FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Exact Approaches to the Network Design Problem with Relays

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Martin Riedler
Matrikelnummer 0828221

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ.Lektorin Mag.rer.nat. Dr.techn. Ivana Ljubić
Mitwirkung: Dipl.-Ing. Dr.techn. Mario Ruthmair
Dipl.-Ing. Dr.techn. Markus Leitner

Wien, 05.08.2014

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Exact Approaches to the Network Design Problem with Relays

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computational Intelligence

by

Martin Riedler
Registration Number 0828221

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ.Lektorin Mag.rer.nat. Dr.techn. Ivana Ljubić
Assistance: Dipl.-Ing. Dr.techn. Mario Ruthmair
Dipl.-Ing. Dr.techn. Markus Leitner

Vienna, 05.08.2014

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Martin Riedler
Weindlau 30, 4432 Ernsthofen

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I would like to thank my advisor Ivana Ljubić for providing this interesting topic, for helping me to improve the models and for supporting me with all aspects of the thesis.

I would also like to thank my co-advisors Markus Leitner and Mario Ruthmair for their suggestions for improvements and new models, for their help with implementation issues and for their comments on the thesis.

I especially want to thank everyone who supported me during the phase of completing the thesis to finish as fast as possible, to give me the opportunity to continue researching as an assistant at the Institute of Computer Graphics and Algorithms at the Vienna University of Technology.

Last but not least, I want to thank my family for all their support.

Abstract

In this thesis we develop exact approaches for solving the Network Design Problem with Relays (NDPR). The NDPR can be motivated as follows. We are given a set $\mathcal{K}$ of vertex pairs that need to communicate with each other in an undirected graph $G = (V, E)$. A signal is sent from a source to a target from $\mathcal{K}$ but after a distance of d_{max} the signal deteriorates and we need to install a relay to regenerate it. Alternatively, we may install new edges in an existing graph (that can also be empty) to shorten the distance. Every edge $e \in E$ has cost w_e and a distance d_e. The cost for installing a relay at vertex $i \in V$ is c_i. The goal of the NDPR is to find a selection of edges to install and vertices where relays are to be placed enabling communication between the pairs in $\mathcal{K}$ s.t. the sum of relay and edge costs is minimal.

The NDPR arises in the context of network design when distance limits have to be kept due to signal deterioration. To cover longer distances equipment for signal regeneration is required. This equipment is expensive and thus minimization is required. Another area of application is e-mobility. E-cars need to recharge after traveling a certain distance. Recharging stations are expensive and one only wants to build as few as necessary.

Previous work on the NDPR mainly focuses on heuristic approaches. In the following we are going to present exact solution approaches based on mixed integer linear programming. We introduce compact models, models with an exponential number of constraints, models with an exponential number of variables and models with an exponential number of variables and constraints. We divide our models w.r.t. the underlying graph transformation. The first set of our models is based on layered graphs and the second one on communication graphs.

We test our models against modified versions of instances from the previous literature. One of our algorithms solves the first set of those instances to optimality and also a large amount of the instances of the second set. Moreover, we present a set of entirely new instances containing a larger number of commodity pairs. Some of our algorithms solve most of these instances to optimality but some of the new instances turned out to be very challenging.

Kurzfassung

In dieser Arbeit entwickle ich exakte Lösungsansätze für das Network Design Problem with Relays (NDPR). Das NDPR kann folgendermaßen motiviert werden. Gegeben ist eine Menge $\mathcal{K}$ von Knotenpaaren die in einem ungerichteten Graphen $G = (V, E)$ kommunizieren müssen. Ein Signal wird vom ersten Knoten des Paares zum anderen gesendet. Nachdem eine Distanz von d_{max} zurück gelegt wurde verschlechtert sich das Signal zu sehr und muss durch Platzieren eines relays aufgefrischt werden. Als Alternative können neue Kanten in ein bestehendes (möglicherweise leeres) Netzwerk eingefügt werden, um die zurückgelegte Distanz zu verkleinern. Jede Kante $e \in E$ hat Kosten w_e und eine Distanz d_e. Die Kosten um ein relay bei Knoten $i \in V$ zu installieren, betragen c_i. Das Ziel des NDPR ist es eine Menge von zu installierenden Kanten und relays auszuwählen, die Kommunikation zwischen allen Paaren in $\mathcal{K}$ ermöglichen, sodass die Summe aus relay- und Kantenkosten minimal ist.

Das NDPR findet Anwendung im Netzwerkentwurf, wo Distanzbegrenzungen eingehalten werden müssen, um eine zu starke Signalverschlechterung zu vermeiden. Das Zurücklegen größerer Distanzen erfordert Komponenten zur Auffrischung des Signals. Diese Komponenten sind teuer und daher ist Minimierung notwendig. Ein anderes Anwendungsgebiet ist e-mobility. Elektroautos können nur eine bestimmte Distanz zurücklegen, bevor sie wieder aufgeladen werden müssen. Ladestationen sind teuer, deshalb möchte man nur so wenige wie nötig bauen.

Die verfügbare Literatur zum NDPR umfasst hauptsächlich heuristische Ansätze. Im Folgenden werde ich exakte Lösungsansätze vorstellen die auf mixed integer linear programming basieren. In dieser Arbeit stelle ich kompakte Modelle, Modelle mit einer exponentiellen Anzahl an Constraints, Modelle mit einer exponentiellen Anzahl an Variablen und Modelle mit einer exponentiellen Anzahl an Constraints und Variablen vor. Die Modelle werden im Bezug auf die verwendete Transformation des Graphen unterteilt. Die erste Gruppe von Modellen basiert auf sogenannten „layered graphs“ und die zweite auf sogenannten „communication graphs“.

Wir testen unsere Modelle mit modifizierten Instanzen aus der Literatur. Ein Algorithmus löst alle Instanzen der ersten Gruppe optimal und einen Großteil der Instanzen der zweiten Gruppe. Darüber hinaus stelle ich neue Instanzen mit einer größeren Anzahl an Knotenpaaren vor. Einige Algorithmen finden für einen großen Teil dieser Instanzen die optimale Lösung aber ein kleiner Teil der Instanzen erwies sich als besonders schwierig.

Contents

CHAPTER 1

Introduction

1.1 Problem Definition and Motivation

The Network Design Problem with Relays (NDPR) is defined on an undirected graph $G = (V, E, c, w, d)$ with relay costs $c : V \to \mathbb{Q}^+$, edge costs $w : E \to \mathbb{Q}_0^+$ and edge delays $d : E \to \mathbb{Q}^+$. The edge set E is the disjoint union of the set of free edges $E^0 = \{e|w(e) = 0\}$ and the set of augmenting edges $E^* = \{e|w(e) > 0\}$. Although we allow rational delays here, some of the models discussed in this thesis require integral delays. Rational delays may, however, be transformed to integral delays by means of scaling.

Furthermore, we are given a maximum delay $d_{\max} \in \mathbb{N}^+$ and a set $\mathcal{K} \subseteq V \times V$ of node pairs that need to communicate with each other. W.l.o.g. we assume $\forall (i, j) \in \mathcal{K} : i < j$. In the following we will refer to the pairs in $\mathcal{K}$ also as *commodities*. According to $\mathcal{K}$ we define two additional sets:

$$\mathcal{K}_S = \{u|(u, v) \in \mathcal{K}\} \qquad \ldots \text{ set of sources}$$
$$K(u) = \{v|(u, v) \in \mathcal{K}\} \qquad \ldots \text{ set of targets that have to be reached by source } u$$

We define the delay of a path p as the sum of the delays of its edges, i.e., $\Delta(p) = \sum_{e \in p} d_e$. Now consider a path $p = (s, p_1, r_1, p_2, \ldots, r_k, p_{k+1}, t)$ for relays $\{r_1, \ldots r_k\}$ and subpaths $\{p_1, \ldots p_{k+1}\}$. We refer to the subpaths from source s to the subsequent relay, between every two consecutive relays and from the last relay to the target t as *segments* of the path. If the path contains no relays the only segment of the path is the path itself. We call a path feasible or *feasible connection* iff none of its segments has a delay larger than the delay bound.[1] Note that the considered paths do not have to be simple, i.e., they might contain some vertices more than once which is often referred to as walk. Details on the structure of the paths will be given in Chapter 2.

[1] If there is a feasible connection between two vertices we also say that these vertices can communicate and that they are connected.

We define the delay of a vertex v w.r.t. some path p as the delay of the segment between the relay preceding v (or the source if no such relay exists) and v. The delay of relays and the source is always zero. As already mentioned vertices might be visited more than once on a path. Thus, a vertex is assigned different delay values depending its the position in the path. We are going to discuss this in detail in Chapter 2. The assignment of delay values to the vertices can be considered as traversing the edges along the path. The current delay starts at zero and whenever we use an edge we increase the current delay by the delay of this edge. When we reach a relay we reset the accumulated delay to zero. The path is a feasible connection iff the accumulated delay never exceeds d_{max}.

A solution to the NDPR consists of a selection of augmenting edges $\hat{E} \subseteq E^*$ to install in the network and a subset of vertices $\hat{V} \subseteq V$ where relays are to be placed. A solution is feasible iff all pairs in $\mathcal{K}$ can communicate using the edges in $\hat{E} \cup E^0$ and relays at vertices $\hat{V}$. The aim of the NDPR is to find a feasible solution $(\hat{V}, \hat{E})$ that minimized the total costs defined by $\sum_{i \in \hat{V}} c_i + \sum_{e \in \hat{E}} w_e$.

In addition to the problem specific definitions we also use the following common notation. W.r.t. a graph with vertex set V we define the complement of some subset $S \subseteq V$ as $\bar{S} = V \setminus S$.

For an undirected graph $G = (V, E)$ we denote the set of directed arcs according to subset X of the edges as $A(X)$, i.e. $A(X) = \{(i, j), (j, i) | \{i, j\} \in X\}$. For set S we define the set of incident edges as $\delta(S) = \{\{i, j\} | i \in S, j \in \bar{S}, S \subset V, \{i, j\} \in E\}$. If S contains only a single vertex $i \in V$ we define $\delta(i) \coloneqq \delta(\{i\})$.

For a directed graph $G = (V, A)$ we define the set of out-going arcs according to set S as $\delta^+(S) = \{(i, j) | i \in S, j \in \bar{S}, S \subset V, (i, j) \in A\}$ and the set of in-coming arcs as $\delta^-(S) = \{(i, j) | i \in \bar{S}, j \in S, S \subset V, (i, j) \in A\}$. We also use these sets w.r.t. undirected graphs using $A = A(E)$ as arc set. Moreover, if S contains only a single vertex $i \in V$ we define $\delta^+(i) \coloneqq \delta^+(\{i\})$ and $\delta^-(i) \coloneqq \delta^-(\{i\})$.

The NDPR arises when commodities need to be transferred between sets of start and target locations. In addition, the maximum distance that might be covered is restricted. To cover distances beyond this limit we require special equipment along the path. The distance limit has to be kept from the start to the first intermediate stop, between consecutive intermediate stops and the last stop and the target location. The aim is to identify the locations of the intermediate stops and the required route.

Typical NDPR application arise in the ?? of communication networks. No matter if we are talking about telecommunication networks or modern optical networks, they have a technical limitation in common. Signals cannot be transmitted over arbitrary distances. At some point the signal deteriorates too much and has to be regenerated. Otherwise, the signal might be lost or the transmitted information might be falsified. However, the regeneration equipment is usually expensive and therefore the goal is to use as few such devices as possible [6].

In telecommunication networks so called repeaters are used to regenerate signals. In a network design project 422 communities in Alberta had to be connected which required to place a repeater at least every 70 km [4].

For optical networks, for example, there exist different forms of regeneration with increasing complexity:

- 1R (reamplification)
- 2R (reamplification and reshaping)
- 3R (reamplification, reshaping and retiming)

The first form is relatively cheap but can only be done a limited number of times before reshaping and probably retiming are required (see [31]). In the following we restrict ourselves to the placement of 3R relays as has been done for the Regenerator Location Problem (RLP). This is motivated by the fact that the placement of this type of relays is more common in practice [6].

A completely different field of application is e-mobility. E-cars can only cover a certain distance before they need to recharge. Thus, recharge stations are required to travel farther. Since such stations are expensive the goal is to have as few as possible whilst enabling travels between arbitrary locations. Furthermore, using certain streets might require a toll. If a company wants to build recharge stations for their fleet they also have to gauge if such connections are to be used. Although building costs occur only once and tolls have to be paid consistently they can be related to each other when considering a longer time span for the tolls.

1.2 State of the Art

The NDPR was originally introduced by Cabral et al. [4] in 2007. They argue that the problem is NP-hard as it is a generalization of the Weight Constrained Shortest Path Problem (WCSPP) which is also NP-hard (see [13]). Besides heuristic approaches, intended to solve large instances, a first exact solution approach is presented. Their approach is reviewed in the following.

The formulation considers a set $\mathcal{K}' = \mathcal{K} \cup \{(v,u)|(u,v) \in \mathcal{K}\}$, i.e., connectivity in both directions is ensured separately. To obtain the arc set A, two arcs are generated for each edge with costs $w'_{ij} = w'_{ji} = \frac{w_{\{i,j\}}}{2}$ and delays $d'_{ij} = d'_{ji} = d_{\{i,j\}}$, respectively. Furthermore, we denote the set of paths from u to v by $P(u,v)$ and the set of relay patterns turning path p into a feasible connection by $R(p)$. W.r.t. the relay patterns $r \in R(p)$ we define constants b_i^r that are set to one if vertex $i \in V$ is a relay in the pattern and to zero otherwise.

The model uses variables $y_i, \forall i \in V$, to identify relays and variables $x_{ij}, \forall (i,j) \in A$, for the arcs. Furthermore, variables $\lambda_{uv}^{pr}, \forall (u,v) \in \mathcal{K}', \forall p \in P(u,v), \forall r \in R(p)$, are set to one if path $p \in P(u,v)$ with relay pattern $r \in R(p)$ is used to connect the pair $(u,v) \in \mathcal{K}$. Using this notation the problem can be modeled as follows.

$$\min \sum_{i \in V} c_i y_i + \sum_{(i,j) \in A} w'_{ij} x_{ij}$$

$$\sum_{p \in P(u,v), r \in R(p)} \lambda_{pr}^{uv} = 1 \qquad \forall (u,v) \in \mathcal{K}' \quad (\gamma^{uv}) \tag{1.1}$$

$$\sum_{p \in P(u,v), r \in R(p): (i,j) \in p} \lambda_{pr}^{uv} \leq x_{ij} \qquad \forall (u,v) \in \mathcal{K}', \forall (i,j) \in A \quad (\mu_{ij}^{uv}) \tag{1.2}$$

$$\sum_{p \in P(u,v), r \in R(p)} b_i^r \lambda_{pr}^{uv} \leq y_i \qquad \forall (u,v) \in \mathcal{K}', \forall i \in V \quad (\alpha_i^{uv}) \tag{1.3}$$

$$x_{ij} - x_{ji} = 0 \qquad \forall \{i,j\} \in E \tag{1.4}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{1.5}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{1.6}$$

$$\lambda_{pr}^{uv} \in \{0,1\} \qquad \forall (u,v) \in \mathcal{K}', \forall p \in P(u,v), \forall r \in R(p) \tag{1.7}$$

The first set of constraints ensures that a feasible connection is selected for each commodity. Constraints (1.2) guarantee that for every selected path all its arcs will be part of the solution. The next set of inequalities states that the relay variables have to be set according to the chosen relay patterns. Equations (1.4) ensure that an arc is either selected in both directions or not at all. This is motivated by the fact that the arcs correspond to edges in the original problem.

The amount of λ-variables is in general exponential. Thus, the model requires column generation. For references on this topic we refer to Section 1.4.

Pricing Subproblem

To state the pricing subproblem, from [4], we relax the path variables to their continuous counterparts, i.e., $\lambda_{pr}^{uv} \geq 0$. To state the dual constraints for the path variables we use dual variables γ^{uv} for Constraints (1.1), dual variables μ_{ij}^{uv} for Constraints (1.2) and dual variables α_i^{uv} for Constraints (1.3):

$$\gamma^{uv} - \sum_{(i,j) \in A: (i,j) \in p} \mu_{ij}^{uv} - \sum_{i \in V} b_i^r \alpha_i^{uv} \leq 0 \qquad \forall (u,v) \in \mathcal{K}', p \in P(u,v), r \in R(p)$$

$$\mu_{ij}^{uv} \geq 0 \qquad \forall (u,v) \in \mathcal{K}', \forall (i,j) \in A$$

$$\alpha_i^{uv} \geq 0 \qquad \forall (u,v) \in \mathcal{K}', \forall i \in V$$

Thus, the pricing subproblem for each $(u,v) \in \mathcal{K}'$ looks as follows:

$$\underset{p \in P(u,v), r \in R(p)}{\arg\min} \left\{ 0 - \left(\gamma^{uv} - \left(\sum_{(i,j) \in A: (i,j) \in p} \mu_{ij}^{uv} + \sum_{i \in V} b_i^r \alpha_i^{uv} \right) \right) \right\}$$

This can be solved by the following subproblem:

$$\forall (u,v) \in \mathcal{K}', \; MCPPR_{uv} = \underset{p \in P(u,v), r \in R(p)}{\arg\min} \left\{ \sum_{(i,j) \in A: (i,j) \in p} \mu_{ij}^{uv} + \sum_{i \in V} b_i^r \alpha_i^{uv} \right\}$$

The problem $MCPPR_{uv}$ that needs to be solved is the Minimum Cost Path Problem with Relays (MCPPR). The MCPPR is NP-hard but fast pseudo-polynomial algorithms are available to solve this problem. A detailed description of this problem and efficient solution methods can be found in [27]. The MCPPR is essentially the path-variant of the NDPR, i.e., if we set $|\mathcal{K}| = 1$ for the NDPR we obtain the MCPPR.

In 2008 Kulturel-Konak and Konak [26] continued to work on the NDPR and presented a hybrid approach based on local search and a genetic algorithm. In Konak [25] an improved genetic algorithm is introduced. In this paper Konak also provides a variant of the exact approach by Cabral et al. [4] based on set covering constraints. In the following we are going to give an overview of this variant.

The model uses the same set of variables but avoids the duplication of $\mathcal{K}$ and as a result also the transformation of costs and delays. The most important difference, however, is that relay patterns are replaced by set covering constraints. To state these constraints we denote by $V^{\mathrm{I}}(p, j)$ the maximal set of nodes that can be traversed on path $p \in P(u, v)$ starting at node j in the direction from node u to node v without violating the delay bound, i.e., the sum of the delays of the corresponding edges must not exceed d_{max} and adding a further vertex causes a violation.

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E} w_e x_e$$

$$\sum_{p \in P(k)} \lambda_p^{uv} = 1 \qquad \forall (u, v) \in \mathcal{K}, \forall p \in P(u, v) \tag{1.8}$$

$$\sum_{(u,v) \in \mathcal{K}, p \in P(u,v): e \in p} \lambda_p^{uv} \leq |E| \cdot x_e \qquad \forall e \in E \tag{1.9}$$

$$\sum_{i \in V^{\mathrm{I}}(p,j)} y_i \geq \lambda_p^{uv} \qquad \forall (u, v) \in \mathcal{K}, \forall p \in P(u, v), \forall j \in p, V^{\mathrm{I}}(p, j) \neq p \tag{1.10}$$

$$x_{ij} - x_{ji} = 0 \qquad \forall \{i, j\} \in E \tag{1.11}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{1.12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E \tag{1.13}$$

$$\lambda_p^{uv} \in \{0, 1\} \qquad \forall (u, v) \in \mathcal{K}, \forall p \in P(u, v) \tag{1.14}$$

The first set of constraint is the same as for the previous model. Inequalities (1.9) differ since they sum up over all commodities and thus require a Big-M constant on the right-hand side. Instead of creating extra variables for all the feasible relay arrangements, Constraints (1.10) are used. These set covering constraints state that one of the vertices in set $V^{\mathrm{I}}(p, j)$ has to be a relay whenever $V^{\mathrm{I}}(p, j) \neq p$ since otherwise it is not possible to go on without violating the delay bound. The rest of the formulation is equivalent to the previous model.

In [25] the observations concerning the set covering constraints introduced in this formulation are used to design a genetic algorithm.

1.2.1 Regenerator Location Problem

The Regenerator Location Problem (RLP) was introduced in 2010 by Chen et al. [6]. The RLP is closely related to the NDPR but focuses on the placement of regenerators (=relays). The authors

do not deal with the selection of edges, i.e., it is assumed that an existing network is given s.t. all edges have a cost of zero. Furthermore, full connectivity is required, i.e., all node pairs have to be able to communicate. NP-hardness of the RLP was shown in [6]. The RLP is a special case of the NDPR for $E^* = \emptyset$ and $\mathcal{K} = \{(i,j)|(i,j) \in V \times V, i < j\}$.

Chen et al. [6] provide several Integer Linear Programming (ILP) models for the solution of the RLP. Unfortunately, these models cannot be directly used to solve the NDPR since they are not capable of selecting edges. Nevertheless, the authors provide some techniques that are useful for solving the NDPR. The most interesting technique used, are communication graphs. We are going to use this graph transformation for solving the NDPR. A detailed description will be given in Chapter 4.

In 2013, Chen et al. [5] introduced the so called Generalized Regenerator Location Problem (GRLP), see also [7]. In addition to the standard RLP node sets $S \subseteq V$ and $T \subseteq V$ are defined. Thereby, S is the set of candidate locations where relays may be installed. Furthermore, T is the set of terminal nodes which need to be able to communicate in a feasible solution, i.e., $\mathcal{K} = \{(i,j)|(i,j) \in T \times T, i < j\}$. Since the GRLP reduces to the RLP for $S = T = V$ it is also NP-hard.

With some slight modifications the NDPR is also able to solve these problems. We just have to assign infinite costs to the vertices in $V \setminus S$. If we obtain a solution with non-infinite costs we obtained a feasible solution. Otherwise it follows that the respective instance is infeasible. Alternatively, we can add constraints that prohibit that the vertices in $V \setminus S$ become relays which is easy for all models that we are going to present in the following.

1.2.2 Maximum Leaf Spanning Tree Problem/Minimum Connected Dominating Set Problem

The goal of the Maximum Leaf Spanning Tree Problem (MLSTP) is to find a spanning tree w.r.t. an undirected graph $G = (V, E)$ with a maximum number of leaves. The MLSTP was shown to be NP-hard by Garey and Johnson [13]. Fujie [12] provided two formulations and a detailed study of the facial structure of the arising polytopes. In 2010 Lucena et al. [30] presented additional formulations. Their first formulation is based on directed graphs which is an improvement of an approach from the previous literature. The second model reformulates the problem as a Steiner arboresence problem.

A closely related problem is the Minimum Connected Dominating Set Problem (MCDSP). A set $D \subseteq V$ of a graph $G = (V, E)$ is called a dominating set iff $\Gamma(D) = V$ for $\Gamma(D) = D \cup \{j \in V | \{i,j\} \in E, i \in D\}$. A dominating set is called connected iff the subgraph $G = (D, E(D))$ is connected for $E(D) = \{\{i,j\} \in E | i \in D, j \in D\}$. The goal of the MCDSP is to find a connected dominating set of minimum cardinality. It is well known that each solution of the MCDSP can be transformed into a solution of the MLSTP (see, e.g. [14]). In addition, Gendron et al. [14] provide two new approaches for the solution of the MCDSP using Benders Decomposition and Branch-and-Cut.

Chen et al. [6] observed that the RLP can be used to solve the MLSTP as well as a variant in which weights are assigned to the vertices. As a consequence this problem can also be solved by means of the NDPR.

1.2.3 Regenerator Placement Problem

Another closely related problem is the Regenerator Placement Problem (RPP). This problem does not restrict the maximum distance that might be covered without visiting a relay but the number of hops. Similar to the RLP it only considers edges of cost zero but the subset of nodes that need to communicate is an arbitrary subset of node pairs. The goal of the RPP is to minimize the number of used relays. Sen et al. [36] give an overview of this problem and present an efficient approximation algorithm. They also point out the importance of considering more general delay constraints. In fact there are various definitions of the RPP and many later versions also consider the same delay constraints as for the NDPR and the RLP. Flammini et al. [10] present complexity results and algorithms for several variants of the RPP.

The RPP with hop constraints can also be solved by NDPR-algorithms. We just have to use delays of one for the edges and then set the delay bound to the maximum number of allowed hops. To minimize the number of relays we assign the same cost to all of them. Thus, minimizing the costs is equivalent to minimizing the number of relays.

1.3 Aim of the Thesis

The NDPR has been solved efficiently using heuristic approaches. The exact solution methods, however, turned out to not work well in practice and are only able to deal with small or simple instances. The exact solution methods for the RLP and the GRLP on the other hand work well in practice but they are not directly applicable to the NDPR.

The aim of this thesis is to develop Mixed Integer Linear Programming (MILP) models that are also able to solve larger instances to provable optimality. We are going to apply the concept of communication graphs used in [6] and combine it with column generation. This will be the first set of solution approaches. Furthermore, we are going to apply layered graphs for the solution of the NDPR. For the development of the models we will also use common MILP techniques such as flow models and advanced techniques like Branch-and-Cut, Branch-and-Price and Branch-Price-and-Cut.

The models are implemented using CPLEX 12.6 (see [23]). In addition we will also use SCIP 3.1.0 (see [1]) to deal with the column generation approaches since CPLEX only allows for column generation in the root node. Developed models are going to be tested on a large set of benchmark instances.

1.4 Structure of the Thesis

We are going to start by developing some structural properties in Chapter 2. We will discuss the problem itself and explain the aspects that make it difficult. Furthermore, we will prove some properties about optimal solutions that will help to create tighter models. Then we are going to present our models for the solution of the NDPR. We start with the models based on layered graphs in Chapter 3 and then continue in the following chapter with the models on communication graphs. In Chapter 5 we will discuss some alterations to our models when

different solution properties are required. Finally we present the computational results on various test instances.

Note that we are not going to repeat the theoretical foundations of MILP as there is already a lot of excellent literature available. There are for example Schrijver [35], Nemhauser and Wolsey [32] and Bertsimas and Tsitsiklis [3]. In addition to the basic techniques we are also going to apply column generation which was introduced in Gilmore and Gomory [15, 16]. For additional literature on column generation and its application in the Branch-and-Bound tree (Branch-and-Price) we refer to Lübbecke and Desrosiers [29] and to Barnhart et al. [2].

CHAPTER 2

Structural Properties and a Basic MILP Model

In this chapter we are going to show certain structural properties concerning feasible and optimal solutions. These properties will be used later on to justify some of the used constraints. Furthermore, we present a first basic MILP model to discuss aspects that make MILP modeling of the NDPR difficult.

We start by presenting an exemplary instance to explain the basic structural properties. The graph shown in Figure 2.1 has three edges, their costs are the numbers in parentheses and the numbers left to the costs denote their delays. Relay costs are the numbers in parentheses to the right of the vertex numbers.

We set $d_{max} = 4$ and consider $\mathcal{K} = \{(0, 3)\}$. The unique optimal solution is to select all edges and to place a relay at node 2 as shown in Figure 2.2. The optimal path connecting 0 and 3 is then $(0, 1, 2, 1, 3)$. Note that this path is not simple as it uses the edge $\{1, 2\}$ twice and also visits the node 1 twice forming a cycle. The previous solution approaches for the NDPR allow such solutions and therefore they will also be considered as valid in the following.

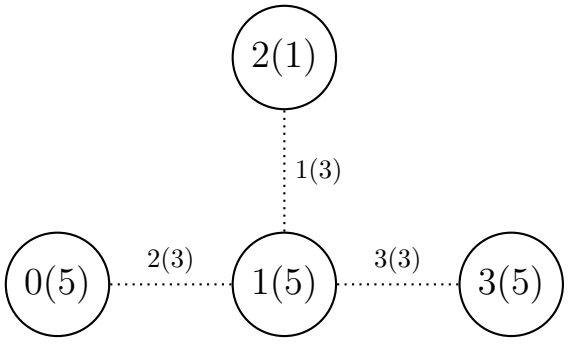


Figure 2.1: Example Instance

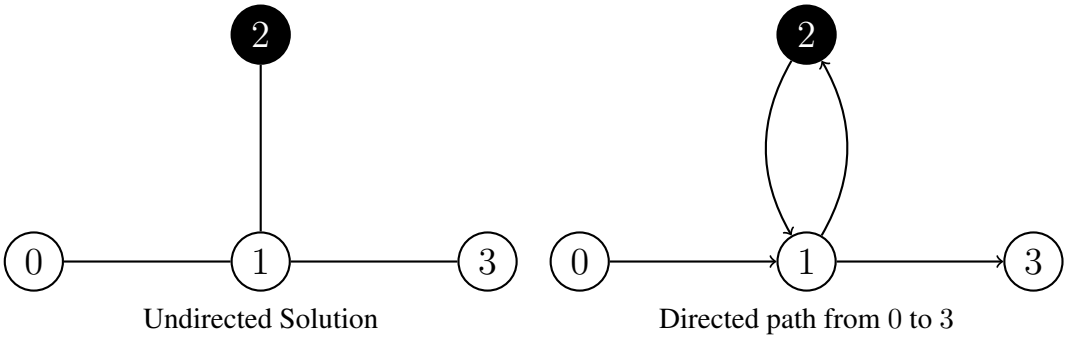


Figure 2.2: Cyclic Solution

2.1 Solution Characteristics

With the preceding problem instance in mind we are now going to prove some characteristics of optimal solutions for NDPR instances. By considering each source-target path as being directed, we may count the in-degree of each node along that path (see Figure 2.2). We start by studying the in-degree of relays. In the following we are going to show that it is not necessary to visit a relay more than once.

Lemma 2.1.1. *In an optimal solution there exists for every pair $(u, v) \in \mathcal{K}$ a path from u to v visiting each relay at most once.*

Proof. By the definition of an optimal solution there has to be a feasible path from u to v for each $(u, v) \in \mathcal{K}$. If the in-degree of all relays in these paths is at most one, we are done.

Now assume there exists a path from u to v for some $(u, v) \in \mathcal{K}$ visiting a relay r more than once, i.e., a path of the following shape: $(u, p_1, r, p_2, r, p_3, v)$ for subpaths p_1 to p_3. W.l.o.g. we assume that $r \notin p_1, r \notin p_3$.

Observe that p_1 reaches r and r reaches p_3. Hence, removing subpath (r, p_2, r) preserves connectivity. Furthermore, note that if we remove an entire segment between two relays this has no effect on the other segments. Thus, removing the subpath maintains feasibility and we obtain a modified path with only a single visit to r: (u, p_1, r, p_3, v).

If the path obtained this way still contains a relay with in-degree greater than one we may iterate this procedure until all superfluous visits to relays have been removed. Note that we only remove edges and nodes from the path and never increase the in-degree of any vertex, thus termination of the procedure is guaranteed. □

According to the instance described in the previous section it makes sense to visit a non-relay vertex a second time. In the following we are going to show that we do not gain anything by further visits.

Lemma 2.1.2. *In an optimal solution there exists for every pair $(u, v) \in \mathcal{K}$ a path from u to v visiting each non-relay vertex at most twice.*

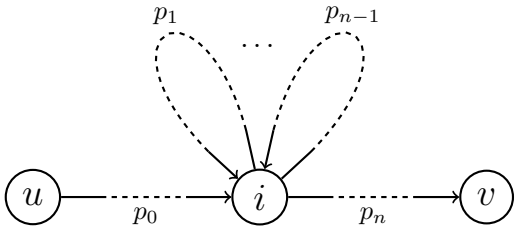


Figure 2.3: Path from u to v visiting vertex i n times

Proof. By the definition of an optimal solution there has to be a feasible path from u to v for each $(u, v) \in \mathcal{K}$. In the following we are going to show that whenever a non-relay vertex $i \in V$ is visited more than twice on such a path we may reduce it to a feasible path visiting i at most twice.

Assume that the vertex i is visited $n > 2$ times. Then there have to be $n-1$ cycles[1] resulting in path $p = (u, p_0, i, p_1, i, \ldots, p_{n-1}, i, p_n, v)$ with subpaths $\{p_0, p_1, \ldots, p_n\}$ as shown in Figure 2.3. Now assume subpath p_i contains no relay then the delay of the segment that contains p_i will stay the same or decrease if we remove the cycle. Thus, we can safely remove all such cycles without affecting feasibility of p. If the in-degree of i is now smaller than or equal to two we are done. Otherwise we consider the remaining cycles $\{p_1, \ldots, p_k\}$ that contain at least one relay. $(2 \leq k \leq n-1)$

Simply removing a cycle containing relays causes two consecutive segments to merge and the delay of the emerging segment might be larger than d_{max}. In the following we are going to show how to modify the path s.t. only one of the relays in cycles $\{p_1, \ldots, p_k\}$ is visited and s.t. the path stays feasible.

Among the remaining cycles we choose the relay r^* closest to i w.r.t. the delay over all relays in subpaths $\{p_1, \ldots, p_k\}$, i.e., either $\Delta((i, q', r^*))$ or $\Delta((r^*, q'', i))$ is minimal for subpaths q', q''. Note that since we are dealing with edges we can use this minimal subpath to reach and to leave r^*. Let q be the subpath used in the minimal connection then we construct a path from u to v as follows: $p' = (u, p_0, i, q, r^*, q, i, p_n, v)$. Note that this path only uses a subset of the edges used in the original path and that it connects u and v. Moreover, it fulfills the desired condition and visits i only twice. However, we still have to show that this path is also feasible.

Let r_0 be the last relay in p_0 or u if such a relay does not exist, r_1 the first relay in p_1, r_k the last relay in p_k and r_n the first relay in p_n or v if such a relay does not exist. In the original path the segments between r_0 and r_1 and r_k and r_n are feasible by definition. Note that the subpaths from r_0 to i and from i to r_k stay the same in the modified path. We chose r^* to be the relay closest to i. Hence, the subpaths from i to r_1 and from r_k to i both have a delay at least as high as when using (i, q, r^*) or (r^*, q, i) respectively. Thus, the new path has to be feasible. Note that if the delay of the subpath between r_0 and i is smaller than or equal to the delay of the subpath (r^*, q, i) we can even remove the cycle (i, q, r^*, q, i).

[1] Note that these cycles do not have to be simple.

We iteratively apply this procedure to all non-relay vertices that are visited more than twice to finally obtain a path fulfilling the proposed condition.

Note that if either (i, q, r^*) or (r^*, q, i) is not contained in p, adding it increases the in-degree of the vertices in q. W.l.o.g. we assume that subpath (i, q, r^*) is not used in p. Then, there has to be a subpath $(i, \hat{q}, r^*)$ s.t. $\Delta((i, \hat{q}, r^*)) \geq \Delta((i, q, r^*))$. Since $k \geq 2$ there has to be at least one further relay r_x reached by some path (i, q_x, r_x) s.t. $\Delta((i, q_x, r_x)) \geq \Delta((i, q, r^*))$. The only subpath we add is (i, q, r^*) and we remove at least subpaths $(i, \hat{q}, r^*)$ and (i, q_x, r_x). Therefore, $\Delta(p') < \Delta(p)$ holds, i.e., the delay of the path strictly decreases each time we process a vertex. Note that we only allow positive delays by definition. Thus, the total delay of the path is bounded by zero, guaranteeing termination. □

From the proof of Lemma 2.1.2 we conclude that the following condition holds:

Corollary 2.1.1. *In an optimal solution there exists for every pair $(u, v) \in \mathcal{K}$ a path from u to v visiting each non-relay vertex at most twice s.t. the delay of the second visit will be strictly smaller than the delay of the first visit.*

In Lemmas 2.1.1 and 2.1.2 we have shown two properties independently. In the following we are going to prove that for an optimal solution there exists a path s.t. both hold simultaneously.

Theorem 2.1.1. *In an optimal solution there exists for every pair $(u, v) \in \mathcal{K}$ a path from u to v visiting each relay at most once and each non-relay vertex at most twice.*

Proof. First of all, by the definition of an optimal solution there has to be a feasible path from u to v for each $(u, v) \in \mathcal{K}$. Furthermore, the proofs of Lemmas 2.1.1 and 2.1.2 showed how an arbitrary connection can be transformed into a connection with the respective properties. We start with the procedure described in Lemma 2.1.2 and then apply the procedure from Lemma 2.1.1. Note that the latter does not increase the in-degree of any vertex. Thus, the resulting path fulfills both properties. □

Finally, we are going to prove that these properties do not only hold for a single commodity. They also hold if we consider all commodities with a common source at the same time.

Theorem 2.1.2. *In an optimal solution there exists for every source $u \in \mathcal{K}_S$ a directed graph rooted at u having a feasible connection to every target $v \in K(u)$ visiting each relay at most once and each non-relay vertex at most twice.*

Proof. We prove the existence of such a graph $\hat{G}$ by construction. Let u be the root of our graph. Then, from Theorem 2.1.1 it follows that for each target $v \in K(u)$ there exists a feasible path from u to v with the desired properties.

Initially we set graph $\hat{G}$ equivalent to a path to one of the targets $v \in K(u)$. At this point the graph has the desired properties according to the definition of our paths. Then, we continue to iteratively insert paths for the remaining targets. Whenever we add a path there are three possibilities:

1. the graph already contains the path's target

   In that case, we do not extend $\hat{G}$ since we already reach the target in the graph.

2. the graph and the path are vertex-independent (except for the source)

   In that case we simply insert the path to $\hat{G}$ because it does not change the in-degree of any vertex. Thus, the new graph still has the required properties.

3. the graph and the path are not vertex-independent (except for the source)

The third case is more difficult since simply adding the path would increase the in-degree for some vertices and thus can destroy some of the properties we need. Thus, we only add parts of the path ensuring connection to the paths' target. Let $p = (u, p_1, z_1, \ldots, p_n, z_n, p_{n+1}, v)$ be the path to add, having n vertices z_1 to z_n in common with the graph $\hat{G}$. Let d_i^p be the delay of the subpath from the preceding relay (or the source) to z_i in p and let d_i^G be the delay of the subpath from the preceding relay (or the source) to z_i in $\hat{G}$.

We start at the common vertex farthest from the source, i.e., z_n. If $d_{z_n}^{\hat{G}} \leq d_{z_n}^p$ we can simply add the sub-path (z_n, p_{n+1}, v) to the graph. Since we arrive in $\hat{G}$ at z_n with smaller delay than on the path, we know that the new segment will be valid according to d_{max}. Furthermore we do not increase the in-degree of any vertex and reach the additional target v. Thus, the graph still has the desired properties. Note that we can also add this subpath if the emerging segment remains valid w.r.t. the delay bound.

If $d_{z_n}^{\hat{G}} > d_{z_n}^p$ and the new segment has a delay larger than d_{max} we also add the sub-path (z_n, p_{n+1}, v) to the graph. However, this results in a delay violation. Thus, we have to perform additional modifications to maintain feasibility. Let x be the closest predecessor of z_n in $\hat{G}$ having either an out-degree greater than one or being a target of u. If $n = 1$ and no predecessor of the described kind exists we set $x = u$. We then consider the sub-path (x, q, z_n) of $\hat{G}$. Next we remove (x, q, z_n) from $\hat{G}$ and add (z_{n-1}, p_n, z_n) instead. Note that again we do not increase the in-degree of any vertex and thus, maintain the required properties. However, the graph only remains valid w.r.t. d_{max} if $d_{z_{n-1}}^{\hat{G}} \leq d_{z_{n-1}}^p$. If this condition does not hold we have to iterate the described procedure until either we arrive at some common vertex z_i s.t. $d_{z_i}^{\hat{G}} \leq d_{z_i}^p$ or until we reach the source ($d_u^{\hat{G}} = d_u^p = 0$).

After we have processed all (u, v)-paths we know that u reaches all its targets in $\hat{G}$. Furthermore, none of our transformations destroyed the required properties. Hence we know that in the final graph every relay is visited at most once and every other vertex at most twice. □

Moreover, we are going to show that in an optimal solution an edge never needs to be traversed more than once in the same direction:

Theorem 2.1.3. *In an optimal solution there always exists a path connecting each $(u, v) \in \mathcal{K}$ without traversing an edge more than once in the same direction.*

Proof. Assume there exists a path $p = (u, p_1, i, j, p_2, i, j, p_3, v)$ with subpaths $\{p_1, p_2, p_3\}$ connecting the pair (u, v) using edge $\{i, j\}$ twice in the same direction. In the following we are going to show how this path can be modified to use the edge only once per direction without affecting optimality and feasibility.

First of all we apply Theorem 2.1.1: In an optimal solution there exists for every $(u, v) \in \mathcal{K}$ a feasible connection from u to v visiting every relay vertex at most once and every non-relay

vertex at most twice. Therefore, we assume that i and j are contained in none of the subpaths $\{p_1, p_2, p_3\}$ and neither of them is a relay.

If the delay of the first visit to j is smaller than or equal to the delay of the second visit then path (u, p_1, i, j, p_3, v) also has to be feasible and we are done. Note that we may also use this path if the second visit to i has larger delay than the first visit. In the following we assume that the delay of the second visit is strictly smaller. Thus, p_2 has to contain at least one relay. Therefore, we now consider the subpath $p_2 = (i, j, q_1, r_1, q_2, \ldots, r_n, q_{n+1}, i, j)$ for n relays $\{r_1, r_n\}$ and subpaths $\{q_1, \ldots, q_{n+1}\}$.

Assume $\Delta((j, q_1, r_1)) \leq \Delta((r_n, q_{n+1}, i))$ then $\Delta((j, q_1, r_1)) < \Delta((r_n, q_{n+1}, i, j))$ and therefore $(u, p_1, i, j, q_1, r, q_1, j, p_3, v)$ has to be feasible. If the opposite holds, i.e., $\Delta((j, q_1, r_1)) > \Delta((r_n, q_{n+1}, i))$ then also $\Delta((i, j, q_1, r_1)) > \Delta((r_n, q_{n+1}, i))$ holds and path $(u, p_1, i, q_{n+1}, r_n, q_{n+1}, i, j, p_3, v)$ has to be feasible.

Thus, in either case we obtain a feasible path using edge $\{i, j\}$ only once per direction. Furthermore, the new path uses a subset of the edges of the original path preserving optimality.

If there are multiple violations of the this kind we may apply the described procedure repeatedly until we obtain the desired path.

Observe that if $\Delta((j, q_1, r_1)) \leq \Delta((r_n, q_{n+1}, i))$ we add subpath (r_1, q_1, j). This can create additional violations. Note that we remove at least path (r_n, q_{n+1}, i, j) and $\Delta((j, q_1, r_1)) < \Delta((r_n, q_{n+1}, i, j))$. The opposite case is symmetric. As a result, the delay of the path strictly decreases during each iteration. Furthermore, to maintain the condition that each non-relay vertex is visited at most twice we might have to apply the procedure described in Lemma 2.1.2[2]. Note that this procedure also guarantees a strict decrease of the paths delay. Since we are only dealing with positive delays we know that the delay of a path is bounded by zero. Hence, termination is guaranteed. □

2.2 Basic MILP Model

To illustrate some difficulties w.r.t. MILP modeling of the NDPR we are going to start with a basic model. The model uses only a simple graph transformation and uses only polynomially many variables. We also tested this model in practice. However, it has only been able to solve instances of few nodes and edges. We use it here only for illustrative purposes.

To ensure that d_{max} is not exceeded, we want to keep track of the delay at every node. According to Corollary 2.1.1, a vertex might be visited twice with different delays. To deal with this problem we split every vertex v into two copies v_1 and v_2. We use v_1 for the initial visit and v_2 if the vertex is visited again. In addition we need arcs between the vertex copies as shown in Figure 2.4 to enable all possible connections.

Each of the shown connections corresponds to a certain case. The arcs between the vertex copies ((u_1, u_2) and (v_1, v_2)) are relay arcs, i.e., if one of these arcs is used this means that we have to place a relay at the corresponding vertex. We arrive at vertex v_1 with a certain delay then we go directly to its copy v_2 and the new delay at v_2 becomes zero. Connections of the type (u_1, v_1) correspond to the most basic case (the first visit of u and v). (u_1, v_2) has to be used if we

[2]Note that together with the initial assumption this also guarantees that in the final path every relay vertex is visited at most once and every non-relay vertex at most twice.

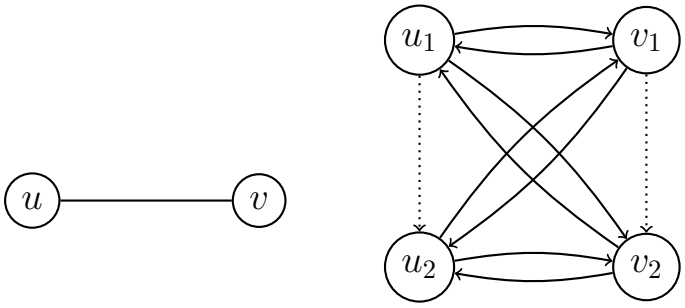


Figure 2.4: Transformation of a single edge

continue after the first visit of u but we already visited v before and now need a different delay value. Note that the delay at v_2 always has to be smaller than the delay at v_1 since otherwise we do not get an improvement by visiting the vertex a second time. (u_2, v_1) means that u has been visited for the second time but we visit v for the first time. Finally, (u_2, v_2) corresponds to the case that u has been visited for the second time and we also visit v a second time. Thus, we get a new graph $G'_{\mathrm{D}} = (V'_{\mathrm{D}}, A'_{\mathrm{D}})$:

$$
\begin{aligned}
V'_{\mathrm{D}} =& \{v_1, v_2 | v \in V\} \\
A^r_{\mathrm{D}} =& \{(v_1, v_2) | v \in V\} \\
A'_{\mathrm{D}} =& \{(u_1, v_1), (u_1, v_2), (u_2, v_1), (u_2, v_2), (v_1, u_1), (v_1, u_2), (v_2, u_1), (v_2, u_2) | \{u, v\} \in E\} \\
& \cup A^r_{\mathrm{D}}
\end{aligned}
$$

Figure 2.5 depicts the graph obtained using the instance shown in Figure 2.1 (relay arcs in dotted lines).

When stating the model we have to take into account that the delay value at every vertex depends on its predecessor. Therefore, we require at least one set of variables per source $u \in \mathcal{K}_S$ to identify the predecessor uniquely. To ensure that all commodities are able to communicate we use cut constraints. The model uses delay variables d^u_i per source $u \in \mathcal{K}_S$ for every vertex $i \in V'_{\mathrm{D}}$. Furthermore, we require binary variables y_i and x_e to represent relays $i \in V$ and augmenting edges $e \in E^*$. Moreover, we use binary arc variables X^u_a per source $u \in \mathcal{K}_S$ for every arc $a \in A'_{\mathrm{D}}$.

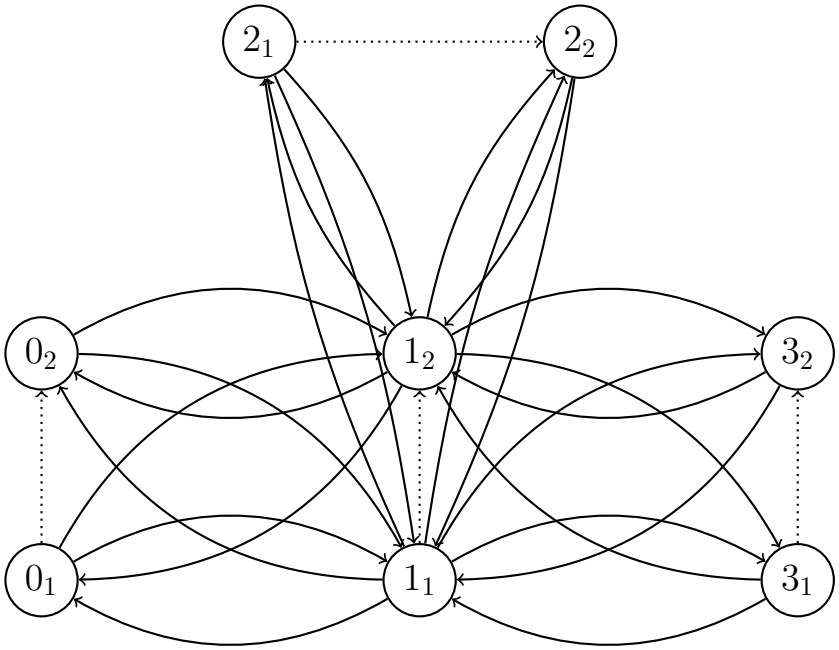


Figure 2.5: Graph $G_{\mathrm{D}}{}' = (V_{\mathrm{D}}{}', A_{\mathrm{D}}{}')$ corresponding to the instance in Figure 2.1

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^-(W)} X^u_a \geq 1 \qquad \forall u \in \mathcal{K}_S, \forall W \subset V'_{\mathrm{D}}, \exists v \in K(u) : \{v_1\} \subseteq W, \{u_1, u_2\} \cap W = \emptyset \quad (2.1)$$

$$d^u_{i_l} + d(i,j) \cdot X^u_{(i_l,j_m)} \leq d^u_{j_m} + d_{max} \cdot (1 - X^u_{(i_l,j_m)}) \qquad \forall u \in \mathcal{K}_S, \forall (i_l, j_m) \in A'_{\mathrm{D}} : j \neq u \quad (2.2)$$

$$d^u_{i_l} + d(i,j) \cdot X^u_{(i_l,j_m)} \geq d^u_{j_m} - d_{max} \cdot (1 - X^u_{(i_l,j_m)}) \qquad \forall u \in \mathcal{K}_S, \forall (i_l, j_m) \in A'_{\mathrm{D}} : j \neq u \quad (2.3)$$

$$d^u_{i_2} \leq d_{max} \cdot (1 - X^u_{(i_1,i_2)}) \qquad \forall u \in \mathcal{K}_S, \forall i \in V : i \neq u \quad (2.4)$$

$$d^u_{u_1} = 0 \qquad \forall u \in \mathcal{K}_S \quad (2.5)$$

$$\sum_{a \in \delta^-(v_1)} X^u_a = 1 \qquad \forall u \in \mathcal{K}_S, \forall v \in K(u) \quad (2.6)$$

$$\sum_{a \in \delta^-(i_1)} X^u_a \leq 1 \qquad \forall u \in \mathcal{K}_S, \forall i \notin K(u) \quad (2.7)$$

$$\sum_{a \in \delta^-(i_2)} X^u_a \leq 1 \qquad \forall u \in \mathcal{K}_S, \forall i \in V \quad (2.8)$$

$$X^u_{(i_1,i_2)} \leq y_i \qquad \forall u \in \mathcal{K}_S, i \in V : i \neq u \quad (2.9)$$

$$\sum_{l \in \{1,2\}, m \in \{1,2\}} X^u_{(i_l,j_m)} \leq x_e \qquad \forall u \in \mathcal{K}_S, \forall e \in E^*, \forall (i,j) \in A(e), \quad (2.10)$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \quad (2.11)$$

$$X^u_a \in \{0,1\} \qquad \forall u \in \mathcal{K}_S, \forall a \in A'_{\mathrm{D}} \quad (2.12)$$

$$0 \leq d^u_i \leq d_{max} \qquad \forall u \in \mathcal{K}_S, \forall i \in V'_{\mathrm{D}} \quad (2.13)$$

The first set of constraints are the cut inequalities. Every set W of arc variables containing a target copy v_1 for some $v \in K(u)$ requires an in-coming arc. Since $\{u_1, u_2\} \cap W = \emptyset$ this ensures that all targets are connected to their corresponding source. Constraints (2.2) to (2.5) ensure that the delay variables are set correctly. The first two of these constraints set an upper and a lower bound on the delay w.r.t. the preceding vertex. If an in-coming arc is selected the upper and lower bound are the same. Otherwise, they have no effect and leave the bound of $0 \leq d_i^u \leq d_{max}$ unchanged. Constraints (2.4) reset the delay to zero when a relay is placed on the vertex and the next set of constraints ensures that we start with a delay of zero at the source. Constraints (2.6) state that each target has exactly one in-coming arc on copy v_1, the initial copy of the other vertices might have at most one in-coming arc (2.7). Furthermore, all vertices might be visited a second time (2.8). The last two constraints link the arc variables to the relay and edge variables. Note that Constraints (2.10) also ensure that per directed arc of an edge and per source, only one of the four variants can be used.

In addition to the required constraints we also use the following optional ones:

$$d^u_{i_2} \leq d^u_{i_1} \qquad \forall u \in \mathcal{K}_S, \forall i \in V : i \neq u \tag{2.14}$$

$$\sum_{a \in \delta^-(i_l)} X^u_a \geq X^u_{(i_l,j_1)} + X^u_{(i_l,j_2)} \qquad \begin{array}{r} \forall u \in \mathcal{K}_S, \forall i \in V'_{\mathrm{D}}, \forall l \in \{1,2\}, \\ \forall j \in \delta^+(i_1), i \neq u \end{array} \tag{2.15}$$

$$\sum_{a \in \delta^-(i_1)} X^u_a \geq \sum_{a \in \delta^-(i_2)} X^u_a \qquad \forall u \in \mathcal{K}_S, \forall i \in V : i \neq u \tag{2.16}$$

$$\sum_{l \in \{1,2\}} \sum_{i_m \in \delta^-(u_l)} X^u_{(i_m,u_l)} = 0 \qquad \forall u \in \mathcal{K}_S \tag{2.17}$$

Constraints (2.14) represent the results of Corollary 2.1.1: If a vertex is visited twice then the delay of the second visit is smaller than the delay of the first visit. Since the delay of the second visit is even strictly smaller we may add one to the left-hand side if we are dealing with integral delays. Constraints (2.15) ensure that a node only has out-going arcs if it has in-coming arcs and that it only targets one of the two vertex copies per original edge. Constraints (2.16) guarantee that the vertex for the second visit is only used if the initial vertex copy has already been visited. Constraints (2.17) ensure that the source never has in-coming arcs on any of its copies. This especially prevents having a relay on the source.

In the following we are going to show an optimal solution in graph $G_{\mathrm{D}}{}'$. We consider $\mathcal{K} = \{(0,3)\}$ and $d_{max} = 4$ for the instance in Figure 2.1. The corresponding optimal solution is depicted in Figure 2.6. We display the values of the delay-variables at every vertex that is reached by some arc.

We start at the commodity source 0_1 with a delay of zero. Then we continue to the initial copy of vertex 1. At this point we cannot reach target vertex 3 without violating the delay bound (since edge $(1,3)$ has a delay of three). Thus, we need to place a relay. To avoid the high cost for a relay at vertex 1 we instead proceed to vertex 2. Here we use the arc $(2_1, 2_2)$ to reset the delay. Hence, we have to place a relay at vertex 2. Now we return to vertex 1. Since we already visited this vertex we have to use copy 1_2. Due to the lower delay we are now able to reach the target node within the delay bound.

The presented MILP model requires a large number of constraints to set the delay variables. This is caused by the fact that the delays are recursive, i.e., the delay at some vertex depends on its predecessor. Unfortunately, the predecessor is chosen dynamically. Naturally, this would

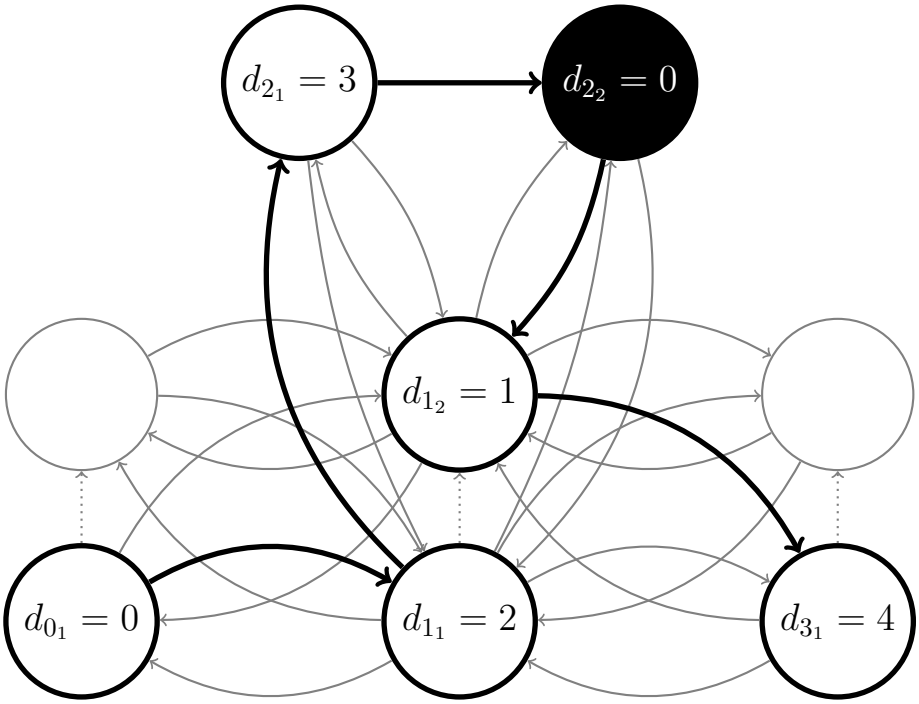


Figure 2.6: Optimal solution for $\mathcal{K} = \{(0, 3)\}$, $d_{max} = 4$

require quadratic constraints of the form $\sum_{(i,j)\in\delta^-(j)} \left((d_i^u + d(i,j))X_{ij}^u\right)$. The linearization of these quadratic constraints using Big-M values induces weak Linear Programming (LP) bounds. With this problem in mind it is desirable to avoid delay variables. The models mentioned in the previous chapter accomplish this by using variables that represent whole paths. Since the paths are known, it is possible to state constraints to enforce a relay placement that turns the path into a feasible connection. The use of path variables implicitly ensures connectivity between source and target of a commodity. The model above requires additional constraints, i.e., the cut constraints, to ensure connectivity.

Although formulations with path variables are able to address both issues at once, the overhead by the required column generation may be high. In the following we are going to deal with both problems separately. We will use graph transformations to deal with the delay bound indirectly and then add constraints (like the cut constraints in the model above) to ensure that all commodities can communicate.

CHAPTER 3

Models on Layered Graphs

The models presented in this chapter use extended, so called layered graphs. This type of graph encodes the delays in its structure which contains only paths that are feasible w.r.t. the delay bound d_{max}.

In the following we are assuming integral delays, again we may deal with rational delays by means of scaling.

Picard and Queyranne [33] were among the first to consider layered graphs in 1978. They use this technique to solve the time-dependent traveling salesman problem. More recent applications of layered graphs by Godinho et al. include [17] for the solution of unit demand vehicle routing problems, [18] for solving the ATSP and in [19] the approach of Picard and Queyranne is extended. Further applications involve Ljubić and Gollowitzer [28] for the solution of the hop constrained connected facility location problem and Gouveia et al. [21] for the solution of hop-constrained and diameter-constrained minimum spanning tree problems. Moreover, the technique has been applied to various tree problems in Ruthmair [34]. An application dealing with multiple layered graphs includes Gouveia et al. [22].

3.1 Model on a Single Layered Graph

We start by providing a model that is based on a single layered graph. First we give a formal definition of the graph and then we discuss the model.

3.1.1 Definitions

Given a graph $G = (V, E, d)$ with delays $d : E \to \mathbb{N}^+$ the layered digraph $G_{\mathrm{L}} = (V_{\mathrm{L}}, A_{\mathrm{L}})$ is defined as follows:

$$V_{\mathrm{L}}^0 = \{v_0 | v \in V\}$$
$$V_{\mathrm{L}}^l = \{v_l | u_m \in V_{\mathrm{L}}^{l-1}, v \in \delta^+(u), m + d(u,v) = l\} \cup V_{\mathrm{L}}^{l-1}$$
$$V_{\mathrm{L}} = V_{\mathrm{L}}^{d_{max}}$$

The set V_{L}^l contains all vertex copies on layers smaller than or equal to l. We define this set recursively. Layer zero contains copies of all vertices $v \in V$. Set V_{L}^l contains all vertices that can be reached with a total delay of l starting at existing vertices on lower layers. To avoid exceeding the delay bound only vertex copies on layers smaller than or equal to the delay bound are considered. Thus, we define $V_{\mathrm{L}} = V_{\mathrm{L}}^{d_{max}}$.

$$A_{\mathrm{L}}^r = \{(i_l, i_0) | i_l \in V_{\mathrm{L}}, l > 0\}$$
$$A_{\mathrm{L}}^a = \{(i_l, j_m) | i_l \in V_{\mathrm{L}}, j_m \in V_{\mathrm{L}}, \{i,j\} \in E, d(i,j) = m - l\}$$
$$A_{\mathrm{L}} = A_{\mathrm{L}}^a \cup A_{\mathrm{L}}^r$$

The arcs in A_{L}^r target layer zero. They correspond to the case of placing a relay at the respective node. The arcs in A_{L}^a connect vertices on different layers w.r.t. to their delays. Vertices have to be connected if there is an edge between them in the original graph and its delay matches the difference between the layers of the vertices. These arcs are always directed from a lower to a higher layer.

Figure 3.1 shows the layered graph corresponding to the instance given in Figure 2.1 and the optimal solution w.r.t. $\mathcal{K} = \{(0,3)\}$. The arcs in A_{L}^r are depicted in dashed lines and the remaining arcs in solid lines. Note that a path starting at layer zero using only arcs in A_{L}^a corresponds to a segment. As soon as an arc in A_{L}^r is used we return to layer zero and start a new segment. Due to the restriction of the allowed vertex copies a segments delay can never exceed the delay bound.

Different Layer Structure

In the layered graph defined above we use a uniform way to reset the delay at relays, i.e., we use a separate arc to return to layer zero and continue from here to the next vertex. Instead of using the arcs in A_{L}^r it is also possible to target the next vertex directly. Hence we reach the vertex on the layer corresponding to the delay of the used arc. The modified set of arcs looks as follows:

$$A_{\mathrm{L}}^{r\prime} = \{(i_l, j_m) | i_l \in V_{\mathrm{L}}, l > 0, \{i,j\} \in E, d(i,j) = m\}$$
$$A_{\mathrm{L}}' = A_{\mathrm{L}}^a \cup A_{\mathrm{L}}^{r\prime}$$

This gives a different layered graph $G_{\mathrm{L}}{}' = (V_{\mathrm{L}}, A_{\mathrm{L}}')$, see Figure 3.2 for an example. This layered graph contains more arcs than the previous one. G_{L} requires one relay arc per vertex copy on a layer greater than zero. However, in $G_{\mathrm{L}}{}'$ we require one relay arc per incident edge for each vertex copy on a layer greater than zero, i.e. the worst case complexity increases from $O(|V_{\mathrm{L}}|)$ to $O(|V_{\mathrm{L}}| \cdot |E|)$. We also tested variants of the following models based on this graph, but they turned out to be less efficient so we decided to stick to the former variant.

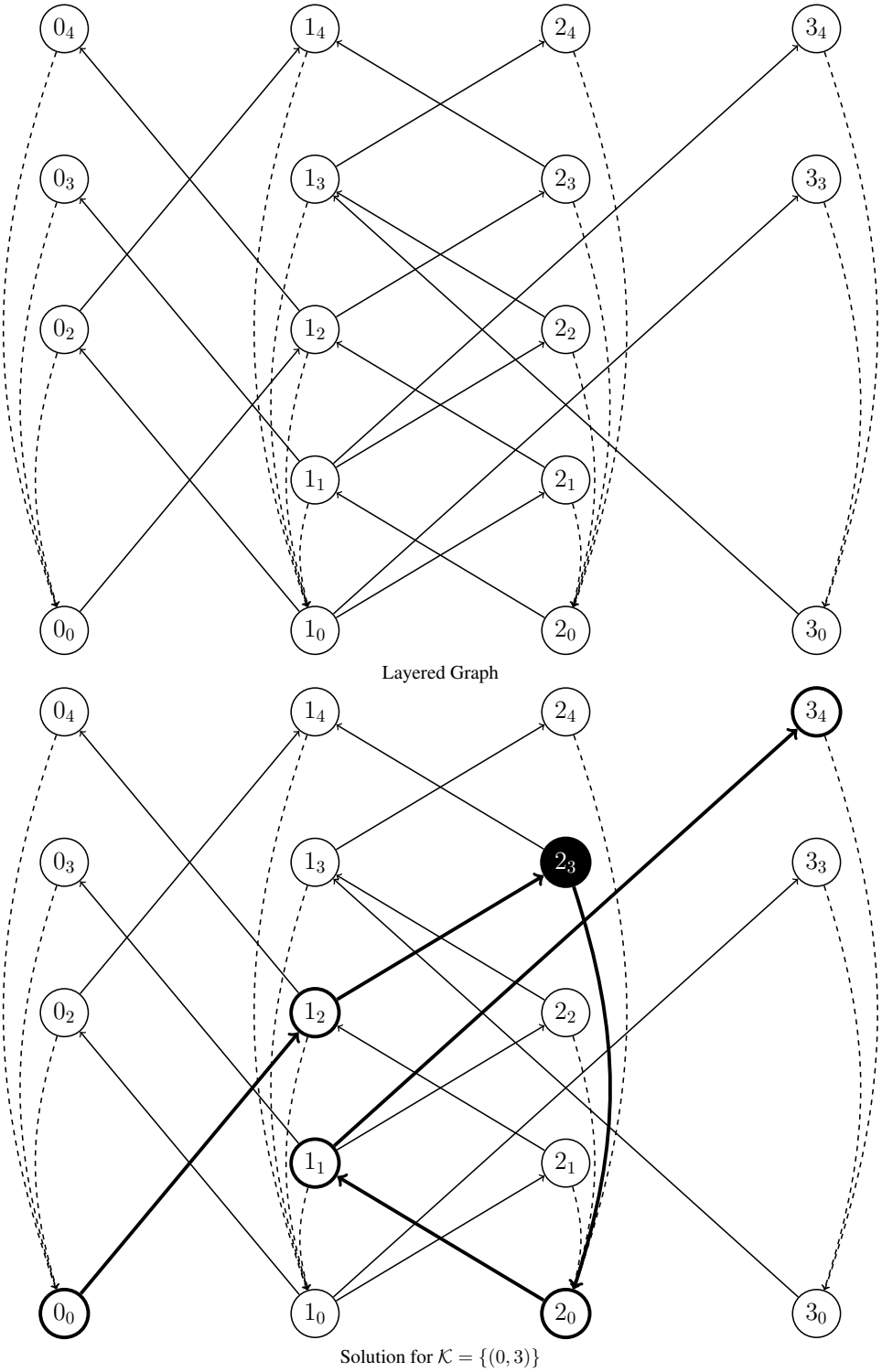


Figure 3.1: Layered graph $G_{\mathrm{L}} = (V_{\mathrm{L}}, A_{\mathrm{L}})$ corresponding to the instance in Figure 2.1 for $d_{max} = 4$

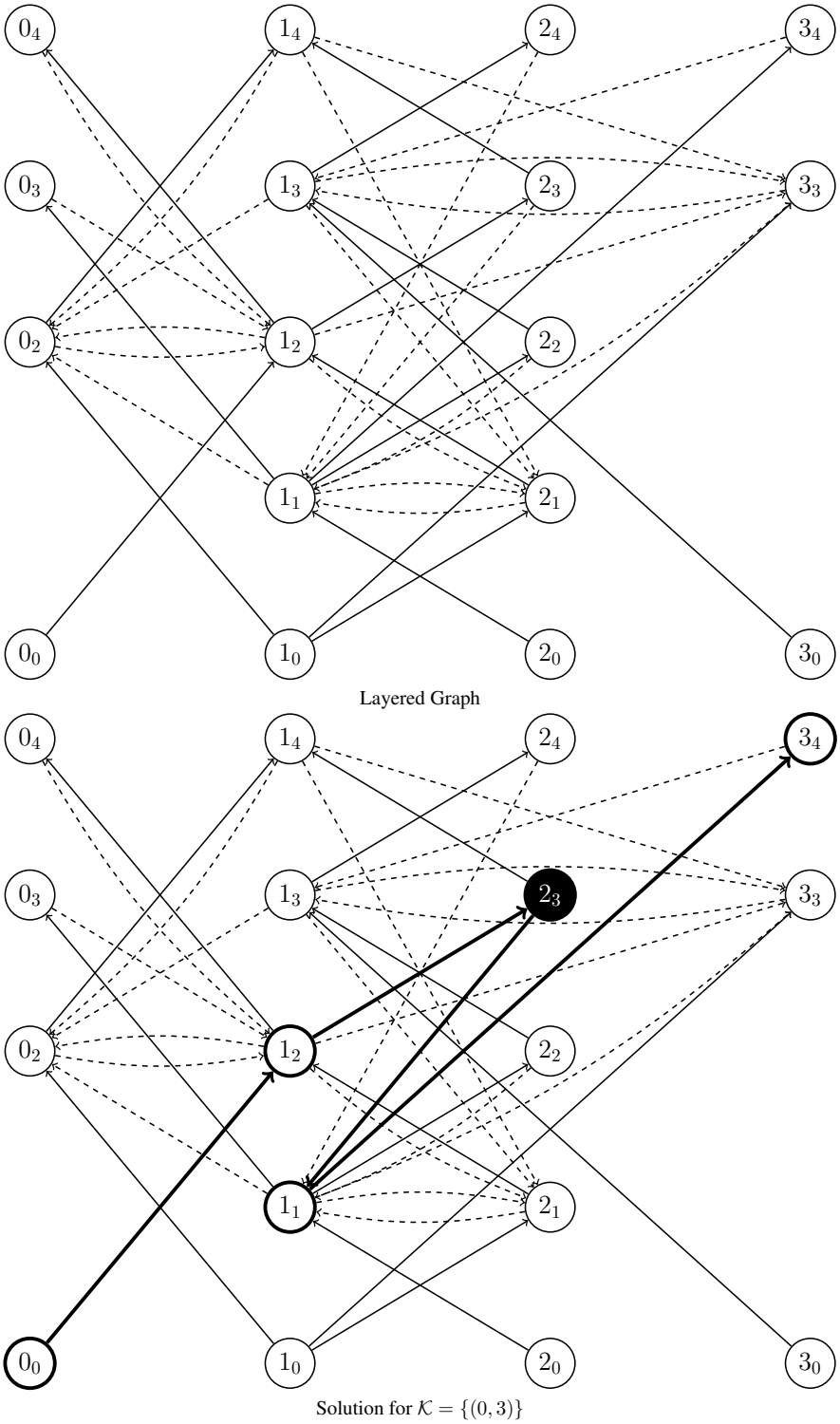


Figure 3.2: Layered graph $G_{\mathrm{L}}{}' = (V_{\mathrm{L}}{}', A_{\mathrm{L}}{}')$ corresponding to the instance in Figure 2.1 for $d_{max} = 4$

3.1.2 Cut Model on a Single Layered Graph

In this section we introduce our so called layered cut formulation on a single layered graph to which we will refer as L_{CUTS}. We use binary arc variables X_a for the layered arcs in $a \in A_{\mathrm{L}}$. In addition we use binary variables $x_e, \forall e \in E^*$, to map the layered arcs to the original edges. Furthermore, we require binary variables $y_i, \forall i \in V$, to identify the relays.

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^-(W)} X_a \geq 1 \qquad \begin{array}{r} \forall u \in \mathcal{K}_S, \forall W \subset V_{\mathrm{L}}, \\ \exists v \in K(u) : \{v_l | v_l \in V_{\mathrm{L}}, l > 0\} \subseteq W, \\ W \cap \{u_l | u_l \in V_{\mathrm{L}}\} = \emptyset \end{array} \tag{3.1}$$

$$X_{(i_l, i_0)} \leq y_i \qquad \forall (i_l, i_0) \in A^r_{\mathrm{L}} \tag{3.2}$$

$$X_{(i_l, j_m)} \leq x_{\{i,j\}} \qquad \forall (i_l, j_m) \in A^a_{\mathrm{L}}, \{i, j\} \in E^* \tag{3.3}$$

$$X_a \in \{0, 1\} \qquad \forall a \in A_{\mathrm{L}} \tag{3.4}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{3.5}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E^* \tag{3.6}$$

The first set of constraints are the cut-inequalities, ensuring that all commodities are able to communicate. Each subset of the vertices containing all copies of some target $v \in K(u)$ has to be connected to the corresponding source u. Note that due to the dependence on subsets the number of these constraints is in general exponential. Corresponding separation methods will be discussed in Chapter 6. Constraints (3.2) link the relay arcs to the relay variables and Constraints (3.3) link the layered arcs to the original augmenting edges. Note that although A^a_{L} contains augmenting and free arcs we only need linking constraints for the augmenting edges since the free edges have no influence on the objective.

In addition to the required constraints we add the following optional constraints:

$$\sum_{\forall (i_l, j_m) \in A^a_{\mathrm{L}}} X_{(i_l, j_m)} \leq |\delta^+(i_l)| \cdot (1 - y_i) \qquad \forall i_l \in V_{\mathrm{L}}, l > 0 \tag{3.7}$$

$$\sum_{a \in \delta^+(i_l)} X_a \leq |\delta^+(i_l)| \cdot \sum_{a \in \delta^-(i_l)} X_a \qquad \forall i_l \in V_{\mathrm{L}}, l > 0 \tag{3.8}$$

Constraints (3.7) are useful to avoid symmetries. They state that whenever we place a relay on some vertex we always have to use it. This means that we do not use any arcs in A^a_{L} starting at vertex copy of $i \in V$ on a layer greater than zero if i is a relay. Constraints (3.8) ensure that a vertex only has out-going arcs if it has an in-coming arc.

To reduce the number of dynamically generated cuts we add for every source $u \in \mathcal{K}$ constraints corresponding to sets $W = \{v_l | v_l \in V_{\mathrm{L}}, l > 0\}$ for all targets $v \in K(u)$ in advance using the fact that every target has to be reached on some layer greater than zero:

$$\sum_{v_l \in V_{\mathrm{L}} : l > 0} \sum_{a \in \delta^-(v_l)} X_a \geq 1 \qquad \forall u \in \mathcal{K}_S, \forall v \in K(u)$$

This model uses only one set of arc variables. As a consequence we have to deal with all pairs in $\mathcal{K}$ at the same time. In the following we are going to develop models that use multiple graphs and disaggregation of commodities.

3.2 Models on Multiple Layered Graphs

We now generate one graph $G^u_{\mathrm{L}} = (V^u_{\mathrm{L}}, A^u_{\mathrm{L}})$ per unique source $u \in \mathcal{K}_S$. To define G^u_{L} we delete a subset of vertices and their incident edges from G_{L}.

$$
\begin{aligned}
V^u_{\mathrm{R}} &= \{u_l | l > 0\} && \forall u \in \mathcal{K}_S \\
V^u_{\mathrm{L}} &= V_{\mathrm{L}} \setminus V^u_{\mathrm{R}} && \forall u \in \mathcal{K}_S \\
A^u_{\mathrm{L}} &= A_{\mathrm{L}} \setminus \{(i,j) | (i,j) \in A_{\mathrm{L}}, i \in V^u_{\mathrm{R}} \vee j \in V^u_{\mathrm{R}}\} && \forall u \in \mathcal{K}_S
\end{aligned}
$$

When dealing with a single source we know that this source requires no in-coming arcs. Since we start with a delay of zero at the source we cannot reach it with a lower delay and thus it makes no sense to return to it. For the same reason it makes no sense to place a relay at the source. Thus, we can remove all copies of the source on layers greater than zero (see set V^u_{R}). In addition, we remove all arcs that start at or target the removed vertices.

The idea followed by the MILP models given in this section is to model feasible paths between every source $u \in \mathcal{K}_S$ and its targets $v \in K(u)$ using layered graphs G^u_{L}. To end up with a feasible solution, these layered graphs are finally joined using variables $x_e, \forall e \in E$, and $y_i, \forall i \in V$, to obtain a solution on the original graph $G = (V, E)$.

3.2.1 Multi-Commodity Flow Formulation on Multiple Layered Graphs

We start with a completely disaggregated variant. To this end, we utilize one set of variables for each commodity $(u, v) \in \mathcal{K}$ in graph G^u_{L}. We define a multi-commodity flow for each variable set.

We will refer to this model as multi-commodity flow formulation on multiple layered graphs (L_{MCFM}). The formulation utilizes three sets of variables. First we have a set of flow variables f^{uv}_a for the arcs $a \in A^u_{\mathrm{L}}$ of each graph $G^u_{\mathrm{L}}, \forall u \in \mathcal{K}_S$, and per target $v \in K(u)$. Furthermore, we need variables to link the individual variable sets to. To this end, we use binary variables $x_e, \forall e \in E^*$, that are set to one if an augmenting edge e is used in any variable set and to zero otherwise. Moreover, we require binary variables $y_i, \forall i \in V$, that are set to one if vertex i is a relay and to zero otherwise.

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^+(u_0)} f_a^{uv} = 1 \qquad \forall (u,v) \in \mathcal{K}, u_0 \in V_{\mathrm{L}}^u \tag{3.9}$$

$$\sum_{a \in \delta^-(i_l)} f_a^u - \sum_{a \in \delta^+(i_l)} f_a^{uv} = 0 \qquad \forall (u,v) \in \mathcal{K}, \forall i_l \in V_{\mathrm{L}}^u, i \neq u, i \neq v \tag{3.10}$$

$$\sum_{v_l \in V_{\mathrm{L}}^u} \sum_{a \in \delta^-(v_l)} f_a^{uv} = 1 \qquad \forall (u,v) \in \mathcal{K} \tag{3.11}$$

$$\sum_{v_l \in V_{\mathrm{L}}^u} \sum_{a \in \delta^+(v_l)} f_a^{uv} = 0 \qquad \forall (u,v) \in \mathcal{K} \tag{3.12}$$

$$\sum_{(i_l, i_0) \in A_{\mathrm{L}}^u} f_{(i_l,i_0)}^{uv} \leq y_i \qquad \forall (u,v) \in \mathcal{K}, \forall i \in V, i \neq u \tag{3.13}$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u} f_{(i_l,j_m)}^{uv} \leq x_e \qquad \forall (u,v) \in \mathcal{K}, \forall e \in E^*, \forall (i,j) \in A(e) \tag{3.14}$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u} f_{(i_l,j_m)}^{uv} \leq 1 \qquad \forall (u,v) \in \mathcal{K}, \forall e \in E^0, \forall (i,j) \in A(e) \tag{3.15}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{3.16}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \tag{3.17}$$

$$0 \leq f_a^{uv} \leq 1 \qquad \forall (u,v) \in \mathcal{K}, \forall a \in A_{\mathrm{L}}^u \tag{3.18}$$

Inequalities (3.9) state that for each commodity the source sends out one unit of flow on layer zero. Constraints (3.10) ensure flow conservation per layer for vertices that are neither source nor target of the commodity. Equations (3.11) guarantee that the target node consumes one unit of flow on one of its layers. We split the usual flow inequality because we know that we do not have to continue after we reached the target. This is enforced by Constraints (3.12). Constraints (3.13) link the relay arcs to the relay variables. From Theorem 2.1.1 it follows that at most one of the relay arcs per vertex might be selected because every relay is visited at most once. Constraints (3.14) link the layered arcs to the corresponding augmenting edges. If an augmenting edge is used on any layer in any variable set then it has to be part of the solution. Observe that we have to use one constraint per direction of an edge because loops are possible. Note that although A_{L}^u contains augmenting and free arcs we only need linking constraints for the augmenting edges since the free edges have no influence on the objective. For the free edges we use Constraints (3.15) to ensure that only one flow arc per direction of an edge is used. These Constraints are based on Theorem 2.1.3 using the fact that an edge is traversed at most once per direction.

In addition to the required constraints we add the following optional constraints:

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u : l > 0} f_{(i_l,j_m)}^{uv} \leq 2 \cdot (1 - y_i) \qquad \forall (u,v) \in \mathcal{K}, \forall i \in V, i \neq u \tag{3.19}$$

Constraints (3.19) are similar to Constraints (3.7) of the previous model. These constraints ensure that whenever a node $i \in V$ is marked as relay for some commodity, all commodities use it as relay. Thus, we may only use the arcs targeting layer zero. Note that the right-hand side

is multiplied by two. This is a consequence of Theorem 2.1.1 and Corollary 2.1.1: A non-relay vertex might be visited at most twice with different delays and thus there can be out-flow on two different layers.

When using this model we have to take into account that it requires a huge amount of variables when $\mathcal{K}$ contains many commodity pairs.

3.2.2 Single-Commodity Flow Formulation on Multiple Layered Graphs

To reduce the number of variables needed we may aggregate this multi-commodity flow approach into a single-commodity flow formulation (per source). To this end, we only use one set of flow variables per graph $G^u_{\mathrm{L}} = (V^u_{\mathrm{L}}, A^u_{\mathrm{L}})$.

Note that this greatly reduces the number of variables when dealing with a large amount of commodities. We might have at most $\frac{n^2-n}{2}$ $\mathcal{K}$-pairs but only a maximum of $(n-1)$ sources. However, if we have only few pairs or few targets per source, we do not gain much.

We call this modified version single-commodity flow formulation on multiple layered graphs (L_{SCFM}). This model also utilizes three sets of variables. First we have a set of flow variables f^u_a for the arcs $a \in A^u_{\mathrm{L}}$ of each graph $G^u_{\mathrm{L}}, \forall u \in \mathcal{K}_S$. In addition, we again use binary variables $x_e, \forall e \in E^*$, to link the augmenting edges and binary variables $y_i, \forall i \in V$, for the relays.

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^+(u_0)} f^u_a = |K(u)| \qquad \forall u \in \mathcal{K}_S, u_0 \in V^u_{\mathrm{L}} \tag{3.20}$$

$$\sum_{a \in \delta^-(i_l)} f^u_a - \sum_{a \in \delta^+(i_l)} f^u_a = 0 \qquad \forall u \in \mathcal{K}_S, \forall i_l \in V^u_{\mathrm{L}}, i \neq u, i \notin K(u) \tag{3.21}$$

$$0 \leq \sum_{a \in \delta^-(i_l)} f^u_a - \sum_{a \in \delta^+(i_l)} f^u_a \leq 1 \qquad \forall u \in \mathcal{K}_S, \forall i_l \in V^u_{\mathrm{L}}, i \in K(u) \tag{3.22}$$

$$\sum_{i_l \in V^u_{\mathrm{L}}} \left(\sum_{a \in \delta^-(i_l)} f^u_a - \sum_{a \in \delta^+(i_l)} f^u_a \right) = 1 \qquad \forall u \in \mathcal{K}_S, i \in K(u) \tag{3.23}$$

$$\sum_{(i_l, i_0) \in A^u_{\mathrm{L}}} f^u_{(i_l, i_0)} \leq |K(u)| \cdot y_i \qquad \forall u \in \mathcal{K}_S, \forall i \in V, i \neq u \tag{3.24}$$

$$\sum_{(i_l, j_m) \in A^u_{\mathrm{L}}} f^u_{(i_l, j_m)} \leq |K(u)| \cdot x_e \qquad \forall u \in \mathcal{K}_S, \forall e \in E^*, \forall (i,j) \in A(e) \tag{3.25}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{3.26}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \tag{3.27}$$

$$0 \leq f^u_a \leq |K(u)| \qquad \forall u \in \mathcal{K}_S, \forall a \in A^u_{\mathrm{L}} \tag{3.28}$$

The model is analogous to the previous one. The difference is that the source sends out flow according to the number of its targets. Consequently, the flow variables might take values greater than one. To accommodate for this we have to use Big-M values in Constraints (3.24) and (3.25). Furthermore, it is now possible that we do not stop at some targets but continue to reach further targets. As a result targets might even be relays. Therefore, we now also have to

enforce flow conservation for the targets. Since we do not know the layer at which we reach the target we have to state these constraints per layer (3.22) and across all layers (3.23). Note that actually both are required. Omitting (3.22) allows to keep the flow balance by having in-flow on some layer and out-flow on a different layer and without (3.23) some targets might not be reached.

We also use an adaption of the optional constraints of the previous model:

$$\sum_{(i_l,j_m)\in A_{\mathrm{L}}^u : l>0} f^u_{(i_l,j_m)} \leq (1-y_i)\cdot|K(u)|\cdot 2 \qquad \forall u\in\mathcal{K}_S, \forall i\in V, i\neq u \tag{3.29}$$

3.2.3 Cut Model on Multiple Layered Graphs

This model also uses one set of variables per graph $G^u_{\mathrm{L}} = (V^u_{\mathrm{L}}, A^u_{\mathrm{L}})$. However, we do not send flow to ensure connectivity. Instead we use connectivity inequalities to ensure that all targets can communicate.

We will refer to this formulation as cut formulation on multiple layered graphs (L_{CUTM}). The model uses binary arc variables X^u_a for the arcs $a \in A^u_{\mathrm{L}}$ per source $u \in \mathcal{K}_S$. In addition, we again use binary variables $x_e, \forall e \in E^*$, for the linking of the augmenting edges and binary variables $y_i, \forall i \in V$, for the relays.

$$\min \sum_{i\in V} c_i y_i + \sum_{e\in E^*} w_e x_e$$

$$\sum_{a\in\delta^-(W)} X^u_a \geq 1 \qquad \begin{array}{r}\forall u\in\mathcal{K}_S, \forall v\in K(u),\\ \{v_l | v_l\in V^u_{\mathrm{L}}\}\subseteq W\subset V^u_{\mathrm{L}},\\ u_0\notin W\end{array} \tag{3.30}$$

$$\sum_{i_l\in V^u_{\mathrm{L}}:l>0}\ \sum_{a\in\delta^-(i_l)} X^u_a \leq 2-y_i \qquad \forall u\in\mathcal{K}_S, \forall i\in V, i\neq u \tag{3.31}$$

$$\sum_{(i_l,i_0)\in A^u_{\mathrm{L}}} X^u_{(i_l,i_0)} \leq y_i \qquad \forall u\in\mathcal{K}_S, \forall i\in V \tag{3.32}$$

$$\sum_{(i_l,j_m)\in A^u_{\mathrm{L}}} X^u_{(i_l,j_m)} \leq x_e \qquad \forall u\in\mathcal{K}_S, \forall e\in E^*, \forall (i,j)\in A(e) \tag{3.33}$$

$$X^u_a \in \{0,1\} \qquad \forall u\in\mathcal{K}_S, \forall a\in A^u_{\mathrm{L}} \tag{3.34}$$

$$y_i \in \{0,1\} \qquad \forall i\in V \tag{3.35}$$

$$x_e \in \{0,1\} \qquad \forall e\in E^* \tag{3.36}$$

The first set of constraints are the cut-inequalities. Each subset of the vertices containing all copies of some target $v \in K(u)$ on all layers requires an in-coming arc. Since $u_0 \notin W$ we know that each target has to be connected to its corresponding source in graph G^u_{L}. Note that due to the dependence on subsets the number of these constraints is in general exponential. Corresponding separation methods will be discussed in Chapter 6. Constraints (3.31) state that every relay is reached at most once and all non-relays are reached at most twice (see Theorem 2.1.2). Constraints (3.13) link the relay arcs to the relay variables. Due to Theorem 2.1.1 we know that at most one of the relay arcs per vertex might be selected. Constraints (3.33) link the layered arcs to the augmenting edges. If an augmenting edge is used on any layer in any graph

then it has to be part of the solution. Observe that we have to use one constraint per direction of an edge because loops are possible. Note that although A^u_{L} contains augmenting and free arcs we only need linking constraints for the augmenting edges since the free edges have no influence on the objective.

In addition to the required constraints we again add some optional constraints:

$$\sum_{(i_l,j_m)\in A^u_{\mathrm{L}}:l>0\wedge m>0} X^u_{(i_l,j_m)} \leq (|K(u)|+1)\cdot(1-y_i) \qquad \forall u\in\mathcal{K}_S, \forall i\in V, i\neq u \quad (3.37)$$

$$\sum_{(i_l,j_m)\in A^u_{\mathrm{L}}:l>0} X^u_{(i_l,j_m)} \leq 2\cdot(1-y_i) \qquad \begin{array}{r}\forall u\in\mathcal{K}_S, \forall i\in V, \forall\{i,j\}\in E,\\ i\neq u, j\neq u\end{array} \quad (3.38)$$

$$\sum_{a\in\delta^+(i_l)} X^u_a \leq \min(|K(u)|, |\delta^+(i_l)|)\cdot \sum_{a\in\delta^-(i_l)} X^u_a \qquad \forall u\in\mathcal{K}_S, \forall i_l\in V^u_{\mathrm{L}}, i\neq u \quad (3.39)$$

Constraints (3.37) are similar to Constraints (3.29) of the previous flow model. These constraints ensure that whenever a vertex $i\in V$ is marked as relay in one variable set, all variable sets use it as relay. Thus, we may only use the arcs targeting layer zero. Note that here we have a stronger restriction for the right-hand side than for L_{SCFM}. We use Theorem 2.1.2 and Corollary 2.1.1: A non-relay vertex might be visited at most twice with different delays and thus there can be out-going arcs on two different layers. Since we are not dealing with flows here we know that we require at most one arc to visit the relay and at most $|K(u)|$ more arcs to reach all targets, i.e., $|K(u)|+1$. Constraints (3.38) are a variant of these constraints that are formulated per edge instead of over all incident edges. However, this only pays off if $|K(u)|$ is large. Inequalities (3.39) ensure that a vertex might only have out-going arcs if it has an in-coming arc and the number of out-going arcs is bounded by the minimum of targets to reach and its out-degree.

To reduce the number of dynamically generated cuts we add for every source $u\in\mathcal{K}$ constraints corresponding to sets $W=\{v_l|v_l\in V^u_{\mathrm{L}}, l>0\}$ for all targets $v\in K(u)$ in advance using the fact that every target has to be reached on some layer greater than zero:

$$\sum_{v_l\in V^u_{\mathrm{L}}:l>0}\ \sum_{a\in\delta^-(v_l)} X^u_a \geq 1 \qquad \forall u\in\mathcal{K}_S, \forall v\in K(u)$$

CHAPTER 4

Models on Communication Graphs

A communication graph is created by adding edges between all vertices that can be connected with a path having delay smaller than or equal to d_{max} in the original graph $G = (V, E)$. We will refer to the edges added this way that are not part of the original graph as *communication edges*. This transformation has been used in [6] and [5] to solve the RLP and GRLP, respectively. Both problems deal with edges of cost zero only. If an arbitrary pair of vertices can be connected using only free edges then we know that this is the only relevant connection between them. Note that there might be different feasible paths using only free edges but it is sufficient if at least one such connection exists. We do not care about the specific path because the costs will stay the same regardless of the used free edges.

However, if there are only feasible connections containing at least one edge with positive costs, it is not sufficient to select one of these connections. Using the cheapest connection is in general not the optimal strategy. An edge might be used in multiple connections. This reuse can make a connection that is dominated for a certain vertex pair still profitable, concerning the overall costs. In Figure 4.1 we depict free edges in solid lines and augmenting edges in dotted lines. The numbers next to the edges denote their delays and the numbers in parentheses their costs. In this example the cheapest connection between nodes 0 and 2 is the augmenting edge $(0, 2)$ and the cheapest connection between nodes 0 and 3 is edge $(0, 3)$, both with a cost of two yielding an overall cost of 4. However, if we use edge $(0, 1)$ with a cost of 3 instead (which is not optimal for either of the individual connections) we can decrease the overall costs to 3. As a result, we have to consider all such connections. Unfortunately, the number of possible connections is in general exponential.

4.1 Definitions

Two nodes can communicate in a given instance without the use of relays if there exists a path between them whose total delay does not exceed $d_{\max}$. Let $\hat{P}_{\{i,j\}}$ be the set of all paths between i and j in $G = (V, E)$. Note that the original graph is undirected. Thus, there exists for each

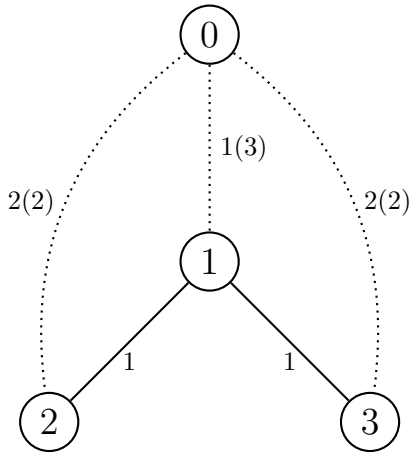


Figure 4.1: Dominated Connection

path a symmetric path in the opposite direction. For our purposes it is sufficient to consider one of them. W.l.o.g. we only consider the paths from i to j for $i < j$. We denote the set of paths between vertices i and j having a delay smaller than or equal to d_{max} by $P_{\{i,j\}}$, i.e., $P_{\{i,j\}} = \{p | p \in \hat{P}_{\{i,j\}}, \Delta(p) \leq d_{max}\}$. If there are different paths containing the same set of augmenting edges having the same delay it is sufficient to select one of them arbitrarily since the free edges have no effect on the costs of the path.

We use the following functions to determine the minimum delay between two vertices w.r.t. a certain edge set:

$$md^0(i,j) = \min_{p \in \hat{P}_{\{i,j\}}} \Delta(p) \text{ w.r.t. } G^0 = (V, E^0)$$
$$md(i,j) = \min_{p \in \hat{P}_{\{i,j\}}} \Delta(p) \text{ w.r.t. } G = (V, E)$$

Accordingly, we define sets of node pairs that are able to communicate w.r.t. these edge sets:

$C^0 = \{\{i,j\} \mid md^0(i,j) \leq d_{max}, i \in V, j \in V, i < j\}$	connected in E^0
$C = \{\{i,j\} \mid md(i,j) \leq d_{max}, i \in V, j \in V, i < j\}$	connected in E
$C^* = C \setminus C^0$	connected in E but not in E^0

In general we transform the original graph $G = (V, E)$ to a communication graph $G_{\mathrm{C}} = (V_{\mathrm{C}}, E_{\mathrm{C}})$ as follows:

$$V_{\mathrm{C}} = V$$
$$E_{\mathrm{C}} = C$$

Figure 4.2 shows the stepwise generation of a communication graph. Figure 4.2 (a) shows the original graph. Free edges are depicted in solid lines and augmenting edges in dotted lines. The numbers next to the edges denote their delays and in the following we assume $d_{max} = 6$.

To create G_{C} we start by adding all connections that are not yet present but that can be established by using only free edges and no relays. The edges we have to add correspond to the set $C^0 \setminus E^0$. In the example we have to add an edge from 0 to 2 (dashed line) according to path $(0, 1, 2)$ with $md^0(0, 2) = 6$ which does not exceed d_{max}. Note that this edge overlaps with an augmenting edge. Although the augmenting edge has smaller delay we consider the free edge as only relevant option for the connection of 0 and 2 because its cost will never be higher than that of the augmenting edge. Nevertheless, we still might use the augmenting edge as part of some other connection. Due to its smaller delay it might enable a connection within d_{max} that cannot be established when using the free edge.

In the next step we add connections that are only possible by installing augmenting edges. The edges we have to add correspond to the set C^* and are depicted in Figure 4.2 (c) in dot-dashed lines. Note that these connections overlap with the augmenting edges. The reason for this is the fact that an augmenting edge is only one possibility of realizing the connection. The edge between 1 and 3 for example can be established by using the augmenting edge $\{1, 3\}$ but also by the paths $(1, 0, 3)$ and $(1, 2, 3)$. It is of central importance that although we do not know the optimal choice to establish the connection, we still know which connections are possible at all. To identify these connections we may use well known all-pairs shortest path algorithms (using the delays as costs) such as the Floyd-Warshall algorithm (see [11]) or Johnson's algorithm (see [24]).

Figure 4.2 (d) shows the final communication graph. We display the connections in C^0 in solid lines and the connections in C^* in dash-dotted lines.

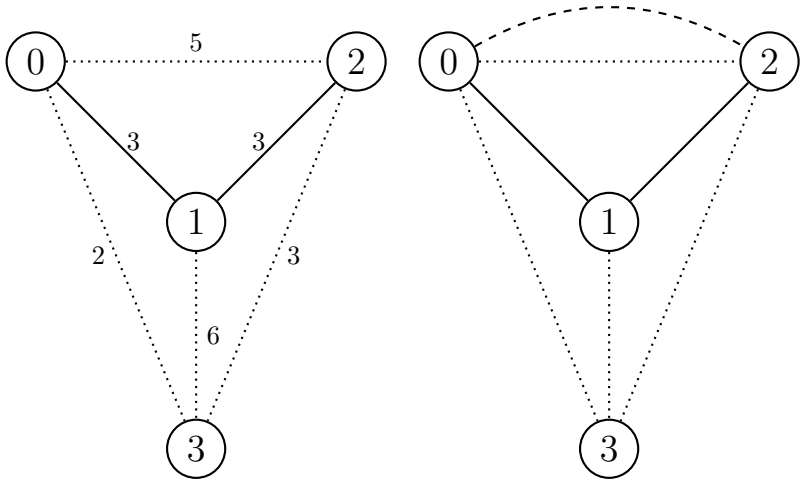
The graph generated this way contains all feasible connections that can be established without the use of relays. Hence, it is admissible to enforce all intermediate vertices on a path in G_{C} to be relays.

4.2 Model on a Single Communication Graph

Similar as for the layered approach we start with a model that is based on a single graph. To identify relays we split each vertex i into two copies i_1, i_2. We then create arcs s.t. for each vertex i all in-coming arcs target i_1 and all out-going arcs start at i_2. Moreover, we add arcs (i_1, i_2) to connect the vertex copies. We obtain the graph $G'_{\mathrm{C}} = (V'_{\mathrm{C}}, A'_{\mathrm{C}})$:

$$
\begin{aligned}
V'_{\mathrm{C}} &= \{i_1, i_2 | i \in V_{\mathrm{C}}\} \\
A^r_{\mathrm{C}} &= \{(i_1, i_2) | i \in V_{\mathrm{C}}\} \\
A'_{\mathrm{C}} &= \{(i_2, j_1), (j_2, i_1) | \{i, j\} \in E_{\mathrm{C}}\} \cup A^r_{\mathrm{C}}
\end{aligned}
$$

The arcs $(i_1, i_2) \in A^r_{\mathrm{C}}$ are used to identify relays. If there is an in-coming arc and an out-going arc but the connection of the vertex copies is not selected then the vertex is target and source but not part of another connection and thus, needs not to be relay. If, however, the



(a) Original Graph

(b) Adding all Connections in C^0 (dashed lines)

(c) Adding all Connections in C^* (dash-dotted lines)

(d) Final Communication Graph

Figure 4.2: Generation of the Communication Graph

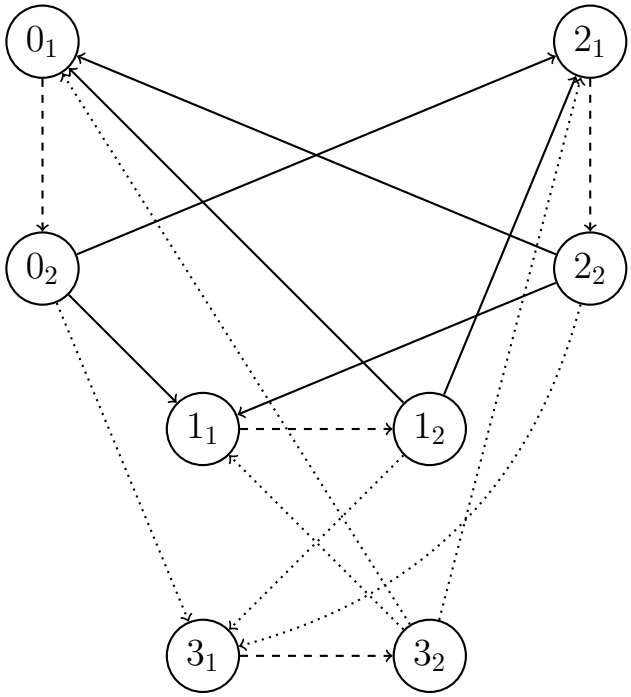


Figure 4.3: Communication Graph $G'_{\mathrm{C}} = (V'_{\mathrm{C}}, A'_{\mathrm{C}})$ according to the instance in Figure 4.2 (a)

connection of the copies is selected we need to place a relay as there has to be a connection going through this vertex. Figure 4.3 shows the modified version of the graph in Figure 4.2. We depict free connections in solid lines, connections with augmenting edges in dotted lines and relay arcs in dashed lines.

4.2.1 Cut Model on a Single Communication Graph

We call this model cut formulation on a single communication graph (CG_{CUTS}). We utilize binary variables x_e for the augmenting edges and binary variables X_a for the arcs of the communication graph. Due to the one-to-one correspondence between the arcs in A^r_{C} and the relays we can use the arc variables directly to identify the relays. Moreover, we use continuous variables λ^p_b for the paths p that connect the node pairs $b \in C^*$ s.t. $\Delta(p) \leq d_{max}$.

$$\min \sum_{i \in V} c_i X_{(i_1,i_2)} + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^-(W)} X_a \geq 1 \qquad \forall u \in \mathcal{K}_S, \forall W \subset V'_{\mathrm{C}}, \exists v \in K(u) : \{v_1\} \subseteq W, u_2 \notin W \tag{4.1}$$

$$|\delta(i)| \cdot X_{(i_1,i_2)} \geq \sum_{a \in \delta^+(i_2)} X_a \qquad \forall i \notin \mathcal{K}_S \tag{4.2}$$

$$\sum_{a \in A(b)} X_a \leq 2 \cdot \sum_{p \in P_b} \lambda_b^p \qquad \forall b \in C^* \quad (\mu_b) \tag{4.3}$$

$$\sum_{p \in P_b : e \in p} \lambda_b^p \leq x_e \qquad \forall e \in E^*, \forall b \in C^* \quad (\alpha_b^e) \tag{4.4}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \tag{4.5}$$

$$X_a \in \{0,1\} \qquad \forall a \in A'_{\mathrm{C}} \tag{4.6}$$

$$\lambda_b^p \geq 0 \qquad \forall b \in C^*, p \in P_b \tag{4.7}$$

The first set of constraints are the cut inequalities. Each subset of the vertices containing a target copy v_1 requires an in-coming arc. Since $u_2 \notin W$ this means that every target has to be connected to its source. Thus, we enforce a connection from source copy u_2 to each target copy v_1. Note that due to the dependence on subsets the number of these constraints is in general exponential. Corresponding separation methods will be discussed in Chapter 6. Constraints (4.2) ensure that non-source vertices might only have out-going arcs if they are relays. The next set of constraints ensures that if a connection in $b \in C^*$ is used then at least one path realization from P_b has to be selected. Due to the presence of multiple sources both arc directions of an edge might be part of the solution. Thus, we need to multiply the right-hand side by two. The final set of constraints ensures that all augmenting edges of the selected realizations are part of the solution.

In addition to the required constraints we add the following optional ones:

$$X_{(i_1,i_2)} \leq \sum_{a \in \delta^+(i_2)} X_a \qquad \forall i \notin \mathcal{K}_S \tag{4.8}$$

$$X_{(i_1,i_2)} \leq \sum_{a \in \delta^-(i_1)} X_a \qquad \forall i \in V \tag{4.9}$$

Constraints (4.8) state that all relays have an out-going arc and Constraints (4.9) state that all relays require an in-coming arc. Note that sources always have at least one out-going arc.

To reduce the number of dynamically generated cuts we add the following constraints in advance:

$$\sum_{a \in \delta^-(v_1)} X_a \geq 1 \qquad \forall v \in \bigcup_{u \in \mathcal{K}_S} K(u)$$

$$\sum_{a \in \delta^+(u_2)} X_a \geq 1 \qquad \forall u \in \mathcal{K}_S$$

The first set of these constraints uses the fact that every target requires an in-coming arc, i.e. we consider sets $W = \{v_1\}, \forall u \in \mathcal{K}_S, \forall v \in K(u)$. The second set of constraints depends

on the fact that every source has at least one out-going arc. Thus, this corresponds to sets $W = V'_{\mathrm{C}} \setminus \{u_2\}, \forall u \in \mathcal{K}_S$.

Pricing Subproblem

The presented MILP model contains an exponential number of variables. To solve the LP relaxation, we will use column generation. The underlying pricing subproblem is defined as follows. To state the dual constraints for the path variables we use dual variables μ_b for Constraints (4.3) and dual variables α_b^e for Constraints (4.4):

$$\begin{aligned} 2 \cdot \mu_b - \sum_{e \in E^* \cap p} \alpha_b^e \leq 0 \qquad & \forall b \in C^*, \forall p \in P_b \\ \mu_b \geq 0 \qquad & \forall b \in C^* \\ \alpha_b^e \geq 0 \qquad & \forall e \in E^*, \forall b \in C^* \end{aligned}$$

Thus, we obtain the following pricing subproblem for each $b \in C^*$:

$$\underset{p \in P(b)}{\arg\min} \left\{ 0 - \left(2 \cdot \mu_b - \sum_{e \in E^* \cap p} \alpha_b^e \right) \right\}$$

This can be solved by the following subproblem (since μ_b is a constant for a fixed b):

$$\forall b \in C^* \; R_b = \underset{p \in P(b)}{\arg\min} \sum_{e \in E^* \cap p} \alpha_b^e$$

The subproblems R_b are Weight Constrained Shortest Path Problems (WCSPPs). The models in the following section require this subproblem as well. We are going to discuss this problem in detail in Section 4.4.

Note that the provided model also uses an exponential number of constraints. Thus, we require Branch-Price-and-Cut to solve it. Details will be given in Chapter 6. Fortunately, we can separate these parts s.t. column generation can be done independently of cut generation, i.e., the added cuts do not influence the structure of the pricing subproblem.

4.3 Models on Multiple Communication Graphs

As for the layered graph approaches we present models using one graph per source vertex. We create graphs $G_{\mathrm{C}}^u = (V_{\mathrm{C}}^u, A_{\mathrm{C}}^u)$ per source $u \in \mathcal{K}_S$. Again we omit the arcs targeting the source vertex:

$$\begin{aligned} V_{\mathrm{C}}^u =& V_{\mathrm{C}} \\ A_{\mathrm{C}}^u =& \{(i,j),(j,i) | \{i,j\} \in E_{\mathrm{C}}\} \setminus \{(i,u) | \{i,u\} \in E_{\mathrm{C}}\} \end{aligned}$$

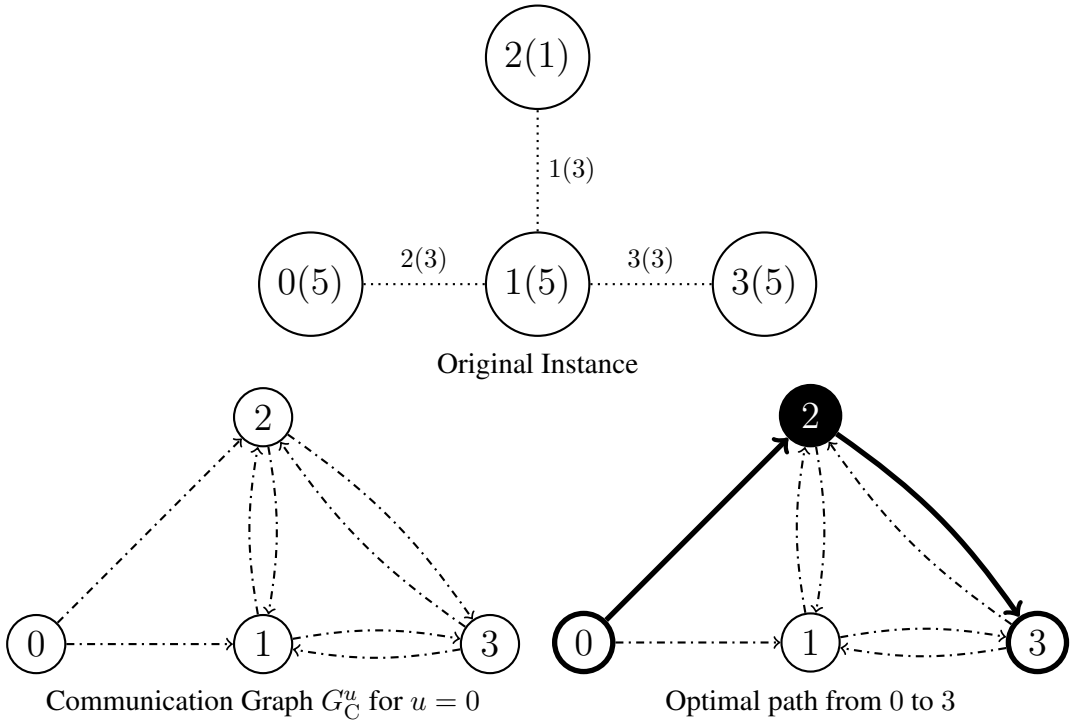


Figure 4.4: Cyclic Solution

Figure 4.4 shows the optimal solution for $d_{max} = 4$ and $\mathcal{K} = \{(0, 3)\}$ in G_C^u w.r.t. the previously introduced example. Note that although the path in the communication graph is acyclic this still corresponds to the cyclic solution shown in Figure 2.2.

Due to the specific structure of the communication graph and the disaggregation per source $u \in \mathcal{K}_S$ we obtain the following result:

Corollary 4.3.1. *In an optimal solution on a communication graph there exists for every $u \in \mathcal{K}_s$ an arborescence rooted at u reaching all targets $v \in K(u)$.*

Proof. From Theorem 2.1.2 it follows that an optimal solution contains for every $u \in \mathcal{K}_S$ a digraph rooted at u reaching all targets $v \in K(u)$ and visiting each relay at most once. In a communication graph all intermediate nodes are relays and we never need to return to the source. Hence all vertices have an in-degree of at most one. Thus, the considered digraph is an arborescence rooted at u. □

We conclude that using one graph per source makes it a lot easier to identify relays. Since the optimal solution per source will be a arborescence all vertices different from the source with out-going arcs have to be relays (see Figure 4.4).

4.3.1 Multi-Commodity Flow Formulation on Multiple Communication Graphs

As for the layered approaches we start with the completely disaggregated variant using one set of variables per pair in $\mathcal{K}$. Then, we define a multi-commodity flow on each of these variable sets. Due to the fact that we only deal with a single commodity we obtain a stronger version of Corollary 4.3.1:

Corollary 4.3.2. *In an optimal solution on a communication graph there exists for every $(u, v) \in \mathcal{K}$ a feasible path visiting each vertex at most once, i.e., a simple path.*

Proof. Theorem 2.1.1 implies that an optimal solution contains for every pair in $\mathcal{K}$ a connection visiting each relay at most once. In a communication graph all intermediate nodes are relays, we never need to return to the source and we do not continue after the target has been reached. Hence, it follows that in a communication graph there exists for every $(u, v) \in \mathcal{K}$ a simple feasible path from u to v. □

We will refer to the new formulation as multi-commodity flow formulation on multiple communication graphs (CG_{MCFM}). We use flow variables f_a^{uv} for all arcs a of the graphs $G_{\mathrm{C}}^u, \forall u \in \mathcal{K}_S$, and for each target $v \in K(u)$. We use variables λ_b^p that correspond to the paths p that have been identified as possible realizations for the connections $b \in C^*$. The variables y_i are set to one if vertex i is used as relay and to zero otherwise. Finally we use variables x_e to link the augmenting edges to the path variables. The MILP model reads as follows

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^-(i)} f_a^{uv} - \sum_{a \in \delta^+(i)} f_a^{uv} = \begin{cases} -1 & i = u \\ 0 & i \neq u, i \neq v \end{cases} \qquad \forall (u,v) \in \mathcal{K}, \forall i \in V_{\mathrm{C}}^u \tag{4.10}$$

$$\sum_{a \in \delta^-(v)} f_a^{uv} = 1 \qquad \forall (u,v) \in \mathcal{K} \tag{4.11}$$

$$\sum_{a \in \delta^+(v)} f_a^{uv} = 0 \qquad \forall (u,v) \in \mathcal{K} \tag{4.12}$$

$$\sum_{a \in \delta^+(i)} f_a^{uv} \leq y_i \qquad \forall (u,v) \in \mathcal{K}, \forall i \in V, i \neq u, i \neq v \tag{4.13}$$

$$\sum_{a \in A(b)} f_a^{uv} \leq \sum_{p \in P_b} \lambda_b^p \qquad \forall (u,v) \in \mathcal{K}, \forall b \in C^* \quad (\mu_b^{uv}) \tag{4.14}$$

$$\sum_{p \in P_b : e \in p} \lambda_b^p \leq x_e \qquad \forall e \in E^*, \forall b \in C^* \quad (\alpha_b^e) \tag{4.15}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{4.16}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \tag{4.17}$$

$$\lambda_b^p \geq 0 \qquad \forall b \in C^*, p \in P_b \tag{4.18}$$

$$0 \leq f_a^{uv} \leq 1 \qquad \forall (u,v) \in \mathcal{K}, \forall a \in A_{\mathrm{C}}^u \tag{4.19}$$

The first set of constraints ensures flow conservation. The source of each variable set sends out one unit of flow. For vertices that are neither source nor target, flow conservation has to hold.

Since we are only dealing with a single target per variable set, this target consumes the single unit of flow (4.11) and has no out-going flow (4.12). Inequalities (4.13) enforce that vertices with out-going flow that are not the source of their corresponding commodity become relays. Constraints (4.14) ensure that flow among connections $b \in C^*$ is only possible if at least one of the available realizations has been selected. Due to Corollary 4.3.2, the solution for each variable set will be a simple path. Hence, we know that only one arc per edge will be selected in each variable set. The last set of inequalities guarantees that for all selected realizations the corresponding augmenting edges will be part of the solution.

Pricing Subproblem

The presented MILP model contains an exponential number of λ variables and for solving its LP relaxation, we will use column generation. The underlying pricing subproblem is defined as follows. To state the dual constraints for the path variables we use dual variables μ_b^{uv} for Constraints (4.14) and dual variables α_b^e for Constraints (4.15):

$$\begin{aligned} \sum_{(u,v)\in\mathcal{K}} \mu_b^{uv} - \sum_{e\in E^*\cap p} \alpha_b^e \leq 0 \qquad & \forall b \in C^*, \forall p \in P_b \\ \mu_b^{uv} \geq 0 \qquad & \forall (u,v) \in \mathcal{K}, \forall b \in C^* \\ \alpha_b^e \geq 0 \qquad & \forall e \in E^*, \forall b \in C^* \end{aligned}$$

Thus, for each $b \in C^*$ the pricing subproblem decomposes into:

$$\underset{p\in P(b)}{\arg\min} \left\{ 0 - \left(\sum_{(u,v)\in\mathcal{K}} \mu_b^{uv} - \sum_{e\in E^*\cap p} \alpha_b^e \right) \right\}$$

This can be solved by the following subproblem:

$$\forall b \in C^* \; R_b = \underset{p\in P(b)}{\arg\min} \sum_{e\in E^*\cap p} \alpha_b^e$$

The problems defined by R_b are WCSPPs. Details on this problem will be given in Section 4.4.

Note that when dealing with a large amount of commodity pairs this model requires a large number of variables.

4.3.2 Single-Commodity Flow Formulation on Multiple Communication Graphs

To reduce the number of variables we again aggregate the pairs in $\mathcal{K}$ as for the layered approach. We then use one set of variables per graph $G_{\mathrm{C}}^u = (V_{\mathrm{C}}^u, A_{\mathrm{C}}^u)$ for each source $u \in \mathcal{K}_S$.

We will refer to this formulation as single-commodity flow formulation on multiple communication graphs (CG_{SCFM}). The formulation we obtain uses the same variables as the previous model. The only difference is that we only need flow variables f_a^u per source $u \in \mathcal{K}_S$.

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^-(i)} f_a^u - \sum_{a \in \delta^+(i)} f_a^u = \begin{cases} -|K(u)| & i = u \\ 1 & i \in K(u) \\ 0 & i \neq u, i \notin K(u) \end{cases} \qquad \forall u \in \mathcal{K}_S, \forall i \in V_{\mathrm{C}}^u \tag{4.20}$$

$$\sum_{a \in \delta^+(i)} f_a^u \leq |K(u)| \cdot y_i \qquad \forall u \in \mathcal{K}_S, \forall i \in V, i \neq u \tag{4.21}$$

$$\sum_{a \in A(b)} f_a^u \leq |K(u)| \cdot \sum_{p \in P_b} \lambda_b^p \qquad \forall u \in \mathcal{K}_S, \forall b \in C^* \quad (\mu_b^u) \tag{4.22}$$

$$\sum_{p \in P_b : e \in p} \lambda_b^p \leq x_e \qquad \forall e \in E^*, \forall b \in C^* \quad (\alpha_b^e) \tag{4.23}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{4.24}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \tag{4.25}$$

$$\lambda_b^p \geq 0 \qquad \forall b \in C^*, p \in P_b \tag{4.26}$$

$$0 \leq f_a^u \leq |K(u)| \qquad \forall u \in \mathcal{K}_S, \forall a \in A_{\mathrm{C}}^u \tag{4.27}$$

The constraints are quite similar to the previous formulation. The difference is that the source sends out flow w.r.t. the number of its targets $|K(u)|$. Thus, we have to use Big-Ms in inequalities (4.21) and (4.22) to accommodate for this. Furthermore, the targets might have out-flow now. Hence, we use the usual flow-balance constraints (4.20).

Pricing Subproblem

The presented MILP model contains an exponential number of variables. To solve the LP relaxation, we will use column generation. The underlying pricing subproblem is defined as follows. To state the dual constraints for the path variables we use dual variables μ_b^u for Constraints (4.22) and dual variables α_b^e for Constraints (4.23):

$$\sum_{u \in \mathcal{K}_S} (|K(u)| \cdot \mu_b^u) - \sum_{e \in E^* \cap p} \alpha_b^e \leq 0 \qquad \forall b \in C^*, \forall p \in P_b$$

$$\mu_b^u \geq 0 \qquad \forall u \in \mathcal{K}_S, \forall b \in C^*$$

$$\alpha_b^e \geq 0 \qquad \forall e \in E^*, \forall b \in C^*$$

Thus, we obtain the following pricing subproblem for each $b \in C^*$:

$$\operatorname*{arg\,min}_{p \in P(b)} \left\{ 0 - \left(\sum_{u \in \mathcal{K}_S} (|K(u)| \cdot \mu_b^u) - \sum_{e \in E^* \cap p} \alpha_b^e \right) \right\},$$

which can be solved by the following subproblem:

$$\forall b \in C^* \; R_b = \operatorname*{arg\,min}_{p \in P(b)} \sum_{e \in E^* \cap p} \alpha_b^e$$

The problems defined by R_b are WCSPPs (see Section 4.4).

4.3.3 Cut Model on Multiple Communication Graphs

For the cut model we also use one set of variables per source vertex. Instead of the flows we utilize cuts to ensure connectivity.

We call this formulation cut formulation on multiple communication graphs (CG_{CUTM}). The model uses variables X_a^u for the arcs a of the graphs G_C^u per source $u \in \mathcal{K}_s$. The remaining variable sets are equivalent to those used in the flow models.

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^-(W)} X_a^u \geq 1 \qquad \forall u \in \mathcal{K}_S, \forall W \subset V_C^u, W \cap K(u) \neq \emptyset, u \notin W \tag{4.28}$$

$$\sum_{a \in \delta^+(i)} X_a^u \leq \min(|K(u)|, |\delta^+(i)| - 1) \cdot y_i \qquad \forall u \in \mathcal{K}_S, \forall i \in V, i \neq u \tag{4.29}$$

$$\sum_{a \in A(b)} X_a^u \leq \sum_{p \in P_b} \lambda_b^p \qquad \forall u \in \mathcal{K}_S, \forall b \in C^* \quad (\mu_b^u) \tag{4.30}$$

$$\sum_{p \in P_b : e \in p} \lambda_b^p \leq x_e \qquad \forall e \in E^*, \forall b \in C^* \quad (\alpha_b^e) \tag{4.31}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{4.32}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \tag{4.33}$$

$$X_a^u \in \{0,1\} \qquad \forall u \in \mathcal{K}_S, \forall a \in A_C^u \tag{4.34}$$

$$\lambda_b^p \geq 0 \qquad \forall b \in C^*, p \in P_b \tag{4.35}$$

The first set of constraints are the cut inequalities. For each graph G_C^u every set W containing one of the targets $v \in K(u)$ has to be connected to the rest of the graph and since $u \notin W$ finally to the source u. Note that due to the dependence on subsets the number of these constraints is in general exponential. Corresponding separation methods will be discussed in Chapter 6. Constraints (4.29) identify the relays. The number of out-going arcs is bounded by the minimum of the amount of targets that have to be reached and the out-degree of the vertex. The out-degree is reduced by one since paths in communication graphs are acyclic and thus the arc targeting the predecessor is never selected. The final two constraints are identical to CG_{SCFM}. They ensure that arcs in C^* are only possible if some realization is selected and that all augmenting edges of the chosen realizations are set to one.

In addition to the required constraints we add the following optional ones:

$$\sum_{a \in \delta^-(v)} X_a^u = 1 \qquad \forall u \in \mathcal{K}_S, \forall v \in K(u) \tag{4.36}$$

$$\sum_{a \in \delta^+(i)} X_a^u \leq \min(|K(u)|, |\delta^+(i)| - 1) \cdot \sum_{a \in \delta^-(i)} X_a^u \qquad \forall u \in \mathcal{K}_S, \forall i \notin K(u), i \neq u \tag{4.37}$$

Constraints (4.36) ensure that all targets have exactly one in-coming arc (see Corollary 4.3.1). To reduce the number of dynamically generated cuts we add Inequalities (4.37). They state that a vertex, different from source and target, might only have out-going arcs if it has an in-coming arc and the number of out-going arcs is bounded by the minimum of the amount targets that have to be reached and the out-degree of the vertex. The out-degree is reduced by one

since paths in communication graphs are acyclic and thus the arc targeting the predecessor is never selected. We do not impose these constraints on the target because targets always require an in-coming arc regardless of their out-degree.

To reduce the number of dynamically generated cuts we add the following constraints in advance using the fact that there is at least one arc leaving the source, i.e., we consider sets $W = V'_C \setminus \{u\}, \forall u \in \mathcal{K}_S$:

$$\sum_{a \in \delta^+(u)} X_a^u \geq 1 \qquad \forall u \in \mathcal{K}_S$$

Pricing Subproblem

The pricing subproblem associated to this MILP model is defined as follows. To state the dual constraints for the path variables we use dual variables μ_b^u for Constraints (4.30) and dual variables α_b^e for Constraints (4.31):

$$\sum_{u \in \mathcal{K}_S} \mu_b^u - \sum_{e \in E^* \cap p} \alpha_b^e \leq 0 \qquad \forall b \in C^*, \forall p \in P_b$$

$$\mu_b^u \geq 0 \qquad \forall u \in \mathcal{K}_S, \forall b \in C^*$$

$$\alpha_b^e \geq 0 \qquad \forall e \in E^*, \forall b \in C^*$$

Thus, we obtain the following pricing subproblem for each $b \in C^*$:

$$\arg\min_{p \in P(b)} \left\{ 0 - \left(\sum_{u \in \mathcal{K}_S} \mu_b^u - \sum_{e \in E^* \cap p} \alpha_b^e \right) \right\},$$

which can be solved by the following subproblem:

$$\forall b \in C^* \; R_b = \arg\min_{p \in P(b)} \sum_{e \in E^* \cap p} \alpha_b^e$$

Again we are dealing with the WCSPP (see below).

Note that this model uses an exponential number of constraints and an exponential number of variables. Thus, we require Branch-Price-and-Cut to solve it. Details will be given in Chapter 6. Fortunately, column generation can be done independently of cut generation, i.e., the added cuts do not influence the structure of the pricing subproblem.

4.4 Solving the Pricing Subproblems

The subproblems $R_b, \forall b \in C^*$, required by the models presented in this chapter are Weight Constrained Shortest Path Problems (WCSPPs). The WCSPP is in general NP-hard but fast pseudo-polynomial exact solution algorithms are available. For the implementation in our models we use the algorithm presented by Gouveia et al. [20]. Algorithm 4.1 shows the corresponding pseudo code using delays as weights. The algorithm is based on dynamic programming

using states (i,h) for $i \in V$ and $h \in \{0, \ldots, d_{max}\}$. The values $f(j,h)$ correspond to the costs of the cheapest path from source s to vertex j with a delay of at most h and can be computed recursively as follows:

$$f(j,h) = \min_{\{j,i\} \in E : d_{\{j,i\}} \leq h} \left(f(j, h - w_{\{j,i\}}) + w_{\{j,i\}} \right)$$

For the source we define $f(s,h) = 0, \forall h \in \{0, \ldots, d_{max}\}$. Furthermore, $MinCost$ labels are used to keep track of the minimum cost found so far, to avoid dealing with already dominated connections. The algorithm runs in time $O(|E| \cdot d_{max})$. However, by using the fact that the edge costs are non negative and that $f(j,h) \geq f(j,h')$ for $h' > h$ the algorithm achieves better runtimes in practice. In addition, Gouveia et al. note that the algorithm can be further improved by using a good upper bound on the cost of the shortest path from the given source to the target. The MILP models presented in this chapter rely on pricing subproblems that only have negative reduced costs if the cost of the cheapest path is smaller than the sum of the μ value(s). Hence, we can use this sum as upper bound.

Note that we do not use the communication graph for pricing. Instead we use the original graph. The reason for this is the fact that we require information on the original augmenting edges used in the path to state the respective linking constraints.

Furthermore, note that the pairs $\{i,j\} \in C^*$ are undirected. As already mentioned in the beginning of this chapter there exists for every path a symmetric one in the opposite direction in the original undirected graph. As a consequence, it is sufficient to solve the problem in either of the directions, w.l.o.g. we consider the weight constrained shortest path from i to j for $i < j$.

Input: $G = (V, E,)$, edge costs w_e, edge delays d_e
Input: Source s, target t

$S_0 = \{s\}$;
forall the $h \in \{1, \ldots, d_{max}\}$ **do**
  $S_h = \emptyset$;
end
$MinCost(s) = 0$;
forall the $h \in \{0, \ldots, d_{max}\}$ **do**
  $f(s, h) = 0$;
end
forall the $i \in V \setminus \{s\}$ **do**
  $MinCost(i) = \infty$;
  forall the $h \in \{0, \ldots, d_{max}\}$ **do**
    $f(i, h) = \infty$;
  end
end
forall the $h \in \{0, \ldots, d_{max} - 1\}$ **do**
  forall the $j \in S_h$ *with* $f(j, h) \leq MinCost(t)$ **do**
    $MinCost(j) = \min(MinCost(j), f(j, h))$;
  end
  forall the $j \in S_h$ *with* $f(j, h) \leq MinCost(t)$ **do**
    for $\{j, i\} \in E$ *with* $d_{\{j,i\}} + h \leq d_{max}$ *and*
    $f(j, h) + w_{\{j,i\}} < \min(MinCost(i), MinCost(t), f(i, h + d_{\{j,i\}}))$ **do**
      if $i \notin S_{h+d_{\{j,i\}}}$ **then**
        $S_{h+d_{\{j,i\}}} = S_{d_{\{j,i\}}+h} \cup \{i\}$;
      end
      $f(i, h + d_{\{j,i\}}) = f(j, h) + w_{\{j,i\}}$;
      if $i = t$ **then**
        $MinCost(t) = f(i, h + d_{\{j,i\}})$;
      end
    end
  end
end

Algorithm 4.1: Weight Constraint Shortest Path Problem

CHAPTER 5

Acyclic Problem Variant

As shown before (cf. Chapter 2), optimal solutions to the NDPR may contain cycles. For practical applications, however, it might be required to enforce acyclic solutions.

5.1 Solution Properties

In terms of the RPP the relevance of acyclic solutions has already been considered by Sen et al. [36]. They distinguish between three types of connections between the commodities. For the acyclic case only simple paths between a pair of vertices are allowed, and for the cyclic case all feasible connections (including cycles) are possible. In addition, they consider a third case for which only connections that traverse an edge twice in the same direction are prohibited, i.e., path (a, b, c, b, d, e) is allowed but (a, b, c, d, b, c, e) is not since edge (b, c) is used twice in the same direction. According to Theorem 2.1.3 this case is irrelevant for the NDPR.

As a consequence of this result we want to consider acyclic solutions by enforcing that every pair $(i, j) \in \mathcal{K}$ has to be connected by a feasible path that is also simple.

In Chapter 2 we presented an instance with a cyclic solution. If we enforce an acyclic solution we get a different result as shown in Figure 5.1. By enforcing this additional property we reduce the amount of possible solutions. As a result this might prevent the former optimal solution and increase the costs significantly.

Observe that enforcing simple paths as connections for the commodities does not mean that the overall solution is acyclic. Figure 5.2 shows an instance with three commodities that are connected with simple paths. When considering the union of the individual paths we get a cycle, which has to be considered when stating MILP models. Note that this is only relevant if we consider more than one source:

Corollary 5.1.1. *In an optimal acyclic solution there exists for every source $u \in \mathcal{K}_S$ a directed graph rooted at u having a feasible connection to every target $v \in K(u)$ visiting each vertex at most once.*

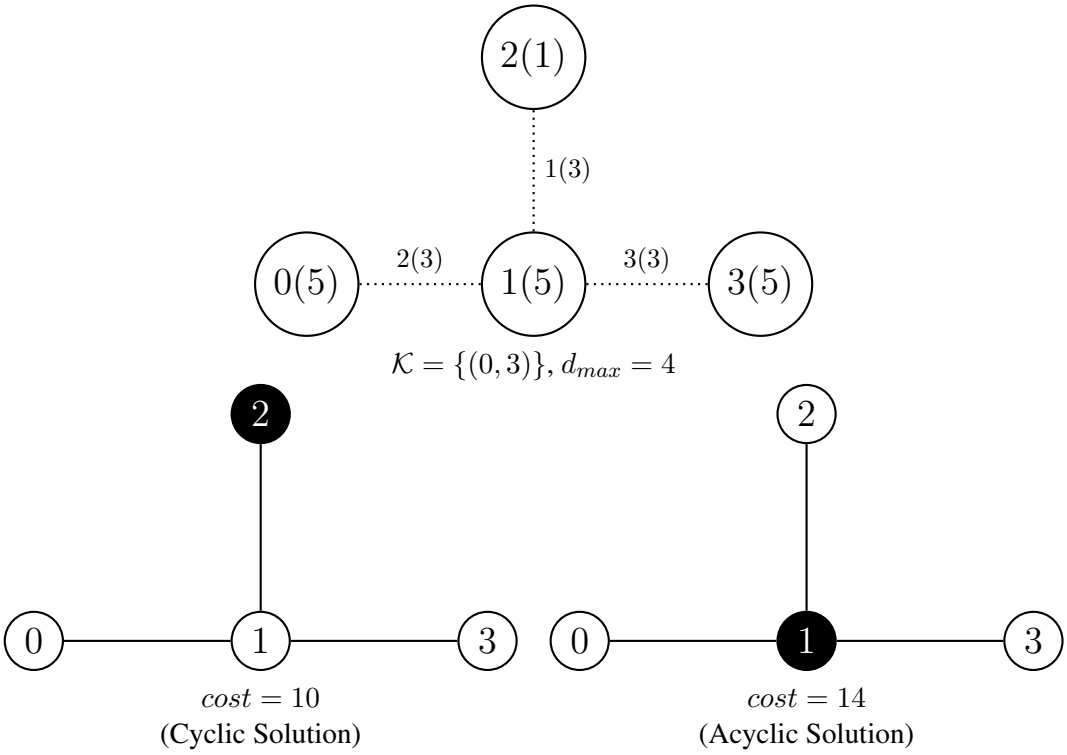
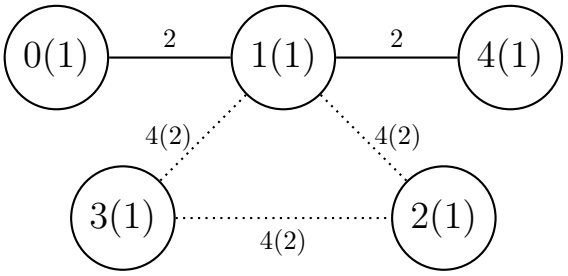


Figure 5.1: Different Solutions

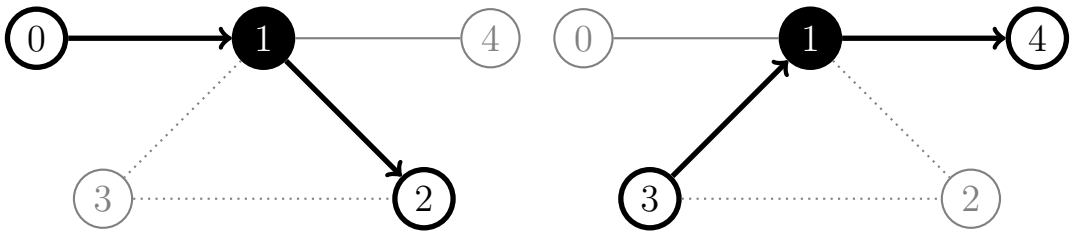
Proof. By the definition of an acyclic solution there has to be for every $(u, v) \in \mathcal{K}$ a simple feasible path from u to v. In Theorem 2.1.2 we have shown how to merge paths with a common source into a single graph without increasing the in-degree of any vertex while preserving feasibility and optimality. If we consider only simple paths we know that each vertex has an in-degree of at most one in the resulting graph. □

Note that Figure 5.2 depicts the only optimal solution for $\mathcal{K} = \{(0, 2), (2, 3), (3, 4)\}$ and $d_{max} = 6$. The numbers next to the edges denote their delay and the numbers in parentheses their costs. The numbers next to the vertices are the relay costs. This instance shows that a similar condition as stated in Corollary 5.1.1 cannot be proposed when considering more than one source at the same time.

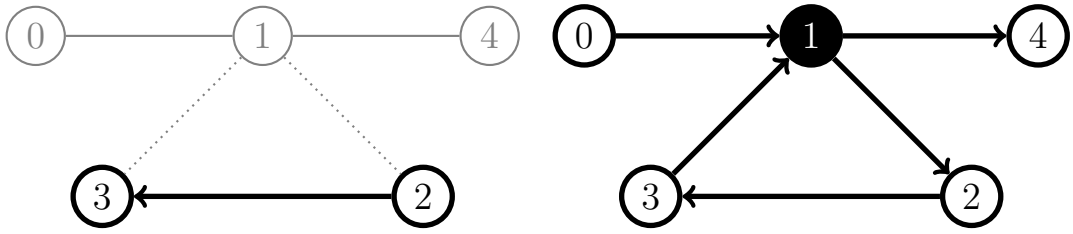
In this chapter we are going to present adaptations of some of our models that only allow simple paths for the connection of the commodities. Note that at this point we focus on the necessary adjustments. For detailed explanations of the underlying models please refer to Chapter 3 (models on layered graphs) and 4 (models on communication graphs).



$\mathcal{K} = \{(0,2),(2,3),(3,4)\}$, $d_{max} = 6$



Simple path connecting $(0,2) \in \mathcal{K}$ | Simple path connecting $(3,4) \in \mathcal{K}$



Simple path connecting $(2,3) \in \mathcal{K}$ | Union of simple paths connecting the individual commodities

Figure 5.2: Exemplary Instance

5.2 Models on Communication Graphs

Unfortunately, some considerations concerning communication graphs make it difficult to enforce acyclic solutions. The reason for this is that we do not consider the individual edges of which a communication edge consists if free edges are involved. Moreover, the original formulation also requires no information on the orientation of the edges. It does not matter which of the arcs is used, we only need to know if it is used. Even worse we required this information per source since it is possible that all pairs in $\mathcal{K}$ can be connected with simple paths but when looking at them altogether we might end off with cycles (see Figure 5.2).

The required modifications would destroy the properties that make the MILP models presented in the previous chapter efficient. Since the necessary adjustments affect the path variables and the associated linking constraints, none of the models on communications graphs is suited to enforce acyclic solutions.

5.3 Models on Layered Graphs

The models on layered graphs are in general well suited to enforce acyclic solution. For some constraints we obtain stronger restrictions due to the required modifications. However, note that model L_{CUTS} is unsuitable to enforce acyclic solutions since it considers more than one source per variable set. In the following we present adaptations of our two most promising approaches based on multiple layered graphs.

5.3.1 Flow Model

We start by presenting a modified version of the layered single-commodity flow formulation on multiple layered graphs (L_{SCFM}). For a detailed description of the underlying model we refer to Section 3.2.2.

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^+(u_0)} f_a^u = |K(u)| \qquad \forall u \in \mathcal{K}_S, u_0 \in V_{\mathrm{L}}^u \quad (5.1)$$

$$0 \leq \sum_{a \in \delta^-(i_l)} f_a^u - \sum_{a \in \delta^+(i_l)} f_a^u \leq 1 \qquad \forall u \in \mathcal{K}_S, \forall i_l \in V_{\mathrm{L}}^u, i \in K(u) \quad (5.2)$$

$$\sum_{a \in \delta^-(i_l)} f_a^u - \sum_{a \in \delta^+(i_l)} f_a^u = 0 \qquad \forall u \in \mathcal{K}_S, \forall i_l \in V_{\mathrm{L}}^u, i \neq u, i \notin K(u) \quad (5.3)$$

$$\sum_{i_m \in V_{\mathrm{L}}^u} \sum_{a \in \delta^-(i_m)} f_a^u - \sum_{a \in \delta^+(i_l)} f_a^u \leq 1 \qquad \forall u \in \mathcal{K}_S, \forall i_l \in V_{\mathrm{L}}^u, i \in K(u) \quad (5.4)$$

$$\sum_{i_m \in V_{\mathrm{L}}^u} \sum_{a \in \delta^-(i_m)} f_a^u - \sum_{a \in \delta^+(i_l)} f_a^u = 0 \qquad \forall u \in \mathcal{K}_S, \forall i_l \in V_{\mathrm{L}}^u, i \neq u, i \notin K(u) \quad (5.5)$$

$$\sum_{i_l \in V_{\mathrm{L}}^u} \left(\sum_{a \in \delta^-(i_l)} f_a^u - \sum_{a \in \delta^+(i_l)} f_a^u \right) = 1 \qquad \forall u \in \mathcal{K}_S, i \in K(u) \quad (5.6)$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u : l > 0 \wedge m > 0} f_{(i_l, j_m)}^u \leq |K(u)| \cdot (1 - y_i) \qquad \forall u \in \mathcal{K}_S, \forall i \in V, i \neq u \quad (5.7)$$

$$\sum_{(i_l, i_0) \in A_{\mathrm{L}}^u} f_{(i_l, i_0)}^u \leq |K(u)| \cdot y_i \qquad \forall u \in \mathcal{K}_S, \forall i \in V, i \neq u \quad (5.8)$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u} f_{(i_l, j_m)}^u + \sum_{(j_l, i_m) \in A_{\mathrm{L}}^u} f_{(j_l, i_m)}^u \leq |K(u)| \cdot x_e \qquad \forall u \in \mathcal{K}_S, \forall \{i, j\} \in E^* \quad (5.9)$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \quad (5.10)$$

$$x_e \in \{0, 1\} \qquad \forall e \in E^* \quad (5.11)$$

$$0 \leq f_a^u \leq |K(u)| \qquad \forall u \in \mathcal{K}_S, \forall a \in A_{\mathrm{L}}^u \quad (5.12)$$

To guarantee that all commodities are connected with simple paths we have to ensure that across all layers at most one vertex copy has an in-coming arc. To accomplish this we add Constraints (5.4) and (5.5). They ensure flow balance between the total in-flow on all layers and the out-flow on each layer. Since these constraints might be difficult to understand we are going to prove them in the following.

Lemma 5.3.1. *Constraints* (5.2) *to* (5.6) *ensure that the solution contains no cycles.*

Proof. We assume that the constraints still allow cycles and derive a contradiction. W.l.o.g. we assume there is in-flow on two layers for some vertex $i \in V$. Then, we also require out-flow on both layers due to Constraints (5.2) and (5.3).

Case 1: Assume we are dealing with a non-target vertex. Furthermore, let g and h be the amount of in-flows s.t. $g \geq h > 0$. Thus, we require out-flows of the same amount. As a result some Constraints of type (5.5) will be violated since $(g + h) - g \neq 0$ and also $(g + h) - h \neq 0$.

Case 2: Now we deal with the case of a target vertex. Again we consider in-flows g and h s.t. $g \geq h > 0$. According to Constraints (5.2) and (5.6) we have either out-flows g and $(h - 1)$ or $(g - 1)$ and h. Thus, Constraints (5.4) supply either of the two lines:

$$\begin{array}{llll} 1. & (g+h)-g \leq 1 & \wedge & (g+h)-(h-1) \leq 1 \quad \text{, or} \\ 2. & (g+h)-(g-1) \leq 1 & \wedge & (g+h)-h \leq 1 \end{array}$$

After simplification we get:

$$\begin{array}{llll} 1. & h \leq 1 & \wedge & g \leq 0 \quad \text{, or} \\ 2. & h \leq 0 & \wedge & g \leq 1 \end{array}$$

Since $g \geq h > 0$ we obtain a contradiction in both cases which concludes the proof. □

Constraints (5.7) are the modification of Constraints (3.29) from model L_{SCFM}. We remove $\cdot 2$ on the right-hand side which accommodates for the fact that without loops the maximum outflow per vertex across all layers is limited by $|K(u)|$. The last modification is to alter the sum of Constraints (5.9). In the original version we have a sum per arc of an edge. When cycles are not allowed we know that it is not possible that both arcs are required. Thus, we can combine the two sums.

5.3.2 Cut Model

The second model we want to adapt is the layered cut formulation on multiple layered graphs (L_{CUTM}) presented in Section 3.2.3. The corresponding MILP model reads as follows:

$$\min \sum_{i \in V} c_i y_i + \sum_{e \in E^*} w_e x_e$$

$$\sum_{a \in \delta^-(W)} X_a^u \geq 1 \qquad \forall u \in \mathcal{K}_S, \forall v \in K(u), \{v_l | v_l \in V_{\mathrm{L}}^u\} \subseteq W \subset V_{\mathrm{L}}^u, u \notin W \tag{5.13}$$

$$\sum_{i_l \in V_{\mathrm{L}}^u : l>0} \sum_{a \in \delta^-(i_l)} X_a^u \leq 1 \qquad \forall u \in \mathcal{K}_S, \forall i \notin K(u), i \neq u \tag{5.14}$$

$$\sum_{i_l \in V_{\mathrm{L}}^u : l>0} \sum_{a \in \delta^-(i_l)} X_a^u = 1 \qquad \forall u \in \mathcal{K}_S, \forall i \in K(u) \tag{5.15}$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u : l>0 \wedge m>0} X_{(i_l,j_m)}^u \leq |K(u)| \cdot (1 - y_i) \qquad \forall u \in \mathcal{K}_S, \forall i \in V, i \neq u \tag{5.16}$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u : l>0 \wedge m>0} X_{(i_l,j_m)}^u \leq 1 - y_i \qquad \forall u \in \mathcal{K}_S, \forall i \in V, \forall \{i,j\} \in E, i \neq u, j \neq u \tag{5.17}$$

$$\sum_{(i_l, i_0) \in A_{\mathrm{L}}^u} X_{(i_l,i_0)}^u \leq y_i \qquad \forall u \in \mathcal{K}_S, \forall i \in V \tag{5.18}$$

$$\sum_{(i_l, j_m) \in A_{\mathrm{L}}^u} X_{(i_l,j_m)}^u + \sum_{(j_l, i_m) \in A_{\mathrm{L}}^u} X_{(j_l,i_m)}^u \leq x_e \qquad \forall u \in \mathcal{K}_S, \forall \{i,j\} \in E^* \tag{5.19}$$

$$X_a^u \in \{0,1\} \qquad \forall u \in \mathcal{K}_S, \forall a \in A_{\mathrm{L}}^u \tag{5.20}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{5.21}$$

$$x_e \in \{0,1\} \qquad \forall e \in E^* \tag{5.22}$$

This model is obtained from the model L_{CUTM} after the following modifications. First of all we split the constraints

$$\sum_{i_l \in V_L^u : l>0} \sum_{a \in \delta^-(i_l)} X_a^u \leq 2 - y_i \quad \forall u \in \mathcal{K}_S, \forall i \in V, i \neq u$$

w.r.t. target and non-target vertices resulting in Constraints (5.14) and (5.15). All non-target vertices are reached at most once and targets are reached exactly once. Note that this is already sufficient to prevent cycles. Due to the binary arc variables it is much easier to ensure cycle-free solutions, compared to the flow formulation. Constraints (5.16) originally had a Big-M of $(|K(u)| + 1)$. This can to be reduced to $|K(u)|$ since the additional arc for the loop is no longer allowed. Constraints (5.17) ensure that an arc leaving node i at some layer greater than zero can only be used if no relay was installed at i. For Constraints (5.19) we again combine the two sums as described for the flow model.

CHAPTER 6

Computational Results

In this chapter we are going to present the computational results for the presented algorithms. We start with some remarks on preprocessing that help to reduce the size of the problem instances for certain cases. Then, we present details for the algorithms that worked well in practice. We especially give details about the parameters used for computing the results. Furthermore, we are going to discuss the used sub-algorithms for cutting and pricing. Then, we give details on the used test instances. Finally, we present the test results of our algorithms w.r.t. these test instances.

6.1 Preprocessing

The easiest form of preprocessing is to remove all edges from both E^0 and E^* having a delay larger than d_{max}. Using one of these edges would result in an immediate violation of the delay bound. Thus, we do not consider them during the solution process.

After these edges have been removed it makes sense to check if there exists a feasible solution for the given instance. If the graph consists of more than one connected component and there exists a pair $(u, v) \in \mathcal{K}$ s.t. u and v belong to different components then there cannot be a feasible solution. If the start and end point of each commodity are part of the same connected component the instance has a feasible solution.

The next thing we want to consider are pairs in $\mathcal{K}$ that can be connected solely within the free edges and without the use of relays. To find such pairs we compute the shortest path distance in $G = (V, E^0)$ for all vertex pairs using an all-pairs shortest path algorithm (see [11] and [24]) and then check if the distance is smaller than or equal to the delay bound. Note that these pairs can be connected without influencing the objective. Therefore, we remove these pairs from $\mathcal{K}$. Next, we solve the reduced problem and check if the removed pairs are connected in the obtained solution. If some pair (u, v) is not connected we compute the shortest path between u and v in $G = (V, E^0)$ and add the missing free edges to the solution. Since these edges have no cost this leaves the overall costs unchanged and thus preserves optimality of the solution.

The next preprocessing technique we want to mention has been introduced by Chen et al. [6] for the RLP but can also be applied to the NDPR with some modifications and extensions. To this end, we consider vertices i that are leaves in the graph and appear as source or target in $\mathcal{K}$. Let $\{j, i\}$ be the edge connecting i to the graph. If all combinations of in-coming edges to j and the edge $\{j, i\}$ exceed the delay bound j has to be a relay in any feasible solution. Otherwise it would not be possible to reach i within the delay bound. Furthermore if $\{j, i\}$ is an augmenting edge we also know that this edge is required since it is the only possible way of connecting i to the graph. If j has degree two, i.e., j is a leaf if we do not consider i, it makes sense to apply this procedure again at j. In general we iterate this procedure until we arrive at some vertex of degree greater than two. This leads to Algorithm 6.1.

Input: $G = (V, E)$ $(E = E^0 \cup E^*)$
Input: Set $\mathcal{K}$ of commodities
Input: Delay bound d_{max}
Result: Set $\mathcal{R}$ of required relays
Result: Set $\mathcal{E}$ of required augmenting edges

$\mathcal{S} = \{i | i \in V, \delta(i) = 1 \wedge \exists j((i, j) \in \mathcal{K} \vee (j, i) \in \mathcal{K})\}$;
$\mathcal{R} = \emptyset$;
$\mathcal{E} = \emptyset$;
forall the $i \in S$ **do**
  $\mathcal{S} = \mathcal{S} \setminus \{i\}$;
  $e = \{j, i\}$;
  if $\forall \{k, j\} \in \delta(j) \setminus \{\{j, i\}\}(d_{\{k,j\}} + d_e > d_{max})$ **then**
    $\mathcal{R} = \mathcal{R} \cup \{j\}$;
    if $\delta(j) = 2$ **then**
      $\mathcal{S} = \mathcal{S} \cup \{j\}$;
      $V = V \setminus \{i\}$;
      $E = E \setminus \{j, i\}$;
    end
  end
  if $e \in E^*$ **then**
    $\mathcal{E} = \mathcal{E} \cup \{e\}$;
  end
end

Algorithm 6.1: Preprocessing Algorithm

The first three preprocessing techniques have been used in the sense that we only consider instances for which they do not apply. Especially, we only consider instances that are connected, i.e., they consist of a single connected component. Furthermore, we define the set $\mathcal{K}$, s.t. pairs that can be connected using only free edges are not part of it. We did not use the last preprocessing technique in our implementation.

Graph Transformation	Model	Type	Results
Layered Graphs	L_{CUTS}	B&C	
	L_{MCFM}	compact	
	L_{SCFM}	compact	✓
	L_{CUTM}	B&C	✓
Communication Graphs	CG_{CUTS}	B&P&C	✓
	CG_{MCFM}	B&P	✓
	CG_{SCFM}	B&P	✓
	CG_{CUTM}	B&P&C	

Table 6.1: Algorithm Overview

6.2 Algorithm Details

In this section we are going to give an overview on the presented algorithms. We also discuss important implementation details. Although we have preliminary tested all the algorithms, we will focus on those with the best performance. Table 6.1 summarizes the presented MILP formulations and associated algorithms. B&C stands for Branch-and-Cut, B&P stands for Branch-and-Price and B&P&C stands for Branch-Price-and-Cut, respectively. Only the models marked in the 'Results' column will be discussed in detail.

Furthermore, we recall that the computational study is only performed for the NDPR in which cycles are allowed.

6.2.1 Separation

Some of our algorithms use cutting planes to ensure that all commodities can communicate with each other. The amount of these constraints is in general exponential. Thus, it is not feasible to add all of them into a black-box MILP solver. They need to be dynamically generated by separation procedures. In this section we give details on the used separation methods.

Separation on Layered Graphs

We are going to explain the separation procedure w.r.t. the model L_{CUTM}[1]. For the layered graph approaches we generate vertex copies w.r.t. the possible layers. We do not know at which layer we will reach the target. Thus, we have to consider all copies (on layers greater than zero) as potential targets. Disconnected source-target pairs can be found by computing the maximum flow between source and target with arc-capacities set according to the current solution. For the maximum flow computation we use the implementation by Cherkassy and Goldberg in [8]. To check if a source is connected to its target we need to check if it is connected to one of the target copies. To avoid multiple maximum flow computations per target we create an artificial linking vertex that is reached by all target copies. Hence for each target $v \in K(u)$ we create a new vertex v^t and arcs $A_v^t = \{(v_l, v^t) | v_l \in V_{\mathrm{L}}^u, l > 0\}$. The capacities of the arcs in A_v^t are set

[1]The approach for L_{CUTS} is almost the same except that we only have a single set of arc values.

to two. Then we compute per source-target pair $(u, v) \in \mathcal{K}$ the maximum flow between u and v^t. If the maximum flow is smaller than one then the pair is not connected and we identified a violated inequality.

The maximum flow algorithm we use gives us not only the flow value but also two cut sets S_u and S_v. Hence, the maximum flow computation between u and v for current arc values X^u is defined by $f_{max} = MaxFlow(G^u_{\mathrm{L}}, X^u, u, v, S_u, S_v)$. S_u contains the source u and induces a minimum cut closest to u. Similarly, S_v contains v and induces a minimum cut closest to it. Thus, $f = \sum_{a \in \delta^+(S_u)} X^u_a = \sum_{a \in \delta^-(S_v)} X^u_a$ holds. We can add forward-cuts using set S_u, back-cuts using set S_v, or both. Tests showed that it is most efficient to only use back-cuts. Algorithm 6.2 illustrates the separation procedure.

Input: Graphs, $G^u_{\mathrm{L}} = (V^u_{\mathrm{L}}, A^u_{\mathrm{L}}), \forall u \in \mathcal{K}_S$
Input: Current values X^uon arcs of $G^u_{\mathrm{L}}, \forall u \in \mathcal{K}_S$
Input: Set of sources $\mathcal{K}_S$
Result: Violated cut-inequalities

```
1  forall the u ∈ K_S do
2      X̃^u = X^u;
3      forall the v ∈ K(u) do
4          G̃^u_L = (V^u_L ∪ {v^t}, A^u_L ∪ A^t_v);
5          X̃^u_a = 2, ∀a ∈ A^t_v;
6          f_max = MaxFlow(G̃^u_L, X̃^u, u, v, S_u, S_v);
7          if f_max < 1 then
8              Add violated cut: Σ_{a∈δ⁻(S_v)} X^u_a ≥ 1;
9              forall the a ∈ δ⁻(S_v) do
10                 X̃^u_a = 2;
11             end
12         end
13     end
14 end
```

Algorithm 6.2: Separation Procedure (L_{CUTM})

Note that we adjust the arc values of the backward cut on line 10. Since we only have one arc set per source, the added cut for some target might also affect the others. To reduce the amount of added cuts we only add arc-disjoint cut-inequalities. This is ensured by setting arc capacities to two, after we added a violated constraint associated to $\delta^-(S_v)$ (cf. lines 9 to 11).

Due to the numerical instability, we prefer sparse cuts, and enforce them by searching for violet cut-set inequalities of minimal cardinality.

Separation on Communication Graphs

Model CG_{CUTS} also uses vertex copies. However, we know that we require a connection from u_2 to v_1 for all $(u, v) \in \mathcal{K}$. Thus, we can compute the maximum flow without any modifications to the graph as shown in Algorithm 6.3.

Input: $G^2_C = (V^2_C, A^2_C)$
Input: Current arc values X
Input: Set of sources $\mathcal{K}_S$
Result: Violated cut-inequalities

```
1  X̃ = X;
2  forall the u ∈ K_S do
3      forall the v ∈ K(u) do
4          f_max = MaxFlow(G²_C, X̃, u_2, v_1, S_u, S_v);
5          if f_max < 1 then
6              Add violated cut: Σ_{a∈δ⁻(S_v)} X^u_a ≥ 1;
7              forall the a ∈ δ⁻(S_v) do
8                  X̃_a = 2;
9              end
10         end
11     end
12 end
```

Algorithm 6.3: Separation Procedure (CG_{CUTS})

We again enforce arc-disjoint cut inequalities by the adjustment on line 8 and we also search for sparse cuts of minimum cardinality.

Implementation Details

Concerning the implementation separation is always performed on integral LP solutions (LazyConstraintCallbackI). When the LP solution is fractional we only add cuts for which $f_{max} < 0.5$ (UserCutCallbackI). However, when computing the quality of LP bounds of the corresponding models we add cuts for which $f_{max} < 1$.

6.2.2 Column Generation

We use column generation to deal with the exponential amount of path variables used in the communication graph models. We already mentioned that the relevant subproblem for pricing is a Weight Constrained Shortest Path Problem (WCSPP) which is in our case defined by:

$$\forall b \in C^* \; R_b = \underset{p \in P(b)}{\arg\min} \sum_{e \in E^* \cap p} \alpha^e_b$$

We look for the shortest path p connecting the pair b within the delay bound d_{max}. The arc lengths of the augmenting edges are given by the dual values of the linking constraints for the augmenting edges and the lengths of the free edges remain zero. The weights of the edges are set according to their delays. We already argued that this problem can be solved using the algorithm by Gouveia et al. [20].

After we have identified the shortest path, we still need to check if this path gives us negative reduced costs. To this end, we have to consider the dual variables μ for the linking of path and

Model	B
CG_{CUTS}	$2 \cdot \mu_b$
CG_{MCFM}	$\sum_{(u,v)\in\mathcal{K}} \mu_b^{uv}$
CG_{SCFM}	$\sum_{u\in\mathcal{K}_S} (\lvert K(u)\rvert \cdot \mu_b^u)$
CG_{CUTM}	$\sum_{u\in\mathcal{K}_S} \mu_b^u$

Table 6.2: Reduced Cost Bounds

arc or flow variables, respectively. The sums of the μ-variables per connection $b \in C^*$ w.r.t. the individual models are shown in Table 6.2.
Using column B of the table we can state the pricing subproblem in a more general form:

$$\arg\min_{p\in P(b)} \left\{ 0 - \left(B - \sum_{e\in E^*\cap p} \alpha_b^e \right) \right\}$$

Thus, the length of the shortest path has to be smaller than B to give us negative reduced costs. Per pricing iteration we might find at most $|C^*|$ columns with negative reduced costs. Hence, we have the choice to add all of them or only a subset in each iteration. After some preliminary tests we decided to only add the first detected column with negative reduced costs and then finish the current iteration. Algorithm 6.4 illustrates the pricing procedure. $WCSPP(G, u, v, d_{max}, p)$ computes the length of the shortest path p from u to v in G with a maximum delay of d_{max} and stores the obtained path in p.

Initial set of Columns

To start with a feasible LP model, we initiate every connection $b \in C^*$ with one possible realization. To this end, we choose a feasible connection having minimal delay. Such a connection can easily be found with Dijkstra's algorithm (see [9]) using the edge delays as costs.

Column generation at the Root Node vs. Branch-and-Price

To find an optimal solution we are required to do column generation in the Branch-and-Bound tree, i.e., to perform Branch-and-Price. Unfortunately, this is not supported by CPLEX and we are limited to column generation at the root node. Therefore, our CPLEX implementation works as follows: Complete column generation is performed at the root node. After that, branch-and-bound is executed on the sub-model with the columns provided at the root node. This is a heuristic procedure.

Tests showed that for all NDPR instances whenever this approach terminated within the time limit, the returned solution has been optimal. However, this is not the case for some ARLP instances. Instances for which the reduced approach cannot find the optimal solution can be recognized by having a negative optimality gap.

Input: $G = (V, A(E), d)$ $(E = E^0 \cup E^*)$
Input: Dual values α and μ
Input: Set of connections C^*
Result: Column with negative reduced costs

forall the $\{i, j\} \in E^0$ **do**
  $l_{ij} = 0$;
  $l_{ji} = 0$;
end
forall the $b \in C^*$ **do**
  forall the $e = \{i, j\} \in E^*$ **do**
    $l_{ij} = \alpha_b^e$;
    $l_{ji} = \alpha_b^e$;
  end
  $(u, v) = b$;
  $L = WCSPP(G, u, v, d_{max}, p)$;
  Set B according to μ ; `// cf. Table 6.2`
  if $L < B$ **then**
    Add column λ_b^p to the model;
    Stop iteration;
  end
end

Algorithm 6.4: Column Generation Procedure

To investigate the influence of doing full Branch-and-Price we also implemented the respective models in SCIP. Details on the results will be given in Section 6.5.

6.2.3 Column-and-Row Generation at the Root Node

Some of our algorithms even require the combination of cutting planes and column generation. Since CPLEX provides no direct support for column generation we implemented our own loop to add columns at the root node. In order to remain consistent with the usual approach and the way SCIP implements this we do pricing until we find no further columns with negative reduced costs and then check for cutting planes once before we continue with pricing. Algorithm 6.5 shows the resulting Branch-Price-and-Cut loop.

6.2.4 Optional Constraints

In addition to the constraints that make the considered models valid, we also introduced various optional ones. All mentioned constraints are implemented and used in our computational experiments except for Constraints (3.38) from model L_{CUTM} (see Subsection 3.2.3):

```
// addColumns() returns true if it added columns and false
   otherwise
// addCuts() returns true if it added cuts and false
   otherwise
repeat
    repeat
        solve_LP();
        col_added = addColumns();
    until col_added == false;
    cut_added = addCuts();
until cut_added == false;
```

Algorithm 6.5: Branch-Price-and-Cut Loop

	LP bound	compact	B&C	B&P	B&C&P
LP solver	dual simplex	dual simplex	dual simplex	dual simplex	dual simplex
Threads	1	1	1	1	1
Presolving		✓	✓	✓	✓
Probing		✓	✓	✓	✓
Heuristics		✓	✓	✓	✓
Cuts		✓			

Table 6.3: Solver Settings

$$\sum_{(i_l,j_m)\in A^u_{\mathrm{L}}: l>0\wedge m>0} X^u_{(i_l,j_m)} \leq 2\cdot(1-y_i) \qquad \forall u\in\mathcal{K}_S, \forall i\in V, \forall\{i,j\}\in E, i\neq u, j\neq u$$

These constraints are essentially a disaggregated variant of Constraints (3.37):

$$\sum_{(i_l,j_m)\in A^u_{\mathrm{L}}: l>0\wedge m>0} X^u_{(i_l,j_m)} \leq (|K(u)|+1)\cdot(1-y_i) \qquad \forall u\in\mathcal{K}_S, \forall i\in V, i\neq u$$

We tried to implement Constraints (3.38) by adding all of them initially and by adding them by means of a separation procedure. It turned out that they are almost never violated. Since their benefits do not compensate for the additionally required runtime we decided to remove them from the implementation and only keep Constraints (3.37).

6.3 Solver Configuration

In this section we are going to explain the used solver settings. Table 6.3 gives an overview of the configurations for the respective model types.

The first choice regards the used LP solver. For column generation models usually the *primal simplex method* is preferred as it maintains primal feasibility after a column has been added. Another common choice are interior point methods like the *barrier method*. We made tests with the available algorithms and figured out that the *dual simplex method* is the best choice (even for column generation). We use for all our models only a single thread. Furthermore, we always use *presolving*, *probing* and the solvers *heuristics*. However, we only allow solver *cuts* for compact models. When computing the LP bound solely we disable all additional help from the solver.

In particular we use CPLEX 12.6 and SCIP 3.1.0. To obtain comparable results we use CPLEX as LP solver for SCIP. We also tried to use the same settings as far as possible. We use SCIP only to implement full Branch-and-Price since this is not supported by CPLEX.

6.4 Test Instances

In this section we are going to give details on the used test instances. We use modified versions of instances introduced in the previous literature and an entirely new set of instances.

6.4.1 NDPR instances

These instances have originally been introduced by Konak [25]. The instances have been randomly generated by placing vertices on a grid and connecting them. The instances use Euclidean distances for the delays. The costs are either set equal to the delays (Type I) or to $c_{\{i,j\}} = d_{max} - d_{\{i,j\}}$ (Type II). The instances with their parameters are shown in Table 6.4. For each vertex size the instances with the same delay maximum use the same underlying graph. Only the number of commodities differs within instances of the same group. The original instances use the Euclidean distances directly and thus have fractional costs and delays. To better fit into the domain of MILP we decided to round these values up to obtain integral instances.

6.4.2 ARLP instances

These instances are entirely new. We call them augmented RLP (ARLP) instances since all commodities need to communicate like for the RLP. However, not all edges have a cost of zero. Thus, it is an augmented variant that can be solved by the NDPR.

The instances have been generated as follows. We start with a 100×100-grid on which we randomly place vertices. Then we add edges between all vertices not farther apart than a Euclidean distance of 30. We set the delay of an edge equal to its rounded up Euclidean distance. We decide whether an edge is augmenting or free at random. We generate instances for which the chance of a free edge is either $20, 50$ or 80% ($20F, 50F, 80F$). For augmenting edges we set their cost $c_{\{i,j\}}$ randomly according to a normal distribution with parameters $\mu = d_{\{i,j\}}, \sigma = 5$ (rounded up). For the relay costs we first compute the average costs of the augmenting edges $\bar{c}$. Then we set the costs of the relays randomly according to the normal distribution $\mu = 10 \cdot \bar{c}, \sigma = 20$ (rounded up). We set $d_{max} = 50$ and enforce connectivity for all pairs except those that can be connected using solely free edges, i.e., $\mathcal{K} = \{(i,j) | (i,j) \in V \times V, i < j\} \setminus C^0$ (see Section 6.1).

Instance	$\|V\|$	$\|E^*\|$	$\|E^0\|$	$\|\mathcal{K}\|$	d_{max}
40N5K30L	40	198	0	5	30
40N5K35L	40	272	0	5	35
40N10K30L	40	198	0	10	30
40N10K35L	40	272	0	10	35
50N5K30L	50	279	0	5	30
50N5K35L	50	372	0	5	35
50N10K30L	50	279	0	10	30
50N10K35L	50	372	0	10	35
60N5K30L	60	305	0	5	30
60N5K35L	60	412	0	5	35
60N10K30L	60	305	0	10	30
60N10K35L	60	412	0	10	35
80N5K30L	80	641	0	5	30
80N5K35L	80	853	0	5	35
80N10K30L	80	641	0	10	30
80N10K35L	80	853	0	10	35
160N5K30L	160	2773	0	5	30
160N5K35L	160	3624	0	5	35
160N10K30L	160	2773	0	10	30
160N10K35L	160	3624	0	10	35

Table 6.4: NDPR Instances

We generate two instances per combination of the vertex number and the probability of creating free edges. This first set of instances is shown in Table 6.5. To investigate the influence of a reduced number of commodities we generated a second set of instances. These instances have been created by removing $\sim 75\%$ of the commodities of the original ARLP-instance. We refer to these instances as 'ARLP - p25'. The second set of instances is shown in Table 6.6.

6.5 Test Results

In this section we are going to present the computational results of our algorithms performed on the introduced instances. The test runs have been executed on a 2x Intel Xeon E5540, 2.53 GHz Quad Core with 24GB RAM and comparable hardware. The execution time limit has been set to 7200 seconds.

The column generation approaches have been implemented with column generation only at the root node followed by branch-and-bound in CPLEX (CG) and also with full branch and price in SCIP (CG^{BP}). Nevertheless, the LP bound for both implementations has to be the same. This holds for models CG_{MCFM} and CG_{SCFM}. Unfortunately, we obtained different LP bounds for model CG_{CUTS}. In the following we provide both results. We believe that both solvers could uses some additional features that we have not been able to deactivate that cause the different LP bound. As a reference value, we took those obtained by CPLEX.

Instance	$\|V\|$	$\|E^*\|$	$\|E^0\|$	$\|\mathcal{K}\|$	d_{max}
40N50L20F_A	40	124	26	724	50
40N50L20F_B	40	123	35	688	50
40N50L50F_A	40	78	89	513	50
40N50L50F_B	40	72	71	586	50
40N50L80F_A	40	32	146	443	50
40N50L80F_B	40	35	154	423	50
50N50L20F_A	50	212	44	1111	50
50N50L20F_B	50	235	59	1022	50
50N50L50F_A	50	157	132	719	50
50N50L50F_B	50	132	117	873	50
50N50L80F_A	50	51	175	788	50
50N50L80F_B	50	58	212	682	50
60N50L20F_A	60	269	72	1549	50
60N50L20F_B	60	268	63	1588	50
60N50L50F_A	60	204	216	1036	50
60N50L50F_B	60	200	197	1103	50
60N50L80F_A	60	85	311	854	50
60N50L80F_B	60	74	283	1041	50
80N50L20F_A	80	557	145	2599	50
80N50L20F_B	80	545	124	2659	50
80N50L50F_A	80	345	313	1922	50
80N50L50F_B	80	375	366	1902	50
80N50L80F_A	80	148	548	1834	50
80N50L80F_B	80	121	536	1709	50

Table 6.5: ARLP Instances

In the following we provide for every set of instances a chart showing the cumulative percentage of instances solved w.r.t. the runtime. Furthermore, we present results considering the quality of the LP bounds. We show the runtime until the LP bound has been found and give the gap between the LP bound and the best known feasible solution in percent. Furthermore, we provide a table showing the total computation time and the gap between lower and upper bound (optimality gap).

We marked in every table row the best value unless too many approaches have the same results. Due to the problems with the LP bound mentioned above, we do not consider the SCIP results when marking optimal results in the LP tables. Furthermore, the SCIP implementation has problems with the memory limit for some instances. The concerned instances have been marked with *ML* in the runtime column. Moreover, we encountered a confirmed CPLEX bug[2] w.r.t. preprocessing for some instances using model L_{CUTM}. The concerned instances have been marked with *CB* in the runtime column.

[2]For details see https://www.ibm.com/developerworks/community/forums/html/topic?id=94f6acaa-60cf-48e8-a773-bbb394c04633&ps=25.

Instance	$\|V\|$	$\|E^*\|$	$\|E^0\|$	$\|\mathcal{K}\|$	d_{max}
40N50L20F_A	40	124	26	181	50
40N50L20F_B	40	123	35	172	50
40N50L50F_A	40	78	89	129	50
40N50L50F_B	40	72	71	147	50
40N50L80F_A	40	32	146	111	50
40N50L80F_B	40	35	154	106	50
50N50L20F_A	50	212	44	278	50
50N50L20F_B	50	235	59	256	50
50N50L50F_A	50	157	132	180	50
50N50L50F_B	50	132	117	219	50
50N50L80F_A	50	51	175	197	50
50N50L80F_B	50	58	212	171	50
60N50L20F_A	60	269	72	388	50
60N50L20F_B	60	268	63	397	50
60N50L50F_A	60	204	216	259	50
60N50L50F_B	60	200	197	276	50
60N50L80F_A	60	85	311	214	50
60N50L80F_B	60	74	283	261	50
80N50L20F_A	80	557	145	650	50
80N50L20F_B	80	545	124	665	50
80N50L50F_A	80	345	313	481	50
80N50L50F_B	80	375	366	476	50
80N50L80F_A	80	148	548	459	50
80N50L80F_B	80	121	536	428	50

Table 6.6: ARLP - p25 - Instances

For some instances we were not able to find an optimal solution. Table 6.7 shows the corresponding instances together with the highest lower (LB) and the smallest upper bound (UB) we obtained. For the respective instances the LP and optimality gaps have been computed using the upper bounds given in this table.

6.5.1 Results on NDPR Instances

The NDPR instances require only five or ten commodities to be connected. Thus, these instances are in general easier than the ARLP instances. However, it turned out that the instances with delays equivalent to the costs (Type I) are significantly more difficult than the instances with indirectly correlated costs (Type II).

Results for Type I Instances

The performance chart is given in Figure 6.1, and Tables 6.8 and 6.9 provide details on the LP and MILP computations respectively.

Instance	Instance set	LB	UB
60N10K30L	NDPR - Type I	658.59	692
80N5K30L		330.51	361
80N5K35L		283.29	344
80N10K30L		354.58	480
80N10K35L		305.15	502
160N5K30L		223.5	342
160N5K35L		212.08	311
160N10K30L		280.25	492
160N10K35L		0	INF
50N50L20F_A	ARLP	725.65	820
50N50L20F_B		557.72	656
60N50L20F_A		656.05	932
60N50L20F_B		727.17	9420
80N50L20F_A		561.74	15879
80N50L20F_B		345.27	17478
50N50L20F_A	ARLP - p25	760.34	815
50N50L20F_B		606.13	661
60N50L20F_A		709.77	1098
60N50L20F_B		778.3	1444
80N50L20F_A		596.18	16730
80N50L20F_B		374.48	16751

Table 6.7: Instances with unknown optimal solution

The obtained results indicate that the layered single commodity flow formulation worked best on this set of instances. Unfortunately, even this approach has not been able to solve the larger instances to optimality. However, many of the smaller instances have been solved fast. L_{CUTM} is slower but provides the best LP gap whenever the computation terminates within the time limit.

The models on multiple communication graphs implemented in SCIP solved six of the smaller instances. The implementations with reduced column generation in CPLEX found the optimal solution for an additional instance. Model CG_{CUTS} solved only a single instance optimally. We believe that this model is outperformed by the flow models due to the small number of commodities. Considering all commodity pairs at once weakens the lower bounds, but does not provide sufficient speed up to compete against the disaggregated flow based models.

Results for Type II Instances

The performance chart for this group of instances is shown in Figure 6.2 Results referring to the LP and MILP performance are provided in Table 6.10 and 6.11, respectively.

The results obtained on this set of NDPR instances are similar to the ones obtained on Type I instances. As already mentioned these instances have been easier to solve. The most successful

approach is again L_{SCFM}. This approach solves all instances to optimality within the time limit. The cut variant is again slower but outperformed all communication graph approaches this time. It has again the smallest LP gaps for all instances for which the LP computation terminated.

The communication graph approaches solved many of the smaller instances to optimality. Reduced column generation is again faster than the full Branch-and-Price implementation. Furthermore, the multi-commodity flow approach is faster than the single-commodity flow approach. The reason for this is that due to the low amount of commodities (we only have few $(1-2)$ targets per source). Hence, we do not gain much by aggregating per source, but loose on the quality of lower bounds. The communication graph model on a single graph exhibits the worst performance.

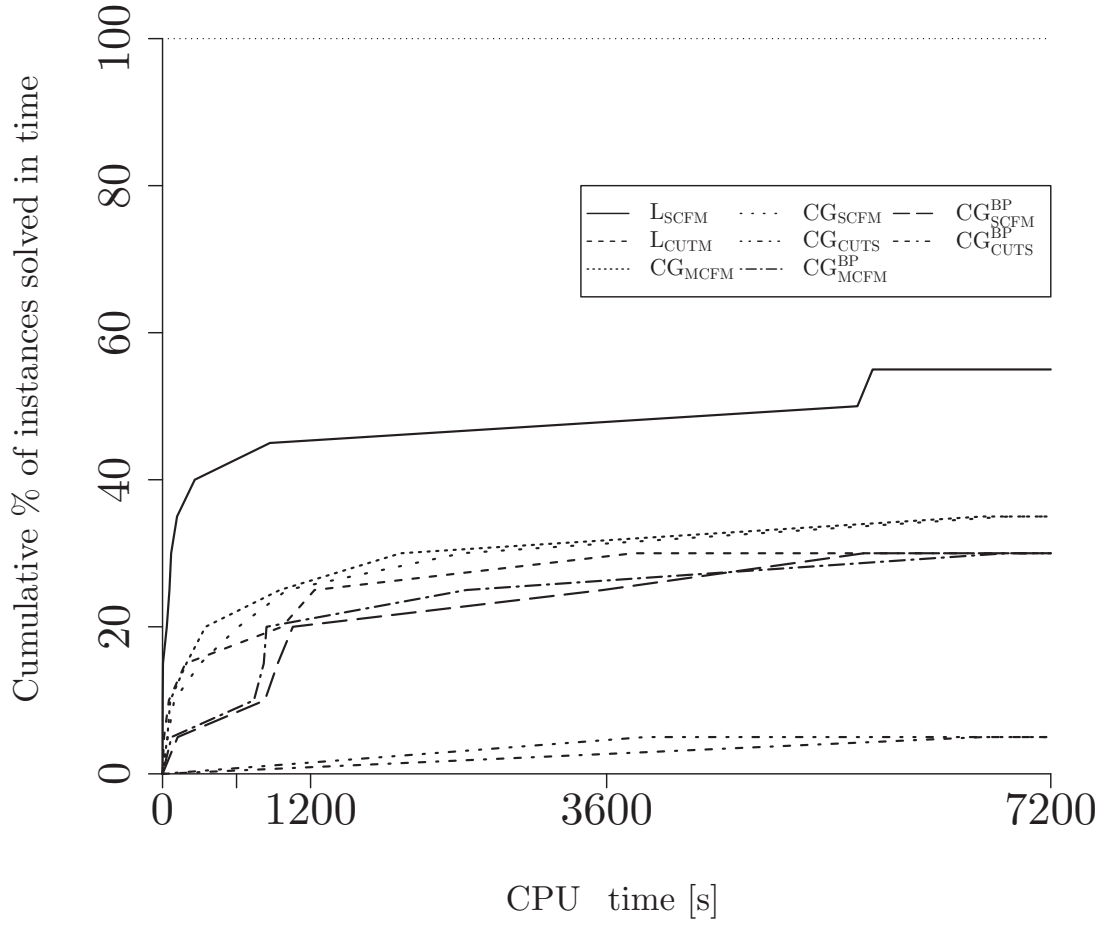


Figure 6.1: Performance Chart - NDPR - Type I instances

Instance	t - LP [s]								LP Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N5K30L	**0.25**	2.53	6.04	4.41	24.22	4.58	3.61	17.76	**12.99**	**12.99**	21.51	21.51	30.87	21.51	21.51	30.87
40N5K35L	**0.78**	39.89	24.53	22.05	128.83	19.25	22.02	86.43	**21.15**	**21.15**	38.47	38.47	51.79	38.47	38.47	52.29
40N10K30L	**1**	455.45	8.01	8.66	26.81	7.7	6.95	23.85	17.61	**15.44**	26.24	26.9	32.37	26.24	26.9	32.37
40N10K35L	**2.2**	890.09	39.78	45.21	149.99	36.68	30.87	130.48	22.12	**21.24**	40.12	40.33	48.92	40.12	40.33	50.15
50N5K30L	**0.31**	5.99	22.12	19.86	114.98	13.42	16.35	97.1	4.45	**1.85**	8.49	13.53	23.95	8.49	13.53	24.02
50N5K35L	**1.66**	57.27	99.98	82.14	274.55	99.54	73.72	274.49	2.35	**0.56**	22.49	25.39	37.5	22.49	25.39	37.5
50N10K30L	**3.14**	2205.63	55.14	54.29	319.74	50.83	55.26	220.25	14.99	**12.61**	24.62	25.16	31.4	24.62	25.16	33.16
50N10K35L	**7.03**	7200.02	310.01	244.11	1183.28	290.06	295.1	631.72	13.98	**11.42**	27.61	28.08	37.04	27.61	28.08	41.28
60N5K30L	**0.78**	38.5	31.14	29.44	233.59	22.1	23.87	199.21	**21.37**	**21.37**	28.12	28.12	35.17	28.12	28.12	35.26
60N5K35L	**2.67**	250.38	123.06	139.29	1272.33	111.89	99.83	1526.43	**10.47**	**10.47**	24.34	24.34	39.31	24.34	24.34	39.65
60N10K30L	**2.77**	806.18	62.27	70.71	320.73	59.16	74.8	189.54	**24.35**	**24.35**	32.26	32.26	41.26	32.26	32.26	43.12
60N10K35L	**10.32**	4837.53	280.06	277.69	1625.43	297.35	325.2	1131.73	**21.21**	**21.21**	33.05	33.05	45.93	33.05	33.05	52.32
80N5K30L	**6.07**	7200.07	1627.13	1844.29	7200.23	1612.85	1356.81	6351.86	24.72	**16.61**	43.66	50.16	INF	43.66	50.16	60.43
80N5K35L	**11.2**	7200.01	6662.88	6346.2	7199.88	7201.04	5095.59	7200.98	29.41	**26.7**	54.01	59.39	INF	INF	59.39	76.77
80N10K30L	**33.38**	7200.03	2860.39	2747.19	7199.31	3423.07	3091.36	5677.31	**32.22**	37.47	53.09	56.49	INF	53.09	56.49	65.08
80N10K35L	**72.94**	7200.04	7200.13	7199.94	7199.71	7201.02	7201.08	7200.83	**43.04**	56.94	INF	INF	INF	INF	INF	82.67
160N5K30L	**341.58**	7200.06	7202.24	7202.46	7201.85	ML	ML	ML	**35.96**	63.44	INF	INF	INF	INF	INF	INF
160N5K35L	**3186.92**	7200.07	7204.06	7204.13	7202.21	ML	ML	ML	**32.48**	69.27	INF	INF	INF	INF	INF	INF
160N10K30L	**2777.99**	7200.1	7202.47	7204.29	7201.34	ML	ML	ML	**43.19**	73.4	INF	INF	INF	INF	INF	INF
160N10K35L	7200.27	7200.16	7207.21	7210.19	7201.5	ML	ML	ML	100	**70.87**	INF	INF	INF	INF	INF	INF

Table 6.8: Comparing LP relaxations on NDPR - Type I instances

Instance	t [s]								Optimality-Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N5K30L	**3.61**	184.74	62.65	95	7199.5	740.97	933.39	7200.02	0	0	0	0	5.48	0	0	13.21
40N5K35L	**117.72**	7200.04	7199.2	7199.25	7199.08	7200.08	7200.08	7200.05	**0**	3.84	10.9	10.45	36.74	17.01	16.68	38.87
40N10K30L	**56.04**	7200.01	1930.8	2425.92	7199.12	7200.05	7200.04	7200.05	**0**	4.99	0	0	11.55	4.9	4.89	15.76
40N10K35L	**5632.78**	7200.02	7199.35	7199.25	7199.09	7200.05	7200.08	7200.07	**0**	23.98	18.45	18.82	36.87	21.57	22.18	43.77
50N5K30L	**0.75**	12.24	39.32	42.41	3912.02	73.02	122.98	6653.18	0	0	0	0	0	0	0	0
50N5K35L	**1.29**	51.63	192.93	305.97	7199.26	820.52	1056.22	7200.16	0	0	0	0	25.48	0	0	25.8
50N10K30L	**261.34**	3817.06	6645.6	6855.81	7199.45	6839.7	5685	7200.07	0	0	0	0	23.06	0	0	26.48
50N10K35L	**870.6**	7200.01	7199.22	7199.79	7199.24	7200.14	7200.19	7200.13	**0**	5.93	6.07	5.62	32.11	13.47	13.61	37.87
60N5K30L	**34.57**	968.74	348.46	615.27	7199.36	842.98	829.41	7200.11	0	0	0	0	21.6	0	0	21.86
60N5K35L	**69.77**	1232.45	950.76	1007.63	7199.43	2462.46	3570.6	7200.17	0	0	0	0	35.5	0	0	34.27
60N10K30L	7200.01	7200.02	7199.3	7199.17	7199.07	7200.12	7200.1	7200.08	**4.83**	20.21	5.82	6.67	32.04	9.24	9.75	34.91
60N10K35L	**5755.97**	7200.02	7199.5	7199.89	7199.97	7200.22	7200.21	7200.21	**0**	26.7	15.88	14.46	42.05	17.99	16.09	49.18
80N5K30L	7200.01	7200.02	7200.02	7199.52	7199.32	7200.49	7200.46	7200.45	**8.45**	22.81	37.38	44.87	INF	41.18	48.63	64.73
80N5K35L	7200.02	7200.05	7200.24	7200.02	7199.66	7201.16	7201.08	7201.14	**17.65**	28.91	53.94	59.17	INF	54.01	59.39	77.45
80N10K30L	7200.03	7200.03	7199.4	7199.7	7199.91	7200.53	7200.46	7200.47	**26.13**	39.05	52.99	53.56	INF	51.86	55.95	67.97
80N10K35L	7200.04	7200.04	7199.47	7199.61	7199.79	7201.13	7201.11	7201.13	**39.21**	57.98	INF	INF	INF	INF	INF	85.63
160N5K30L	7200.07	7200.06	7203.19	7202.71	7203.74	ML	ML	ML	**34.65**	63.76	INF	INF	INF	INF	INF	INF
160N5K35L	7200.15	7200.07	7210.81	7203.39	7206.56	ML	ML	ML	**31.81**	68.6	INF	INF	INF	INF	INF	INF
160N10K30L	7200.11	7200.13	7206.15	7203.73	7203.04	ML	ML	ML	**43.04**	71.97	INF	INF	INF	INF	INF	INF
160N10K35L	7200.2	7200.16	7206.48	7209.96	7204.88	ML	ML	ML	100	**69.84**	INF	INF	INF	INF	INF	INF

Table 6.9: Comparing MILP models on NDPR - Type I instances

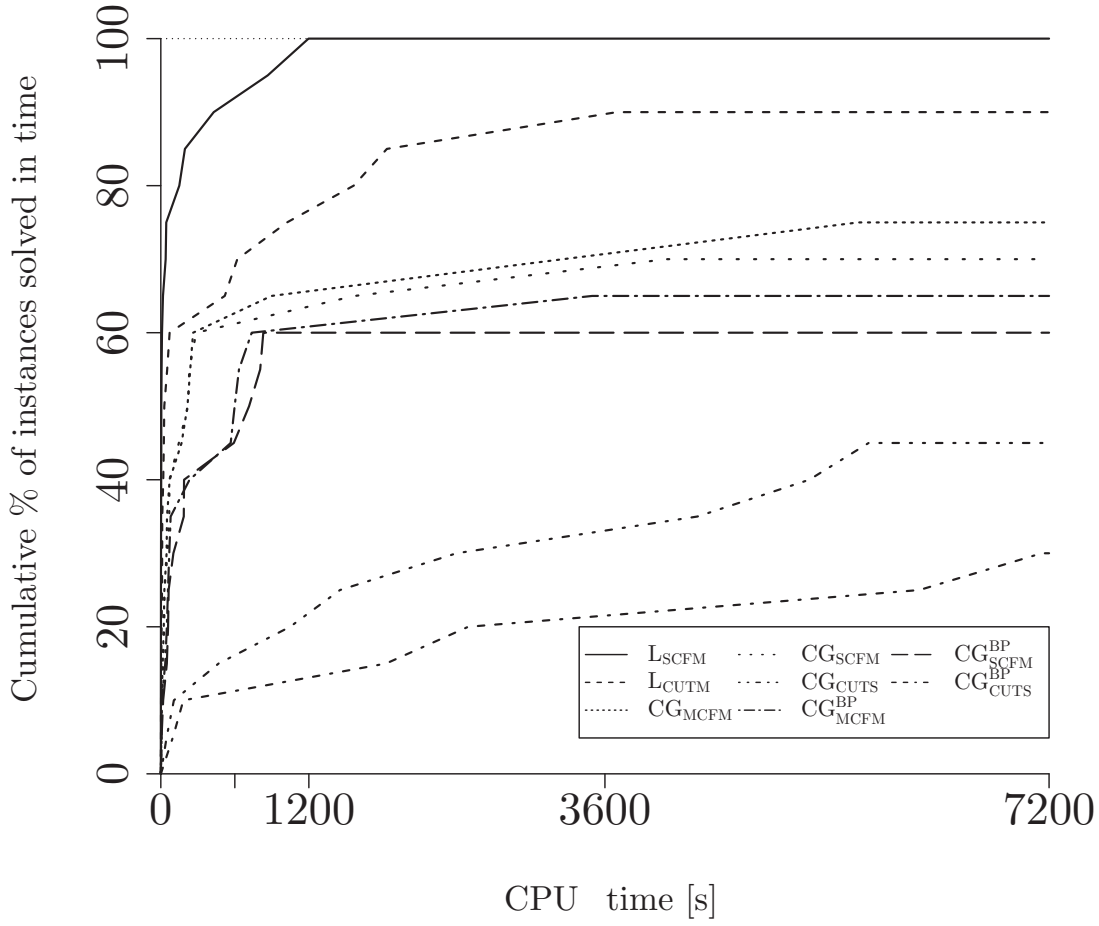


Figure 6.2: Performance Chart - NDPR - Type II instances

Instance	t - LP [s]								LP Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N5K30L	**0.08**	0.35	3.62	3.14	12.48	2.43	4.06	10.57	**0**	**0**	0.26	0.26	12.45	0.26	0.26	12.45
40N5K35L	**0.37**	0.48	19.48	16.87	60.17	14.06	18.01	55.61	**7.33**	**7.33**	7.76	7.76	24.14	7.76	7.76	24.14
40N10K30L	**0.61**	6.82	11.04	8.25	38.44	7.81	7.61	26.28	3.95	**1.79**	6.63	8	18.78	6.63	8	19.7
40N10K35L	**2.4**	19.93	57.83	39.87	168.91	57.24	34.05	146.07	8.28	**7.05**	10.06	10.89	27.96	10.06	10.89	30.38
50N5K30L	**0.18**	0.39	4.54	4.35	31.32	4.76	4.22	43.53	6.5	**0**	**0**	6.5	17.48	0	6.5	17.48
50N5K35L	**0.39**	2.15	46.64	41.24	772.42	59.9	38.15	580.56	9.75	**0.63**	1.89	9.84	27.36	1.89	9.84	27.36
50N10K30L	**1.27**	12.74	14.58	12.97	155.47	13.42	12.95	144.58	3.82	**0.35**	2.5	5.71	17.69	2.5	5.71	18.78
50N10K35L	**2.7**	20.64	156.39	108.73	3715.29	164.52	124.25	1077.27	0.7	**0**	2.97	3.97	20.68	2.97	3.97	30.85
60N5K30L	**0.5**	5.3	18.97	19.97	172.08	16.86	16.56	149.84	**13.73**	**13.73**	14.89	14.89	21.92	14.89	14.89	22.02
60N5K35L	**0.34**	2.31	53.62	58.83	143.57	44.58	37.76	193.34	0	0	0	0	21.1	0	0	21.1
60N10K30L	**3.23**	120.78	59.95	46.93	551.17	40.41	45.77	310.81	**10.97**	**10.97**	13.72	13.72	24.62	13.72	13.72	28.55
60N10K35L	**4.22**	233.37	149.26	133.54	1146.36	137.46	96.69	710.11	**1.69**	**1.69**	6.94	6.94	26.03	6.94	6.94	40.53
80N5K30L	**1.67**	21.81	572.22	436.13	7199.75	381.64	357.95	3463.27	13.83	**5.67**	7.85	15.93	INF	7.85	15.93	34.15
80N5K35L	**5.28**	145.94	2042.66	1661.43	7199.69	2074.14	1545.12	7201.14	13.64	**0.91**	6.16	16.14	INF	6.16	16.14	42.19
80N10K30L	**10.64**	849.66	991.46	1210.31	7200.24	1165.53	985.78	7200.58	11.8	**6.6**	12.1	16.14	INF	12.1	16.14	39.4
80N10K35L	**72.96**	7200.04	5700.94	6646.51	7200.4	5892.59	6450.45	7200.92	9.56	**5.59**	18.92	23.36	INF	18.92	23.36	53.25
160N5K30L	**331.1**	406.84	7201.36	7202.77	7203.14	ML	ML	ML	**5.95**	**5.95**	INF	INF	INF	INF	INF	INF
160N5K35L	**288.63**	590.5	7201.06	7201.2	7209.94	ML	ML	ML	**9.7**	**9.7**	INF	INF	INF	INF	INF	INF
160N10K30L	**1096.47**	7200.1	7203.47	7204.38	7203.26	ML	ML	ML	**4.73**	4.93	INF	INF	INF	INF	INF	INF
160N10K35L	**661.84**	7200.17	7210.86	7208.31	7206.54	ML	ML	ML	**4.17**	15	INF	INF	INF	INF	INF	INF

Table 6.10: Comparing LP relaxations on NDPR - Type II instances

Instance	t [s]								Optimality-Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N5K30L	**0.19**	1.28	3.81	3.21	47.36	7.28	6.16	98.61	0	0	0	0	0	0	0	0
40N5K35L	**0.99**	3.14	23.96	29.63	1047.92	79.69	58.8	2491.84	0	0	0	0	0	0	0	0
40N10K30L	**2.22**	11	15.42	19.7	468.18	58.07	101.93	1825.22	0	0	0	0	0	0	0	0
40N10K35L	**4.18**	19.7	168.31	146.22	5256.59	566.45	715.43	7200.06	0	0	0	0	0	0	0	12.08
50N5K30L	**0.38**	1.9	6.6	10.66	106.02	6.37	18.98	182.79	0	0	0	0	0	0	0	0
50N5K35L	**1.08**	6.52	50.04	69.81	5737.47	68.66	188.34	7200.16	0	0	0	0	0	0	0	19.27
50N10K30L	**1.56**	9.61	28.91	27.71	4350.26	35.84	48.87	7200.06	0	0	0	0	0	0	0	7.76
50N10K35L	**3.17**	28.21	261.68	230.62	7199.17	633.58	830.89	7200.16	0	0	0	0	15.21	0	0	19.26
60N5K30L	**3.98**	24.64	46.84	54.43	2397.31	232.74	186.21	7140.1	0	0	0	0	0	0	0	0
60N5K35L	**0.7**	11.65	71.92	68.66	1452.69	49	63.35	6148.97	0	0	0	0	0	0	0	0
60N10K30L	**44.19**	620.08	215.94	286.73	7199.46	598.58	805.56	7200.09	0	0	0	0	6.45	0	0	11.81
60N10K35L	**9.52**	518.23	238.36	220.78	7199.82	736.63	591.82	7200.16	0	0	0	0	16.54	0	0	18.12
80N5K30L	**6.32**	71.39	896.71	1570.59	7200.09	3496.7	7200.47	7200.45	0	0	0	0	INF	0	9.55	34.69
80N5K35L	**17.65**	51.75	3260.86	4103.22	7200.27	7200.8	7200.87	7200.91	0	0	0	0	INF	4.07	12.79	44.12
80N10K30L	**39.36**	1563.18	5652.88	7199.93	7199.6	7200.51	7200.51	7200.64	0	0	0	5.26	INF	10.75	13.55	36.56
80N10K35L	**149.99**	3697.08	7199.67	7199.7	7200.3	7201.04	7201.18	7201	0	0	18.65	23.33	INF	18.92	23.36	55.75
160N5K30L	**195.42**	1832.64	7204.17	7201.84	7202.56	ML	ML	ML	0	0	INF	INF	INF	INF	INF	INF
160N5K35L	**429.03**	1022.72	7202.83	7210.23	7213.09	ML	ML	ML	0	0	INF	INF	INF	INF	INF	INF
160N10K30L	**1197.28**	7200.08	7205.21	7204.06	7203.79	ML	ML	ML	**0**	41.13	INF	INF	INF	INF	INF	INF
160N10K35L	**866.65**	7200.14	7207.02	7204.66	7205.61	ML	ML	ML	**0**	18.92	INF	INF	INF	INF	INF	INF

Table 6.11: Comparing MILP models on NDPR - Type II instances

6.5.2 Results on ARLP Instances

These instances are in general much harder than the NDPR instances since they consider a much larger number of commodities. On the other hand they also include free edges, s.t. sometimes they become easier. Therefore, the most difficult instances are those with only 20% of free edges. We were only able to solve the smallest two instances of this type. The easiest instances are those containing 80% of free edges.

Approach L_{CUTM} did not solve any of these instances and also the LP computation did not terminate within the time limit. Thus, we have omitted this algorithm from the performance charts. The performance chart is given in Figure 6.3, and Tables 6.12 and 6.13 provide details on the LP and MILP computations respectively.

In contrast to the NDPR instances the models on communication graphs now have a better performance than the approaches on layered graphs. Furthermore, due to the fact that we have a larger number of targets per source the single-commodity flow approach on multiple communication graphs now has the best performance and solves more instances than any other approach. Moreover, the multi-commodity flow formulation on communication graphs exhibits the worst performance. This is caused by the high amount of variables induced by the large number of commodities. The approach on a single communication graph has been ineffective on the NDPR instances but works well for the ARLP instances. In most cases it is even faster than CG_{SCFM} but it solves one instance less. The better performance is again a consequence of the large number of commodities since having only one set of variables helps to reduce the model size in this case.

Model L_{SCFM} is not able to find the optimal solution in many cases but it has small LP gaps. This approach especially excels when dealing with instances with a low amount of free edges. The reason for this is that augmenting edges have less influence on the layered graph approaches since they only increase the amount of linking constraints. For the communication approaches however, the larger number of augmenting edges increases the amount of connections that require column generation, and this also makes column generation itself more difficult.

ARLP - p25

The performance chart for this group of instances is shown in Figure 6.4 Results referring to the LP and MILP performance are provided in Table 6.14 and 6.15, respectively.

Surprisingly, reducing the amount of commodities by 75% had only a small impact on the algorithmic performance of our approaches. Those that require a large number of variables depending on the number of commodity pairs could solve additional instances, but the faster approaches did not find optimal solutions for further instances. Moreover, in many cases the optimal solution remains the same as for the original instance.

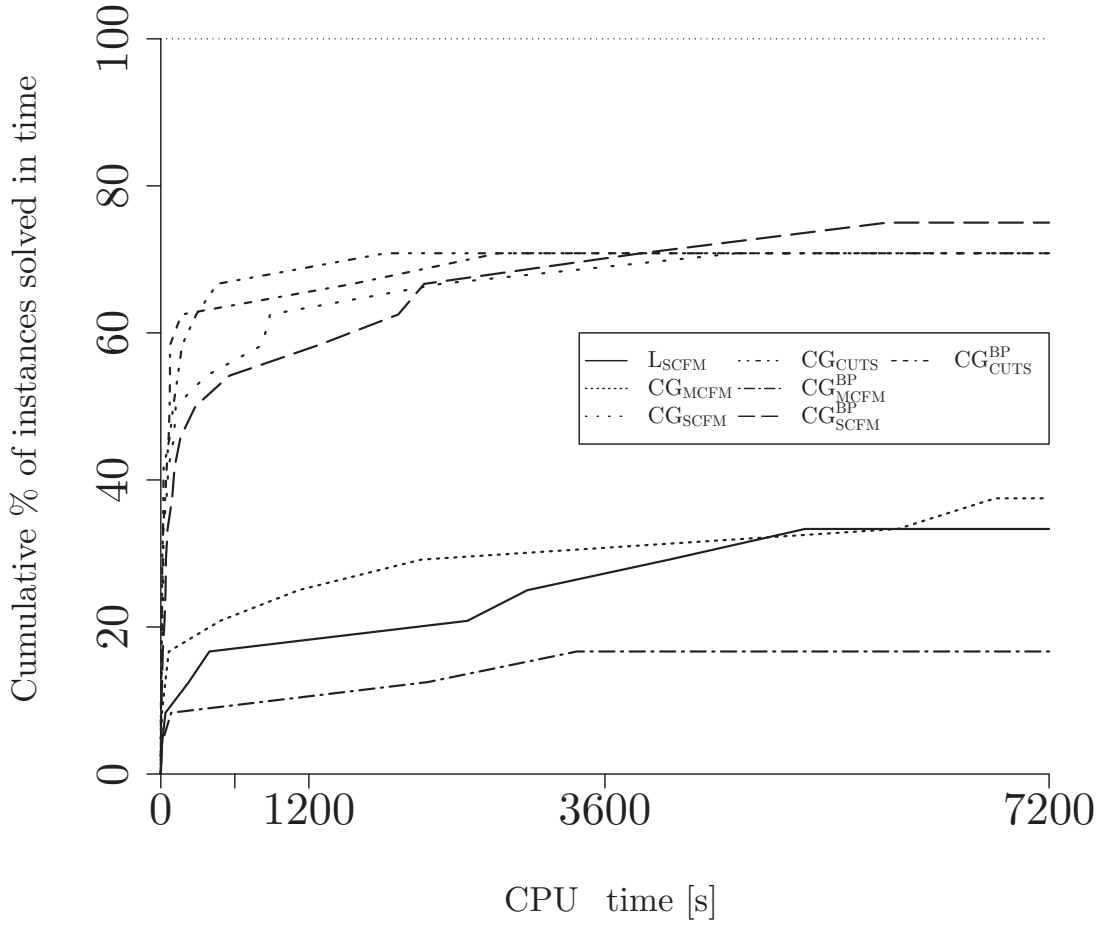


Figure 6.3: Performance Chart - ARLP instances

Instance	t - LP [s]								LP Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	CUT	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N50L20F_A	295.33	7200.37	7199.8	**431.73**	512.85	7201.51	353.41	269.15	23.27	71.66	INF	31.54	**16.46**	INF	31.54	22.96
40N50L20F_B	611.6	7200.16	7199.44	**247.38**	270.21	7201.45	207.96	386.89	**20.15**	63.72	INF	30.84	20.58	INF	30.84	23.89
40N50L50F_A	63.79	7200.18	283.77	**2.17**	4408.78	83.23	1.81	40.94	16.77	61.15	**15.59**	19.55	15.59	15.59	19.55	26.1
40N50L50F_B	81.16	7200.15	3994.72	**3.03**	427.59	618.35	2.75	381.58	9.54	43.34	**6.56**	13.24	**6.56**	6.56	13.24	6.56
40N50L80F_A	8.9	7200.17	2.22	**0.18**	40.08	3.66	0.27	651.26	0	35.19	0	0	0	0	0	0
40N50L80F_B	22.25	7200.2	18.45	**0.65**	845.73	9.92	0.9	575.35	8.09	49.08	**6.63**	8.73	**6.63**	6.63	8.73	7.26
50N50L20F_A	**1509.79**	7200.23	7201.25	6621.96	6261.55	ML	5082.6	3349.44	17.89	80.92	INF	26.78	**11.66**	INF	26.78	13.13
50N50L20F_B	**1739.22**	7200.65	7201.02	5960.35	7199.47	ML	4803.03	884.62	**16.27**	100	INF	26.07	INF	INF	26.07	32.9
50N50L50F_A	204.46	7200.21	7199.56	**27.41**	7199.19	ML	28.5	7200.04	**13.14**	90.73	INF	14.65	INF	INF	14.65	29.57
50N50L50F_B	152.83	7200.23	7199.84	**20.41**	1688.28	ML	20.1	192.69	7.42	72.21	INF	9.79	**6.13**	INF	9.79	24.32
50N50L80F_A	48.69	7200.18	7.75	**0.85**	7199.19	ML	0.93	406.32	14.05	82.54	**11.11**	14.08	INF	INF	14.08	23.71
50N50L80F_B	51.13	7200.18	7.19	**0.48**	1708.71	ML	0.9	7200.01	0.35	80.08	**0**	0.35	**0**	INF	0.35	3.15
60N50L20F_A	**7036.04**	7200.35	7202.21	7199.65	7199.44	ML	7200.38	769.9	**29.61**	88.51	INF	INF	INF	INF	INF	54.84
60N50L20F_B	**5641.07**	7200.35	7202.08	7199.67	7199.95	ML	7200.33	658.47	**92.67**	98.09	INF	INF	INF	INF	INF	96.36
60N50L50F_A	1627.48	7201.18	7199.44	**76.61**	7199.46	ML	77.44	379.41	**13.19**	100	INF	14.26	INF	INF	14.26	53.25
60N50L50F_B	957.36	7200.25	7199.83	**123.76**	7199.3	ML	123.28	1354.8	**18.28**	95.31	INF	20.44	INF	INF	20.44	24.53
60N50L80F_A	438	7200.36	215.76	**4.31**	7199.32	ML	4.84	7200.08	2.24	99.93	**2.09**	2.24	INF	INF	2.24	35.79
60N50L80F_B	662.32	7200.65	91.52	**3.48**	7199.54	ML	3.39	7200.03	12.09	86.95	**9.94**	12.09	INF	INF	12.09	18.07
80N50L20F_A	7200.84	7201.23	7206.46	7199.6	7199.32	ML	7201.41	4790.51	**100**	**100**	INF	INF	INF	INF	INF	97.95
80N50L20F_B	7200.63	7200.61	7206.73	7199.55	7199.41	ML	7201.44	7200.81	100	**99.66**	INF	INF	INF	INF	INF	98.35
80N50L50F_A	7200.75	7201.46	7200.22	**613.03**	7200.16	ML	570.33	7200.68	100	100	INF	**21.17**	INF	INF	21.17	72.19
80N50L50F_B	7200.84	7200.57	7200.66	**683.1**	7199.66	ML	701.76	7200.36	100	98.27	INF	**11.67**	INF	INF	11.67	31.01
80N50L80F_A	2042.42	7200.51	1893.38	**34.7**	7199.52	ML	40.68	7200.2	**1.86**	85.07	2.56	2.62	INF	INF	2.62	21.88
80N50L80F_B	727.49	7200.57	1369.11	**35.56**	7199.72	ML	44.11	7200.04	3.98	97.32	**3.91**	4.15	INF	INF	4.15	23.74

Table 6.12: Comparing LP relaxations on ARLP instances

Instance	t [s]								Optimality-Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N50L20F_A	7200.08	7200.16	7199.34	2268.63	**1820.83**	7202.05	3893.79	2736.41	15.96	38.19	INF	0	0.01	INF	0	0
40N50L20F_B	7200.18	CB	7199.76	885.76	**457.55**	7201.54	2135.6	1555.45	15.35	INF	INF	0	0	INF	0	0
40N50L50F_A	5221.32	CB	2110.84	19.78	**10.72**	2163.19	55.1	17.86	0	INF	0	0	0	0	0	0
40N50L50F_B	4121.8	CB	6762.84	13.63	11.63	3372.91	38.56	**10.54**	0	INF	-0.91	0	0	0	0	0
40N50L80F_A	9.97	7200.15	2.34	0.34	**0.16**	8.68	0.69	0.24	0	35.36	0	0	0	0	0	0
40N50L80F_B	226.7	7200.13	64.64	**3.24**	23.43	84.04	15.9	17.81	0	41.99	-1.42	-1.42	0	0	0	0
50N50L20F_A	7200.18	7200.21	7202.54	7199.47	7199.47	ML	7200.24	7200.16	16.61	70.48	INF	23.67	**11.51**	INF	26	11.52
50N50L20F_B	7200.24	CB	7202.25	7199.94	7199.27	ML	7200.24	7200.16	14.98	INF	INF	24.5	INF	INF	24.08	**13.89**
50N50L50F_A	7200.19	7200.27	7199.94	46.38	**18.15**	ML	111.11	35.45	8.33	88.56	INF	0	0	INF	0	0
50N50L50F_B	7200.16	7200.16	7199.38	107.5	100.71	ML	280.17	**71.65**	2.69	72.64	INF	0	0	INF	0	0
50N50L80F_A	393.7	7200.14	36.32	5.01	5.14	ML	10.06	**3.31**	0	81.03	0	0	0	INF	0	0
50N50L80F_B	37.93	7200.17	8.72	0.9	**0.61**	ML	2.81	0.75	0	71.93	0	0	0	INF	0	0
60N50L20F_A	7200.3	7200.36	7204.17	7199.75	7199.4	ML	7200.42	7200.38	**29.61**	88.37	INF	INF	INF	INF	INF	31.01
60N50L20F_B	7200.25	7200.75	7204.16	7199.51	7199.25	ML	7200.36	7200.26	92.67	98.48	INF	INF	INF	INF	INF	**92.28**
60N50L50F_A	7200.51	7200.26	7200.24	368.28	172.6	ML	553.87	**74.13**	10.03	92.14	INF	-3.34	0	INF	0	0
60N50L50F_B	7200.21	7200.27	7200.11	821.28	**72.53**	ML	1277.42	74.31	14.95	91.29	INF	-1.48	0	INF	0	0
60N50L80F_A	2970.53	7200.23	481.34	20.49	5.59	ML	41.39	**5.37**	0	99.54	0	0	0	INF	0	0
60N50L80F_B	2484.6	7200.22	1115.58	21.74	**5.4**	ML	30.02	9.61	0	84.46	0	0	0	INF	0	0
80N50L20F_A	7200.59	7200.13	7211.15	7200.11	7199.78	ML	7201.53	7201.25	100	100	INF	INF	INF	INF	INF	**96.46**
80N50L20F_B	7200.61	7200.13	7215.49	7199.73	7200.81	ML	7201.47	7201.81	100	100	INF	INF	INF	INF	INF	**98.02**
80N50L50F_A	7200.61	7200.14	7200.11	7199.53	7199.5	ML	**5888.25**	7200.59	100	100	INF	16.44	INF	INF	**0**	20.79
80N50L50F_B	7200.61	7200.17	7200.56	4668.89	287.24	ML	1924.26	**167.51**	100	100	INF	0	0	INF	0	0
80N50L80F_A	7200.47	7200.14	7200.11	61.9	**20.03**	ML	92.1	47.86	1.82	100	0.43	0	0	INF	0	0
80N50L80F_B	7200.39	7200.15	5973.68	120.72	137.93	ML	160.33	**63.54**	3.71	100	0	0	0	INF	0	0

Table 6.13: Comparing MILP models on ARLP instances

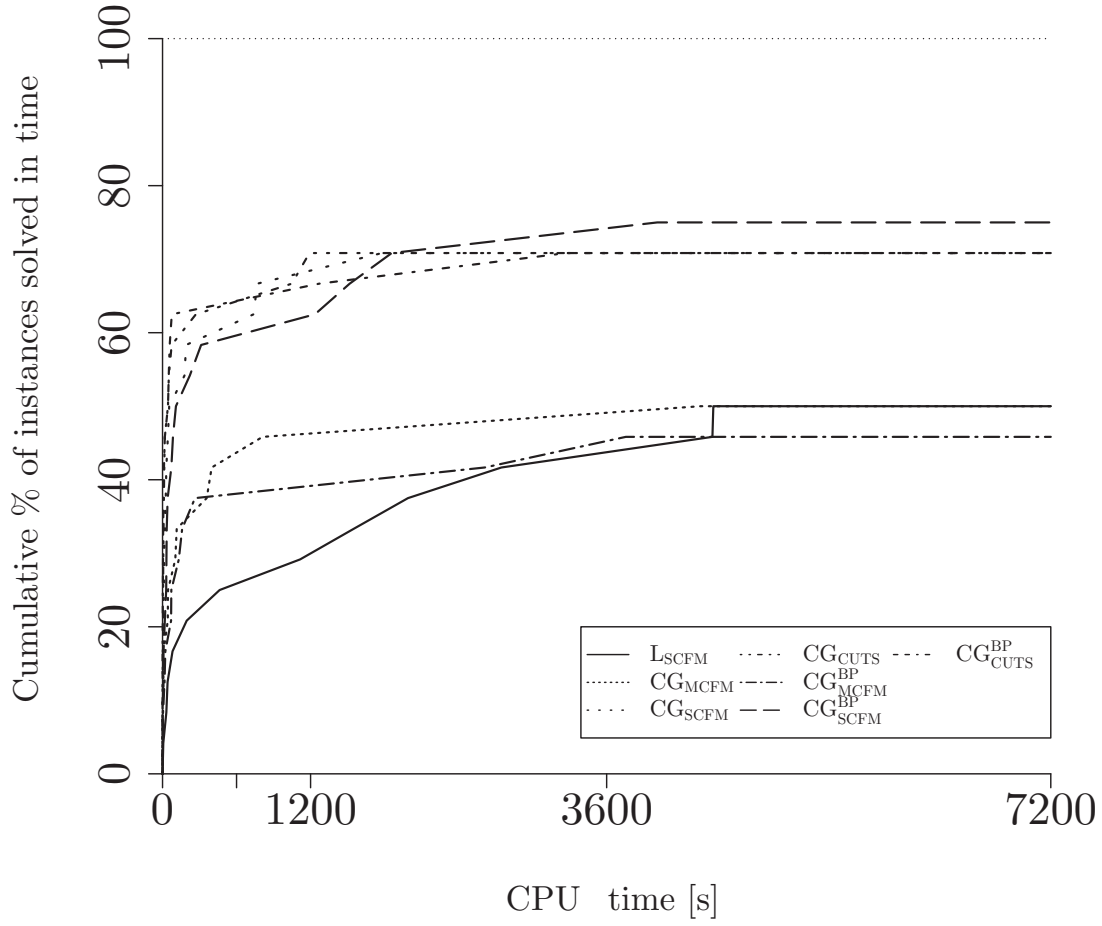


Figure 6.4: Performance Chart - ARLP - p25 instances

Instance	t - LP [s]								LP Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N50L20F_A	**348.09**	7200.09	7200.04	383.69	390.89	7200.32	430.53	80.94	**13.3**	30.95	INF	22.55	16.88	INF	22.55	38.75
40N50L20F_B	423.86	7200.15	7199.14	234.68	**193.51**	7200.26	243.48	33.07	**15.31**	35.04	INF	24.04	20.58	INF	24.04	43.9
40N50L50F_A	27.58	7200.11	5.55	**1.33**	1032.63	4.8	1.54	282.47	**13.5**	37.32	15.59	15.59	15.59	15.59	15.59	16.61
40N50L50F_B	70.96	7200.1	24.07	**2.77**	106.95	22.46	2.82	17.55	**6.53**	32.89	6.56	9.07	7.65	6.56	9.07	14.41
40N50L80F_A	6.78	7200.05	0.23	**0.22**	10.8	0.7	0.29	488.08	0	13.36	0	0	0	0	0	0
40N50L80F_B	10.69	7200.11	1.32	**0.31**	110.31	1.3	0.42	2.98	**6.24**	19.12	6.63	6.63	6.63	6.63	6.63	21.99
50N50L20F_A	**1198.82**	7200.19	7199.26	7199.19	1899.58	7201.17	4773.81	798.01	17.29	77.81	INF	INF	**11.18**	INF	26.72	13.61
50N50L20F_B	**1592.53**	7200.51	7199.42	7200.01	7199.75	7200.79	7200.22	2568.75	**8.74**	80.37	INF	INF	INF	INF	INF	28.56
50N50L50F_A	154.49	7200.2	830.71	**14.21**	7199.3	148.6	14.46	3677.03	10.12	62.66	**9.55**	12.42	INF	9.55	12.42	25.01
50N50L50F_B	67.96	7200.12	3642.31	**11.11**	2661.62	273.86	7.3	62.54	**5.03**	48.2	6.13	6.46	6.13	6.13	6.46	20.57
50N50L80F_A	25.58	7200.27	2.66	**0.48**	7199.33	2.97	0.63	37.73	11.36	65.25	**11.11**	11.52	INF	11.11	11.52	22
50N50L80F_B	42.77	7200.21	0.95	**0.51**	509.86	1.96	0.49	7200	0	20.41	0	0	0	0	0	0
60N50L20F_A	**4779.3**	7200.17	7200.48	7199.95	7199.69	7198.78	7200.39	6539.86	**35.36**	84.58	INF	INF	INF	INF	INF	44.49
60N50L20F_B	**3799.29**	7200.53	7200.63	7200.03	7199.11	7201.91	7200.34	163.09	**46.1**	88.02	INF	INF	INF	INF	INF	86.41
60N50L50F_A	1219.92	7200.53	7199.17	**69.99**	7199.45	1583	62.01	433.85	**7.6**	79.85	INF	8.17	INF	8.04	8.17	33.24
60N50L50F_B	684.12	7200.3	7199.57	**52.08**	7199.34	1968.79	59.59	495.78	**8.03**	72.75	INF	9.1	INF	8.5	9.1	29.33
60N50L80F_A	225.67	7200.24	14.79	**3.56**	7199.58	15.72	5.54	4805.19	**2.09**	66.67	**2.09**	**2.09**	INF	2.09	2.09	6.33
60N50L80F_B	331.13	7200.27	14.57	**1.96**	7199.34	15.95	2.43	112.09	11.32	74.2	**9.94**	11.32	INF	9.94	11.32	40.14
80N50L20F_A	7200.62	7200.48	7201.15	7200.1	7199.68	ML	7200.17	2057.28	100	**99.53**	INF	INF	INF	INF	INF	98.62
80N50L20F_B	7200.62	7200.55	7202.02	7199.46	7199.85	ML	7201.31	6573.59	100	**99.61**	INF	INF	INF	INF	INF	97.6
80N50L50F_A	6490.44	7200.49	7200.24	**574.68**	7199.34	ML	638.93	7200.17	**18.81**	93.11	INF	19.29	INF	INF	19.29	47.06
80N50L50F_B	7200.6	7200.55	7199.58	**386.9**	7199.63	ML	477.02	7200.21	100	92.76	INF	**8.89**	INF	INF	8.89	30.97
80N50L80F_A	1424.23	7200.5	148.74	**25.6**	7199.51	ML	26.28	7200	**1.82**	81.89	2.56	2.56	INF	INF	2.56	30.59
80N50L80F_B	418.31	7200.44	115.48	**11.93**	7199.42	ML	22.35	7200.02	**3.91**	90.75	**3.91**	**3.91**	INF	INF	3.91	39.88

Table 6.14: Comparing LP relaxations on ARLP - p25 instances

Instance	t [s]								Optimality-Gap [%]							
	L		CG			CG^{BP}			L		CG			CG^{BP}		
	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$	$SCFM$	$CUTM$	$MCFM$	$SCFM$	$CUTS$	$MCFM$	$SCFM$	$CUTS$
40N50L20F_A	7200.05	7200.06	7199.33	**767.79**	1047.65	7200.38	1858.02	3272.94	12.31	27.62	INF	-1.6	0	INF	0	0
40N50L20F_B	7200.09	CB	7199.47	**738.16**	1190.94	7200.3	1235.15	1269.38	7.42	27.62	INF	-0.11	0.01	INF	0	0
40N50L50F_A	462.92	CB	103.38	8.01	**5.61**	131.88	33.95	7.8	0	24.42	0	0	0	0	0	0
40N50L50F_B	195.89	CB	114.98	10.5	8.33	159.13	31.72	**3.9**	0	27.78	0	0	0	0	0	0
40N50L80F_A	6.21	7200.1	0.52	0.3	**0.07**	1.68	0.62	0.08	0	12.74	0	0	0	0	0	0
40N50L80F_B	40.89	7200.11	6.77	**1.51**	3.77	16.2	8.14	2.21	0	22.54	-1.42	-1.42	0	0	0	0
50N50L20F_A	7200.24	7200.18	7199.66	7199.45	7199.85	7201.15	7200.23	7200.07	15.43	60.12	INF	INF	**6.71**	INF	26.72	9.6
50N50L20F_B	7200.17	7200.27	7199.66	7199.29	7199.21	7201.04	7200.3	7200.12	8.3	74.67	INF	INF	INF	INF	INF	9.72
50N50L50F_A	4457.32	7200.15	810.93	30.78	18.17	263.13	41.53	**5.12**	0	50.11	0	0	0	0	0	0
50N50L50F_B	1989.47	7200.13	4359.43	**26.12**	51.16	2611.81	108.64	50.22	0	49.1	0	0	0	0	0	0
50N50L80F_A	81.02	7200.16	8.13	2.08	2.02	22.76	6.37	**0.7**	0	52.79	0	0	0	0	0	0
50N50L80F_B	30.85	7200.12	1.84	0.52	**0.14**	4.09	1.04	0.27	0	13.86	0	0	0	0	0	0
60N50L20F_A	7200.27	7200.25	7199.52	7200	7199.3	7202.29	7200.39	7200.17	**35.36**	86.59	INF	INF	INF	INF	INF	42.4
60N50L20F_B	7200.23	7200.34	7200.18	7199.93	7199.95	7202.3	7200.36	7200.16	**46.1**	86.03	INF	INF	INF	INF	INF	47.11
60N50L50F_A	7200.26	7200.41	7199.77	190.87	73.84	3755.41	220.94	**44.15**	5.5	68.95	INF	-2.23	0	0	0	0
60N50L50F_B	7200.24	7200.24	7199.21	177.88	**44.07**	7201.38	312.88	58.16	5.85	70.97	INF	-1.48	0	7.34	0	0
60N50L80F_A	1115.32	7200.21	42.58	19.83	4.97	71.16	29.55	**2.03**	0	45.49	0	0	0	0	0	0
60N50L80F_B	1550.46	7200.21	44.42	13.2	3.22	69.39	18.56	**1.68**	0	58.92	0	0	0	0	0	0
80N50L20F_A	7200.58	7200.14	7204.17	7199.55	7200.09	ML	7201.48	7200.79	100	100	INF	INF	INF	INF	INF	**96.44**
80N50L20F_B	7200.56	7200.13	7203.99	7199.35	7199.82	ML	7201.62	7201.03	100	100	INF	INF	INF	INF	INF	**97.76**
80N50L50F_A	7200.78	7200.32	7199.39	7199.18	7199.35	ML	**4010.51**	7200.17	18.31	93.47	INF	12.82	INF	INF	**0**	10.49
80N50L50F_B	7200.59	7200.15	7199.6	1778.93	270.56	ML	1516.73	**75.02**	100	100	6.04	0	0	INF	0	0
80N50L80F_A	2747.14	7200.48	398.45	43.38	**14**	ML	71.34	18.84	0	100	0	0	0	INF	0	0
80N50L80F_B	4464.64	7200.12	357.97	54	**10.93**	ML	84.54	11.68	0	100	0	0	0	INF	0	0

Table 6.15: Comparing MILP models on ARLP - p25 instances

CHAPTER 7

Conclusion

In this thesis we provided various MILP formulations for the solution of the NDPR. An extension of the NDPR with zero-cost edges has also been considered. We started by proving several conditions holding for optimal solutions to help stating the MILP models (cf. Chapter 2).

In Chapter 3 we presented the first set of models based on layered graphs. Our first model is based on a single graph and uses cut inequalities to ensure connectivity. Then we introduced further models based on multiple layered graphs, one for each source vertex. Two of these models are based on flows including a multi-commodity and a single commodity flow formulation. The third model uses cut-set inequalities.

In Chapter 4 we considered models based on communication graphs. Again we started with a cut model based on a single graph and then continued with models utilizing multiple graphs. To this end we stated two models using flow approaches and another model based on connectivity constraints as for the layered graphs.

In Chapter 5 we investigated required changes when acyclic solutions are required. We first showed that prohibiting cycles traversing an edge more than once in the same direction imposes no restriction concerning optimality. Next, we argued that our models on communication graphs are not well suited to enforce acyclic solutions. Then, we discussed the necessary adjustments to prevent cycles using selected models on layered graphs.

Finally, in Chapter 6 we provided details on our computational study. We presented the required cutting plane and column generation algorithms to deal with exponential amounts of constraints and/or variables. Furthermore, we discussed the settings used for the MILP solvers. Then, we introduced modified versions of instances from the previous literature and two sets of entirely new instances. We solved the one set of modified instances to optimality using one of our layered graph approaches. The other set of modified instances turned out to be more challenging but we still found optimal solutions for 11 out of 20 instances. Regarding the two sets of new instances we found in each set optimal solutions for 18 out of 24 instances. It turned out that for these instances that compose a large number of commodity pairs, models on communication graphs perform better in practice

7.1 Future work

In the first chapter we argued that the NDPR generalizes many other combinatorial optimization problems, such as the RLP or the MCPPR. In terms of usability of our algorithms this means that the proposed models and algorithmic techniques can also be applies to all simplifications of the NDPR. However, there is a trade off between the generality of our models and their performance in particular cases. We think that considering specific cases for certain parameters might help to develop faster algorithms. One option would be to consider a rooted case for which a single node needs to communicate with all other nodes but the remaining nodes do not require to communicate directly, i.e., we consider set $\mathcal{K} = \{(r, i) | i \in V, i \neq r\}$ for some root $r \in V$. Another case would be to consider only sets $\mathcal{K}$ containing all node pairs as has been done for the RLP. In general a high number of commodities makes the problem difficult but knowing more about the structure might allow additional modeling approaches.

Finally, note that many of our optional constraints require Big-M constants. Most of them can be disaggregated to impose stronger bounds. We decided to use aggregated variants since this helps to reduce the overall number of constraints. Nevertheless, it might be interesting to also consider disaggregation, either as replacement or as additional constraints. Dynamic separation would also be an option.

APPENDIX A

Acronyms

NDPR Network Design Problem with Relays

MCPPR Minimum Cost Path Problem with Relays

RLP Regenerator Location Problem

GRLP Generalized Regenerator Location Problem

RPP Regenerator Placement Problem

WCSPP Weight Constrained Shortest Path Problem

MLSTP Maximum Leaf Spanning Tree Problem

MCDSP Minimum Connected Dominating Set Problem

LP Linear Programming

ILP Integer Linear Programming

MILP Mixed Integer Linear Programming

Bibliography

[1] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[3] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[4] E. A. Cabral, E. Erkut, G. Laporte, and R. A. Patterson. The network design problem with relays. *European Journal of Operational Research*, 180(2):834–844, 2007.

[5] S. Chen, I. Ljubić, and S. Raghavan. The generalized regenerator location problem. In *Proceedings of the International Network Optimization Conference (INOC 2009)*, 2009.

[6] S. Chen, I. Ljubić, and S. Raghavan. The regenerator location problem. *Networks*, 55(3): 205–220, 2010.

[7] S. Chen, I. Ljubić, and S. Raghavan. The generalized regenerator location problem. *INFORMS Journal on Computing*, 2014. to appear.

[8] B. V. Cherkassy and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. In *Proceedings of the 4th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 157–171, London, UK, UK, 1995. Springer-Verlag.

[9] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[10] M. Flammini, A. Marchetti-Spaccamela, G. Monaco, L. Moscardelli, and S. Zaks. On the complexity of the regenerator placement problem in optical networks. *IEEE/ACM Trans. Netw.*, 19(2):498–511, 2011.

[11] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345–, 1962.

[12] T. Fujie. The maximum-leaf spanning tree problem: Formulations and facets. *Networks*, 43(4):212–223, 2004.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[14] B. Gendron, A. Lucena, A. S. da Cunha, and L. Simonetti. Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. Technical report.

[15] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, 9(6):849–859, 1961.

[16] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem–Part II. *Operations Research*, 11(6):863–888, 1963.

[17] M. T. Godinho, L. Gouveia, and Magnanti T. L. Combined route capacity and route length models for unit demand vehicle routing problems. *Discrete Optimization*, 5(2):350 – 372, 2008. In Memory of George B. Dantzig.

[18] M. T. Godinho, L. Gouveia, and P. Pesneau. On a Time-Dependent Formulation and an Updated Classification of ATSP Formulations. In A. R. Mahjoub, editor, *Progress in Combinatorial Optimization*, pages 223–254. ISTE-Wiley, 2011.

[19] M. T. Godinho, L. Gouveia, and P. Pesneau. Natural and extended formulations for the time-dependent traveling salesman problem. *Discrete Applied Mathematics*, 164:138–153, 2014.

[20] L. Gouveia, A. Paias, and D. Sharma. Modeling and solving the rooted distance-constrained minimum spanning tree problem. *Computers and Operations Research*, 35 (2):600–613, 2008.

[21] L. Gouveia, L. Simonetti, and E. Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. *Mathematical Programming*, 128(1-2):123–148, 2011.

[22] L. Gouveia, M. Leitner, and I. Ljubić. Hop constrained steiner trees with multiple root nodes. *European Journal of Operational Research*, 236(1):100 – 112, 2014.

[23] IBM. IBM ILOG CPLEX: High-performance software for mathematical programming and optimization. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`. accessed 2014-06-17.

[24] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977.

[25] A. Konak. Network design problem with relays: A genetic algorithm with a path-based crossover and a set covering formulation. *European Journal of Operational Research*, 218 (3):829–837, 2012.

[26] S. Kulturel-Konak and A. Konak. A local search hybrid genetic algorithm approach to the network design problem with relay stations. In S. Raghavan, B. Golden, and E. Wasil, editors, *Telecommunications Modeling, Policy, and Technology*, volume 44 of *Operations Research/Computer Science Interfaces*, pages 311–324. Springer US, 2008. ISBN 978-0-387-77779-5.

[27] G. Laporte and M. M. B. Pascoal. Minimum cost path problems with relays. *Computers & Operations Research*, 38(1):165–173, 2011. ISSN 0305-0548.

[28] I. Ljubić and S. Gollowitzer. Layered graph approaches to the hop constrained connected facility location problem. *INFORMS Journal on Computing*, 25(2):256–270, 2013.

[29] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[30] A. Lucena, N. Maculan, and L. Simonetti. Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Science*, 7(3):289–311, 2010.

[31] B. Mukherjee. WDM optical communication networks: progress and challenges. *Selected Areas in Communications, IEEE Journal on*, 18(10):1810–1824, 2000.

[32] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[33] J. C. Picard and M. Queyranne. The Time-dependent Traveling Salesman Problem and its Application to the Tardiness Problem in One-machine Scheduling. *Operations Research*, 26(1):86–110, 1978.

[34] M. Ruthmair. *On Solving Constrained Tree Problems and an Adaptive Layers Framework*. PhD thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Vienna, Austria, 2012.

[35] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[36] A. Sen, S. Banerjee, P. Ghosh, S. Murthy, and H. Ngo. Brief announcement: On regenerator placement problems in optical networks. In *Proceedings of the Twenty-second Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '10, pages 178–180, New York, NY, USA, 2010. ACM.