both formulation did not find a solution, or proved it be optimal within 10 hours. After which the program was halted.

Table 5: Basic information of the used instances in the SND-library

Instance Name	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert K\rvert$	R
abilene	12	30	132	0.40
atlanta	15	44	210	0.34
cost266	37	114	1332	0.32
dfn-bwin	10	90	90	0.12
di-yuan	11	84	22	0.16
france	25	90	300	0.28
geant	22	72	462	0.31
janos-us	25	168	650	0.16
newyork	16	98	240	0.17
nobel-eu	28	82	378	0.34
nobel-germany	17	52	121	0.33
nobel-us	14	42	91	0.34
pdh	11	68	24	0.19
polska	12	36	66	0.34
sun	27	204	67	0.14

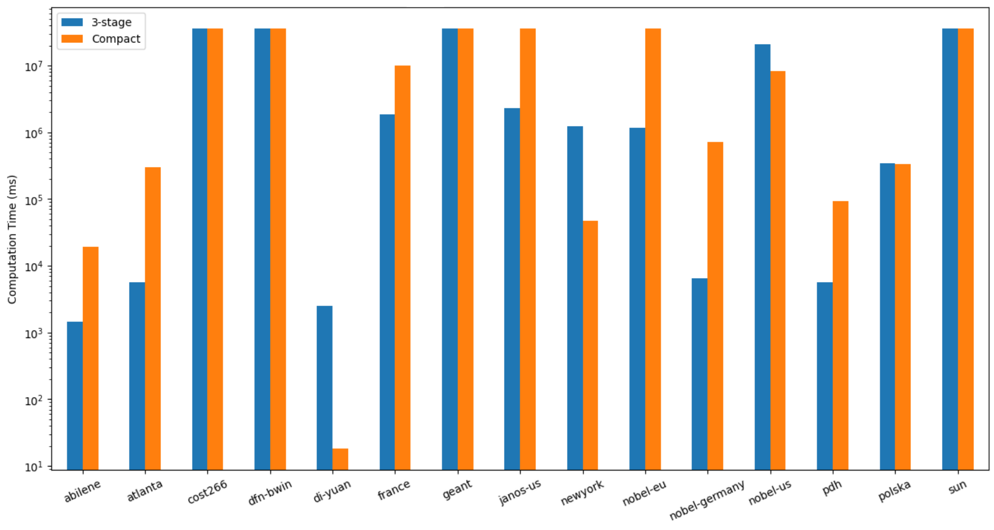
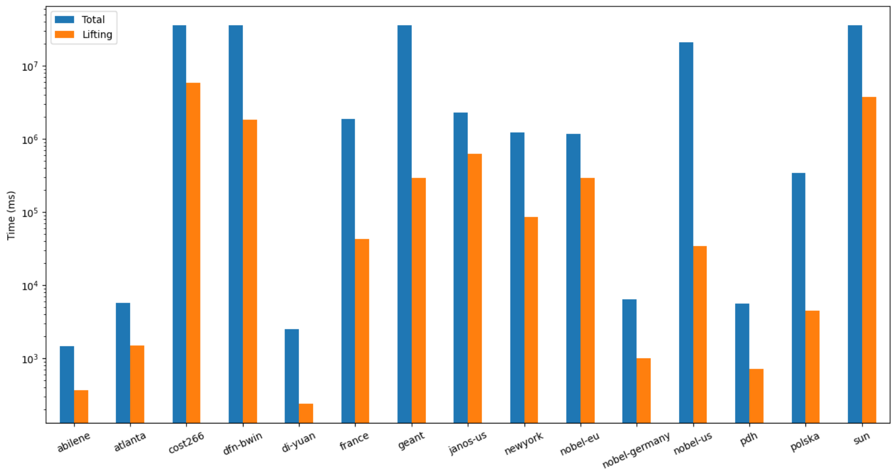


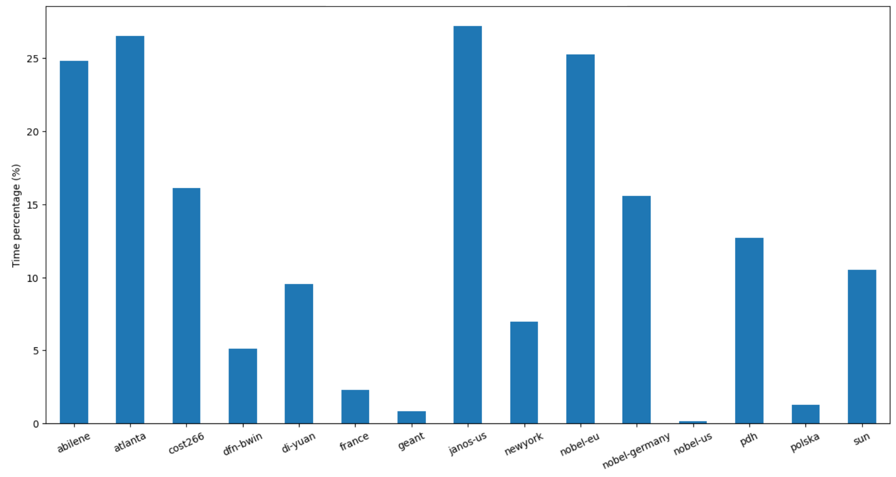
Figure 2: Comparison of total computation time for solving an instance to optimality for the compact formulation and the 3-stage formulation on instances of the SND-Library. Note that the y-axis is in logarithmic scale.

The reason why the 3-stage formulation outperforms the compact formulation and vice versa is obvious clear. The 3-stage formulation has two LPs to solve iteratively. The time the solver spends on solving the RLP is also measured. This is shown in Fig. 3a. This figure is shown with logarithmic scale in order to visualise the data of all instances. Therefore Fig. 3b shows the percentage of the total time spent in the RLP.
Clearly in all cases the solver spends less than 30% of the total time solving the RLP.

In the column generation part of the 3-stage formulation shortest paths are generated according to its dual length. During this stage many paths are generated, which are unused in an optimal solution this is shown in Fig. 4. The percentage of used paths of the total found paths is shown in Fig. 4b. For many of the instances this is below 10%, indicating that many paths are found during solving, on which no flow will be sent in an optimal solution.



(a) Used time solving the RLP in the 3-stage formulation in comparison to the total computation time per instance. Note that the y-axis is in logarithmic scale.



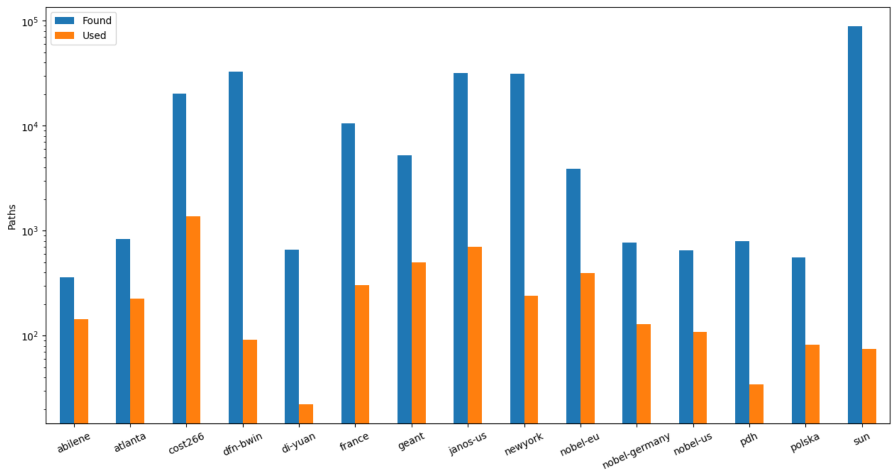
(b) Percentage of the total computation time spent on solving the RLP per instance.

Figure 3: Computation times for solving the RLP in the 3-stage formulation on instances of the SND-Library.

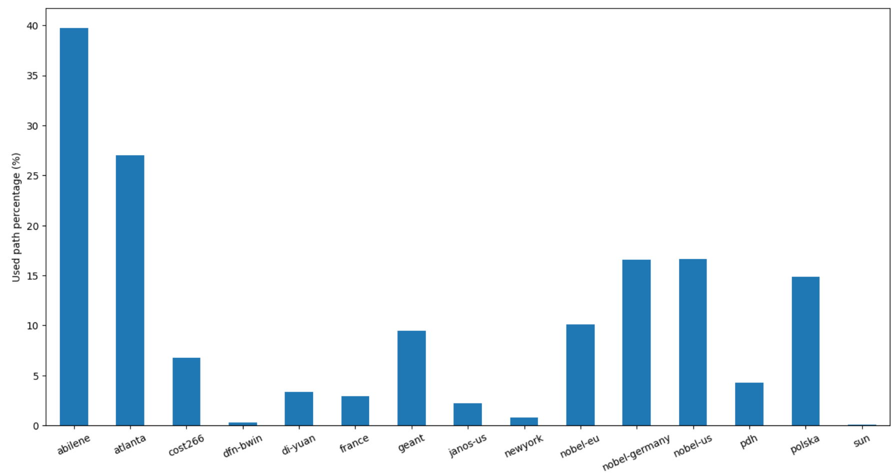
4.2 CommaLab instances

This library contains instances for many different types of network design. In this thesis however, only the instances provided in the section "Multicommodity Network Design Problems: The Canad problems" have been used. These problems all have two extra attributes, an `F` or `V` indicating whether the fixed cost is predominant over the variable cost, i.e. whether opening an arc costs more than routing more flow over an arc. The second attribute is a `T` or `L` indicating whether the capacities are tight or loose. A capacity is tight if an optimal solution exists using all available capacity. An overview of the instance information is shown 6. Again the total computation time was measured for both the compact and 3-stage formulation and is shown in Fig. 5.

In Fig. 6 the information about the RLP is shown and in Fig. 7 that of the paths.



(a) Total number of obtained paths while solving an instance compared to the total number of paths that end up used in the best found solution. Note that the y-axis is in logarithmic scale.



(b) Percentage of obtained paths used in the best found solution.

Figure 4: Paths obtained and used in the 3-stage formulation of instances of the SND-Library.

Table 6: Overview of basic information of CommaLab instances. Instances with the same amount of nodes, arcs etc are different with respect to the extra attribute. The first instance of a row will always be `V-L`, the second `V-T`, the third `F-L` and the fourth `F-T`. Note that c34 is not in the instances, the corresponding extra attributes `V-T` are also missing.

Instances	|N|	|A|	|K|	R
c33, c35, c36	20	230	40	0.11
c37, c38, c39, c40	20	230	200	0.09
c41, c42, c43, c44	20	300	40	0.09
c45, c46, c47, c48	20	300	200	0.07
c49, c50, c51, c52	30	520	100	0.07
c53, c54, c55, c56	30	520	400	0.06
c57, c58, c59, c60	30	700	100	0.05
c61, c62, c63, c64	30	700	400	0.05

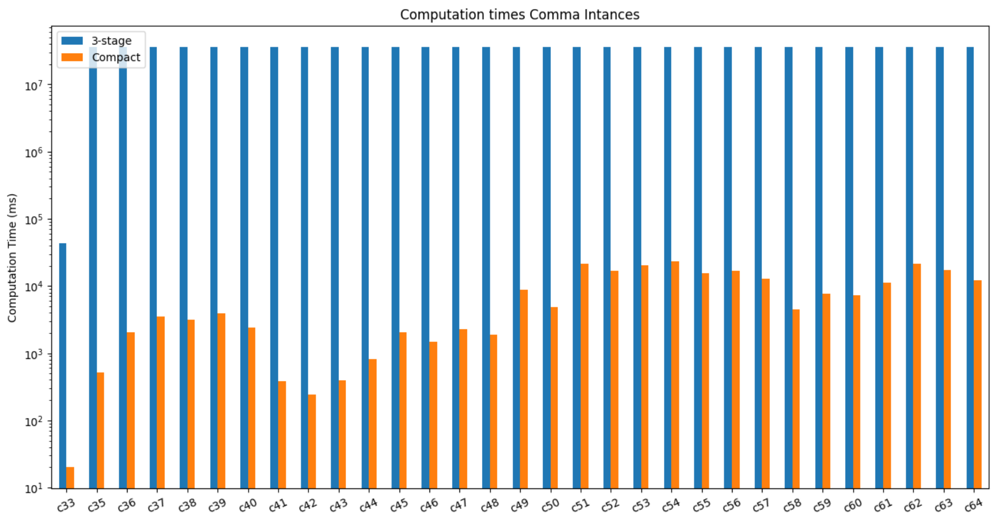


Figure 5: Comparison of total computation time per formulation of instances in the CommaLab library.

5 Discussion/Conclusion

Solving the 3-stage formulation does sometimes result in better computation times. However it depends on the instances. So far it is clear that some instances of an SMND with positive modular capacity are solved faster with the 3-stage formulation than with the compact formulation as can be seen in Fig. 2. In the case of zero modular capacity the compact formulation is always solved faster as can be seen in Fig. 5. The reason why one outperforms the other is not immediately clear. In this section some possible reasons are discussed.

5.1 Weak constraints

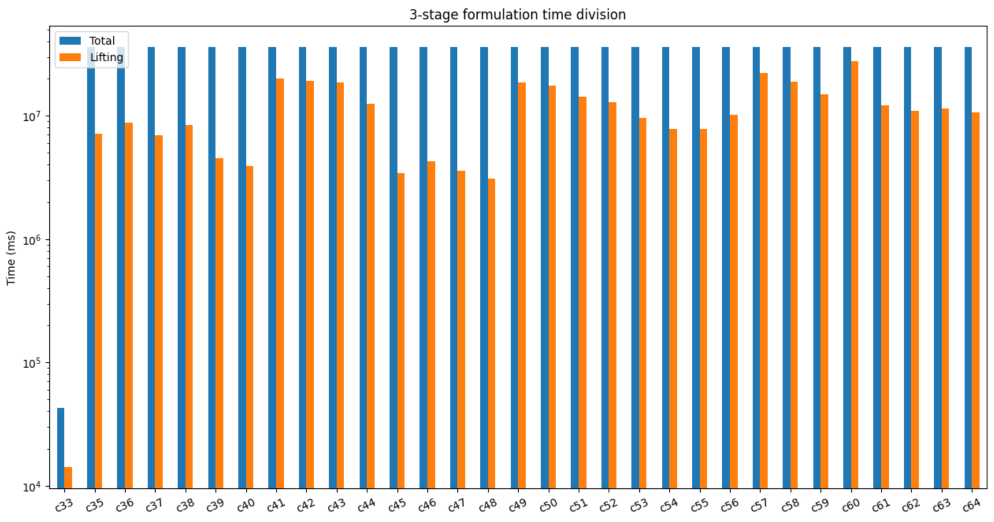
The time spent solving the RLP is in most instances relatively low in comparison to the total solving time. For the SND-library instances it is always below 30% as can be seen in Fig. 3a. For the CommaLab instances it is usually below 45%. This means that most of the time the solver is busy solving the RMP. Difficulties in solving the RMP could arise by having weak Benders' constraints, since the tighter the constraints are the faster an integer solution can be found. These constraints are possibly weak if generated inequalities differ only slightly between consecutive iterations. For a specific commodity the difference between two inequalities could for instance be only two arcs with a slightly higher bound. Such a small difference could occur when there is also a small difference in the paths. Therefore it might be worth investigating a better way to determine new paths.

5.2 Implementation issues

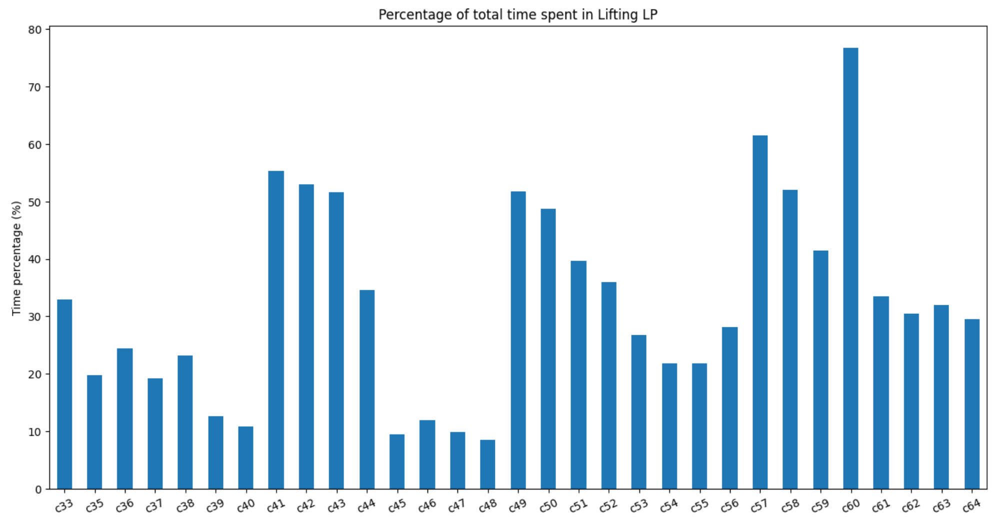
Throughout the thesis, implementation bugs were a headache. In the end, everything was working yet a small bug was still found in the code. This bug caused Benders' constraints to be generated multiple times. Since mathematically speaking this is not possible, it is most likely a numerical computation issue. The bug happens when checking for violated constraints. In the code this violation is checked within some ϵ of the actual value. This causes some paths to be considered as violated while in truth this is not the case. This can lead however to a significant delay in computation time, due to complete iterations resulting in no additional information to the model.

5.3 Modular capacity benefits 3-stage formulation

A big difference between the instances of the SND-library and the CommaLab instances, is that the CommaLab libary only contains instances of the SMND in which the modular capacity is zero. The



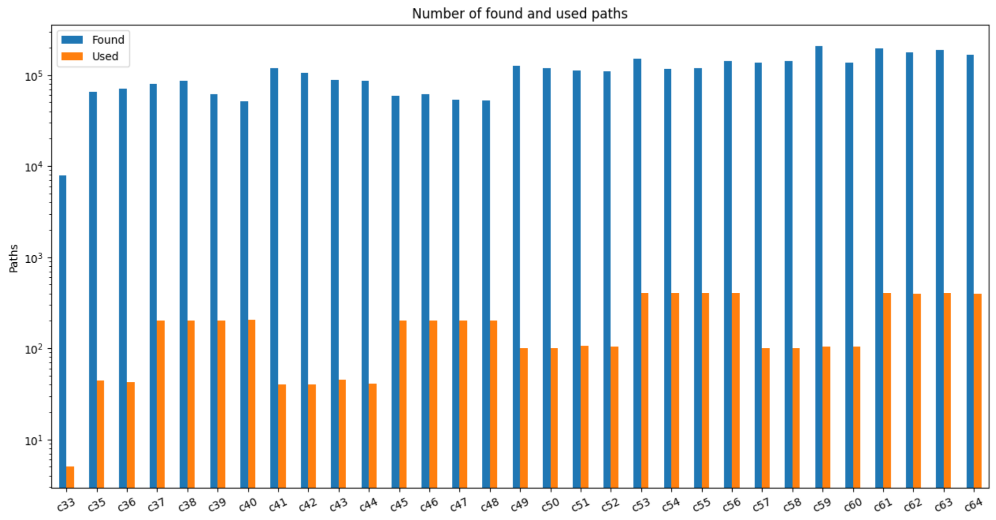
(a) Total computation time for solving the RLP compared to the total computation time.



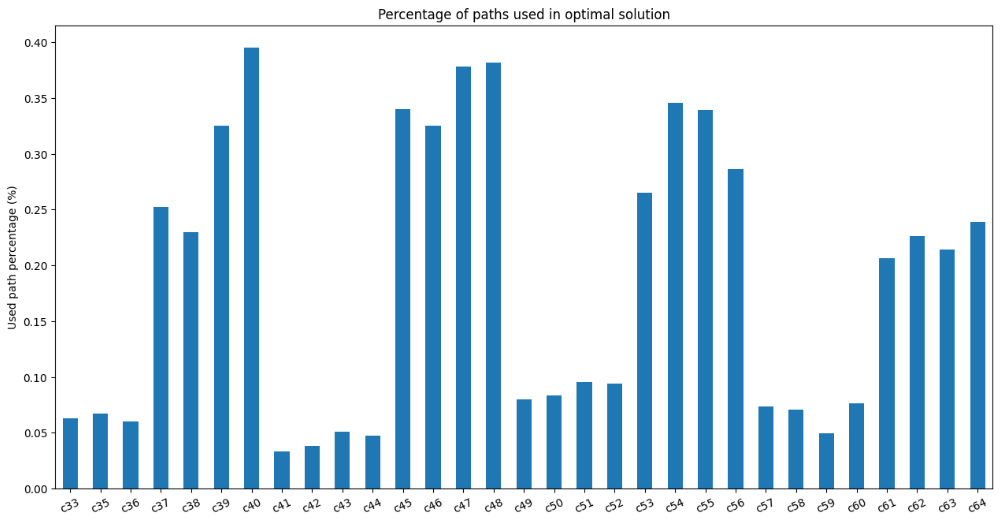
(b) Percentage of time spent solving the RLP in respect to the total computation time.

Figure 6: Computation times of the 3-stage formulation of the CommaLab instances.

SND-library however only contains instances in which the modular capacity is always greater than zero. This benefits the 3-stage formulation. To realise this, consider the following situation. Somewhere during solving of the 3-stage formulation a solution to the RMP is computed. This yields a set of capacities on certain arcs. If this is not the optimal solution, a feasibility or optimality constraint will be generated. In both cases the dual of the RLP is computed and the obtained multipliers are used in the constraint. From Constraint (6b) it is clear that the non-zero multipliers are of importance. Which multipliers are non-zero can be seen by looking at the dual of the RLP. As the only non-zero Π_a are the ones that belong to the paths in the optimal solution on which not enough capacity is installed. This means that if infinite capacity can be installed on arcs the generated constraint ensures enough capacity is installed on those arcs the next iteration.



(a) Total number of paths with positive flow in the best found solution compared to the total number of obtained paths during solving.



(b) Percentage of total obtained paths with positive flow in the best found solution.

Figure 7: Paths obtained during solving of the 3-stage formulation in the CommaLab instances

5.4 Possible improvements

Instead of warm starting the 3-stage formulation with the shortest paths it might be useful to start the 3-stage algorithm with the optimal paths in the worst feasible solution, i.e. opening/buying all arcs. This might especially be useful in the case of the CommaLab instances. As there are no module cost column generation would find paths in the graph with gradually declining routing cost. Of course this could still take a long time but it removes some iterations solving the RMP.

References

[1] J. F. Benders. “Partitioning procedures for solving mixed-variables programming problems”. en. In: *Computational Management Science* 2.1 (Jan. 2005), pp. 3–19. ISSN: 1619-6988. DOI: 10.1007/s10287-004-0020-y. URL: https://doi.org/10.1007/s10287-004-0020-y (visited on 07/14/2023).

[2] Daniel Bienstock, Sunil Chopra, Oktay Günlük, and Chih-Yang Tsai. “Minimum cost capacity installation for multicommodity network flows”. en. In: *Mathematical Programming* 81.2 (Apr. 1998), pp. 177–199. ISSN: 1436-4646. DOI: 10.1007/BF01581104. URL: https://doi.org/10.1007/BF01581104 (visited on 07/12/2023).

[3] Alysson M. Costa, Jean-François Cordeau, and Bernard Gendron. “Benders, metric and cutset inequalities for multicommodity capacitated network design”. en. In: *Computational Optimization and Applications* 42.3 (Apr. 2009), pp. 371–392. ISSN: 1573-2894. DOI: 10.1007/s10589-007-9122-0. URL: https://doi.org/10.1007/s10589-007-9122-0 (visited on 10/16/2023).

[4] George B. Dantzig and Philip Wolfe. “Decomposition Principle for Linear Programs”. en. In: *Operations Research* (Feb. 1960). Publisher: INFORMS. DOI: 10.1287/opre.8.1.101. URL: https://pubsonline.informs.org/doi/abs/10.1287/opre.8.1.101 (visited on 10/16/2023).

[5] E. W. Dijkstra. “A note on two problems in connexion with graphs”. en. In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271. ISSN: 0945-3245. DOI: 10.1007/BF01386390. URL: https://doi.org/10.1007/BF01386390 (visited on 10/18/2023).

[6] L. R. Ford and D. R. Fulkerson. “A Suggested Computation for Maximal Multi-Commodity Network Flows”. In: *Management Science* 5.1 (Oct. 1958). Publisher: INFORMS, pp. 97–101. ISSN: 0025-1909. DOI: 10.1287/mnsc.5.1.97. URL: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.5.1.97 (visited on 10/10/2023).

[7] Andrew V. Goldberg and Robert E. Tarjan. “A new approach to the maximum-flow problem”. In: *Journal of the ACM* 35.4 (1988), pp. 921–940. ISSN: 0004-5411. DOI: 10.1145/48014.61051. URL: https://dl.acm.org/doi/10.1145/48014.61051 (visited on 10/18/2023).

[8] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com.

[9] *Multicommodity Flow Problems – CommaLAB*. en-US. URL: https://commalab.di.unipi.it/datasets/mmcf/ (visited on 10/04/2023).

[10] Orlowski S, Pióro M, Tomazweski A., and Wessäly R. *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*. English. Apr. 2007. URL: http://sndlib.zib.de (visited on 07/11/2023).

[11] Khodakaram Salimifard and Sara Bigharaz. “The multicommodity network flow problem: state of the art classification, applications, and solution methods”. en. In: *Operational Research* 22.1 (Mar. 2022), pp. 1–47. ISSN: 1866-1505. DOI: 10.1007/s12351-020-00564-8. URL: https://doi.org/10.1007/s12351-020-00564-8 (visited on 10/10/2023).

[12] Gary R. Waissi. “Network flows: theory, algorithms and applications”. In: *Interfaces* 24.4 (1994). Ed. by R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Publisher: INFORMS, pp. 133–135. ISSN: 00922102, 1526551X. URL: http://www.jstor.org/stable/25061920 (visited on 10/10/2023).