Engineering Combinatorial Optimization Algorithms to Improve the Lifetime of OLED Displays

Dissertation

zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

von

Andreas Karrenbauer

Saarbrücken
2007

Datum des Kolloquiums: 22. Dezember 2007
Dekan der Naturwissenschaftlich-Technischen Fakultät I:
Prof. Dr. Thorsten Herfet
Prüfungsausschuss:
Prof. Dr. Gerhard Weikum (Vorsitzender)
Prof. Dr. Kurt Mehlhorn (Berichterstattender)
Prof. Dr. Friedrich Eisenbrand (Berichterstattender)
Prof. Dr. Chihao Xu (Berichterstattender)
Dr. Ernst Althaus

Zusammenfassung

In dieser Arbeit betrachten wir ein Optimierungsproblem, das im Zusammenhang mit der Entwicklung von Steuergeräten für OLED Bildschirme auftritt. Unser Ziel ist es, die Amplitude der Stromstärke, die einen direkten Einfluss auf die Lebensdauer eines solchen Bildschirms hat, zu minimieren. Dazu modellieren wir das Problem als ein ganzzahliges lineares Programm. Danach verfeinern wir diese Formulierung, indem wir die kombinatorische Struktur des Problems ausnutzen. Wir erstellen ein allgemeines Approximationssystem basierend auf Netzwerkflusstechniken. Wir wenden einen *algorithm engineering* Ansatz an, um unsere Heuristiken durch wechselwirkende experimentelle Auswertungen und theoretische Untersuchungen zu verbessern. Zum Beispiel zeigen wir auf der theoretischen Seite einen vollkombinatorischen Linearzeitalgorithmus für das MAXFLOW Problem auf Graphen mit beschränkter Bandbreite und entsprechend nummerierten Knoten. In Bezug auf unsere Anwendung implementieren wir eine maßgeschneiderte Version dieses Algorithmus, um ein wichtiges Teilproblem zu lösen. Letztendlich haben wir einen vollkombinatorischen Approximationsalgorithmus entwickelt, der gut geeignet ist, um als Schaltkreis in ein Steuergerät für einen OLED Bildschirm integriert zu werden. Darüber hinaus erreichen unsere Algorithmen nahezu optimale Lösungen in der Praxis, was zu einer deutlich längeren Lebensdauer eines OLED Bildschirms mit passiver Matrix führt.

Abstract

In this thesis, we consider an optimization problem arising in the design of controllers for OLED displays. Our objective is to minimize amplitude of the electrical current through the diodes which has a direct impact on the lifetime of such a display. To this end, we model this problem as an integer linear program. Subsequently, we refine this formulation by exploiting the combinatorial structure of the problem. We establish a general approximation framework based on network flow techniques. We apply an algorithm engineering approach to improve our heuristics by interacting experimental evaluations and theoretical investigations. For example on the theoretical side, we present a fully combinatorial linear-time algorithm for the MAXFLOW problem on graphs with bounded bandwidth and accordingly numbered nodes. With respect to our application, we implement a custom-made version of this algorithm to solve an important subproblem. Finally, we develop fully combinatorial approximation algorithms well suited for being implemented in the hardware of a control device that drives an OLED display. Moreover, our algorithms achieve near optimal solutions in practice yielding significantly longer lifetimes of passive matrix OLED displays.

Acknowledgments

I thank Prof. Dr. Friedrich Eisenbrand for giving me the opportunity to work in his former Discrete Optimization group at the Max-Planck-Institute for Computer Science. He roused my interest in integer programming and combinatorial optimization. His guidance, advice, and encouragement have been invaluable throughout my master's and doctoral studies.

I am grateful to Prof. Dr. Kurt Mehlhorn for his advice with respect to all aspects of research and for his commitment to let me finish my thesis in his group when Friedrich Eisenbrand accepted a professorship at Dortmund.

I was most fortunate to meet Prof. Dr. Chihao Xu who introduced me to the field of OLED displays. We have had many fruitful discussions about the technical background and the requirements on the algorithms with respect to the real-world application. Moreover, I thank the members of his group at the Electrical Engineering Department of Saarland University. In particular, I am grateful to Cosmin Codrea and Kian Min Soh who worked long hours to provide nice pictures of a prototype display.

During the last years, I have greatly enjoyed the warm atmosphere at the Max-Planck-Institute for Computer Science. In particular, I am grateful to my former officemate Markus Behle for the very pleasant time on the 4th floor. I thank Pascal Schweitzer and Alantha Newman for reading parts of this manuscript and providing very helpful comments to improve the presentation of this thesis. I am grateful to Tobias Jung for performing lots of experiments on massive testsets as part of his bachelor's thesis, and I thank Markus Tetzlaff for providing a voluminous testset of digital photographs. For support in LaTeX and some formatting tips, I am grateful to Arno Eigenwillig and Tobias Friedrich.

Contents

Chapter 1

Introduction

1.1 Motivation

Organic **L**ight **E**mitting **D**iodes (**OLED**s) have received growing attention recently as more and more commercial products are equipped with such displays. Though they have many advantages over current technology, such as Liquid Chrystal Display (LCD) [1], only small-sized OLED displays have entered the mass market.

OLED displays are considered to be the displays of the future. The image and video displayed has a very high contrast and a viewing angle of nearly 180 degrees. It reacts within 10 microseconds, which is much faster than the human eye can catch and is therefore well suited for video applications. Moreover, the display is physically flexible.

There are two different OLED technologies called *active matrix* (AM) and *passive matrix* (PM). The former is more expensive but offers a longer lifetime than the latter. Their limited lifetime is one major reason why there are only small-sized displays on the mass market. For mobile phones or digital cameras, large state of the art OLED displays are either too expensive or suffer from insufficient lifetime.

What causes this short lifetime? Briefly, the reason is the high electrical current flowing through the diodes that occurs with the traditional addressing techniques. Though it seems that every pixel shines continuously with a certain brightness, rows of pixels are continually lit up one after another. This works at a sufficiently high frame rate since the perception of the human eye is the average intensity emitted by each diode. The problem is that this row-by-row activation scheme causes long idle times of the diodes and extreme stress upon activation.

While a lot of research is conducted on the material science side, a joint project of the Max Planck Institute for Computer Science and the Electrical Engineering Department of Saarland University was started in mid-2005 to tackle this problem from the display driver point of view. As a first result of this cooperation, we filed a patent [2] describing the **C**onsecutive **M**ulti**L**ine **A**ddressing

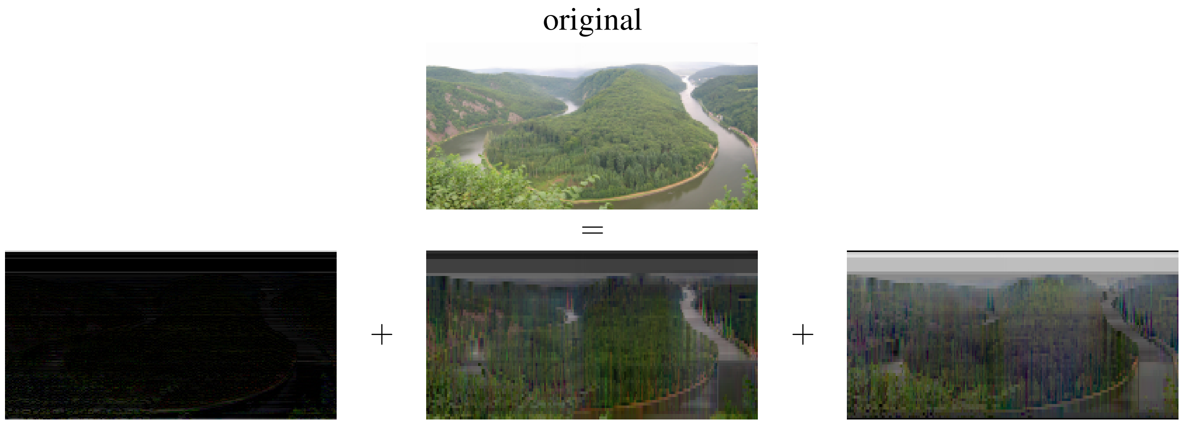


Figure 1.1: Decomposition of an image for CMLA2 such that every two rows of the double parts (the two images on the right) have the same content and the sum of all row maxima is minimized. The singleline part is the almost completely dark image on the left.

(**CMLA**) scheme. It is a method to drive two or more consecutive rows simultaneously to lower the stress on the diodes. We will call the number k of rows that we maximally drive simultaneously the *multiline order* in the following. Occassionally, we will denote the CMLA driving scheme of a particular order k by CMLAk, e.g. CMLA2 for doubleline addressing. The higher the multiline order, the more we can potentially reduce the amplitude of the electrical current and, hence, increase the lifetime of passive matrix devices.

For such displays rows can only be simultaneously displayed if their content is equal. Therefore the goal is to decompose an image into several subframes, the overlay (addition) of which is equal to the original image. Figure 1.1 shows such a decomposition. The first image (single) is traditionally displayed row-after-row. In the other two images (double) every two rows have the same content so that one can display these rows simultaneously. We thereby reduce the maximum intensities of the rows which directly relate to the necessary amplitude of the electrical current. Intuitively, the images on the bottom of Figure 1.1 are much darker than the original one. The decomposition should be performed in such a way that the amplitude of the electrical current needed to display the picture is as small as possible. This involves an optimization problem that must to be solved or approximated in real-time on a chip that drives such a device.

In this thesis, we engineer an approximation algorithm that is currently being implemented in hardware to actually drive such an OLED display (see Figure 1.2 for a demonstrator). This imposes some restrictions on the methods and techniques we shall use. First of all, such an algorithm has to compute a feasible solution in real-time, i.e. faster than the perception of a human eye. Moreover, it should be implemented on a chip of low cost meaning that for example we are not able to use a general purpose LP solver or a general purpose CPU with an IEEE floating point unit. We therefore look for algorithms that are sufficiently simple and easy to implement. Moreover, the algorithms should not suffer from numerical instabilities. That is, we want to compute a lossless decomposition. But exact rational arithmetic is not an option for such a real-time application. Rather, we want to use only fixed-precision number types, i.e. integers of

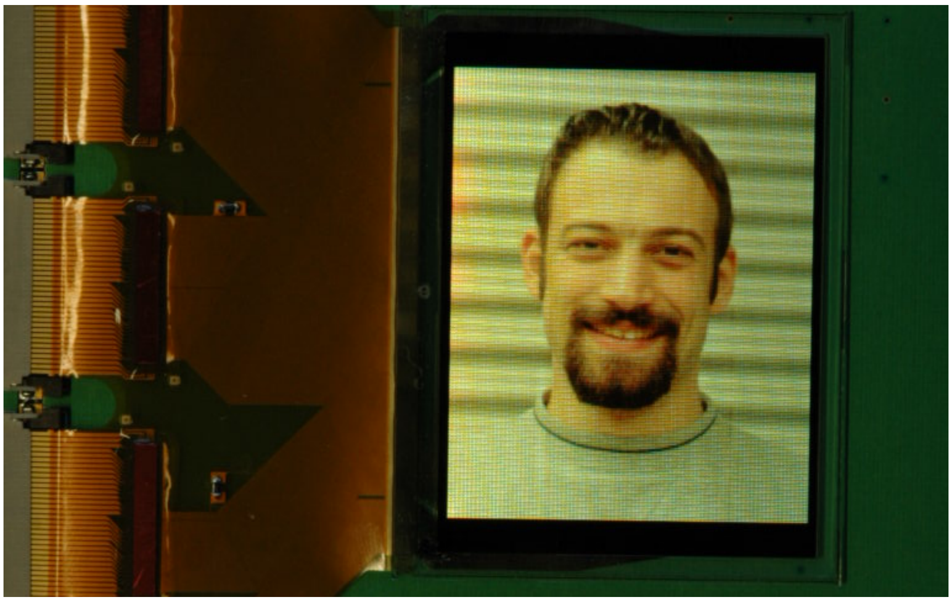


Figure 1.2: A demo board with an OLED display to test CMLA. The image is displayed using CMLA2.

a fixed size. Hence, we aim at a fully combinatorial algorithm using only addition, subtraction, and comparison.

Though this particular optimization problem with limited computational resources motivates the topic of this thesis, the ramifications of this work could be quite broad. We demonstrate the power of combinatorial optimization techniques to handle such real-world problems. We also showcase the algorithm engineering process consisting of the refinement of the mathematical model by an interaction of theoretical investigations and practical evaluations until an algorithm is found that matches all design goals. In addition, we develop a fully combinatorial linear-time algorithm for the TRANSSHIPMENT problem for the special graphs from our application that may be of independent interest as it is also applicable to arbitrary graphs with bounded bandwidth and accordingly numbered nodes.

1.2 Related Work

Independently, the British company *Cambridge Display Technologies* (CDT), a pioneer in polymer OLED substrates, filed a patent in 2005 [3] concerning the addressing of multiple lines. They later named their technique *Total Matrix Addressing* (TMA) [4]. However, their approach is both technically as well as mathematically different from CMLA.

Because TMA is based on *Non-negative Matrix Factorization* [5, 6], there is a clear border between their approach and our design goals. They use an iterative method to approximate the original image by a product of two non-negative matrices, whereas our design goals target a lossless decomposition into a sum of consecutive multiline matrices computed by a fully combinatorial algorithm. We want to achieve a cheap hardware implementation directly on the driver chip of an OLED display, which is unlikely for CDT's decomposition.

1.3 Outline

The remainder of this thesis is organized as follows. In Chapter 2, we briefly discuss the technical background of OLEDs, passive matrix display driving, and CMLA. In particular, we explain how this method reduces the necessary amplitude of the electrical current. Though understanding this technical background is not a must for the rest of this thesis, it might be helpful to be familiar with to the design decisions. Additionally, we formulate the design goals at the end of that chapter.

The theoretical foundations of CMLA follow in Chapter 3. That is, we first model the decomposition problem in mathematical terms. Based on this, we derive an integer linear programming formulation. Afterwards, we show that it is equivalent to a certain network-flow problem on a special graph.

We examine this class of graphs more closely and we show some structural properties that will become useful to develop efficient custom-made algorithms for our purpose. In particular, we show that there is a fully combinatorial algorithm for the TRANSSHIPMENT problem on graphs with bounded bandwidth and accordingly numbered nodes running in linear time.

To analyze the complexity of the CMLA problem, we reduce it to an equivalent formulation as a covering (integer) linear program. We are thus able to relate the decomposition problem for monochrome images to the Directed Steiner Network problem. Depending on the assumptions on the graph, we show some cases where the problem is NP-complete and some cases that are polynomial-time solveable. Moreover, this new formulation is helpful in practice as it is the basis of our first fully combinatorial heuristics and the subsequent improvements towards a competitive linear-time approximation algorithm for general images. To this end, we identify a subset of the constraints that permit an exact combinatorial solution in linear time. Furthermore, we prove the existence of a polynomial-time approximation scheme (PTAS) for fixed multiline order. This result is not only be of purely theoretical interest, since the idea of the proof is also used in a practical implementation.

We conclude Chapter 3 by considering the canonical online problem related to CMLA. We give tight upper and lower bounds on the competitive ratio in the special case of doubleline addressing of monochrome images. In particular, we develop a set of rules that we incorporate into our practical algorithm to improve its average per-instance approximation ratio to less than 1% in practice.

In Chapter 4, we describe the engineering process. That is, we present the evaluations of the methods that we have derived in the different stages. Moreover, we describe the interaction with the theory part. In the first section of this chapter, we consider the first generic formulation as a linear program and the network design formulation of the doubleline case. We evaluate them with the commercial solver CPLEX (see `www.ilog.com`). It follows the presentation of the separation and augmentation approach, which is based on the iterative combinatorial separation of the exponentially many inequalities of the network design formulation by a MAXFLOW/MINCUT computation. Afterwards, we consider the special case of Doubleline Addressing. We extend the previous approach and significantly cut its running time by using the ideas of the theory chapter to remove the bottleneck that we have identified in the preceding experiments. That is, we introduce a competitive algorithm that approximates the solution of the network-design formulation by a single pass over all arcs and subsequently computes the decomposition using the fully combinatorial linear-time algorithm for the TRANSSHIPMENT problem on graphs with bounded bandwidth. Furthermore, we present two methods to make this algorithm memory-aware. We conclude this chapter by presenting generalizations to higher multiline orders.

1.4 Sources

The basics of CMLA have been published in the patent application [2]. There have been two subsequent publications in the proceedings of the *European Symposium on Algorithms* [7] in 2006 and in the proceedings of the *Workshop on Experimental Algorithms* [8] in 2007, where the former includes the network model, the linear time separation, and the segmentation and augmentation approach, and the latter includes a polynomial time algorithm for doubleline addressing of monochrome images, the treatment of the online problem, and the evaluation of the competitive algorithm for the doubleline case. Both papers have been invited to the corresponding special issues of best papers and are to appear in *Algorithmica* and the *ACM Journal of Experimental Algorithmics*, respectively. A further paper focusing more on the technical background has been published at the *Society for Information Display 2007 International Symposium* [9].

Chapter 2

Technical Background

2.1 OLED Displays

To understand the objective of our optimization problem, we need to explain in an informal way how OLED displays work. Organic Light Emitting Diodes consist of a double layer of organic thin films embedded between a transparent anode and a metal cathode (see Figure 2.1).

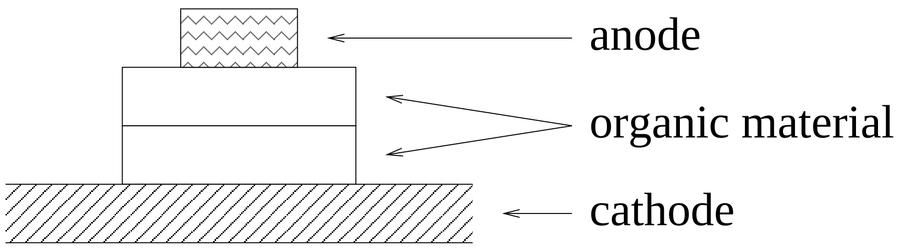


Figure 2.1: Organic Light Emitting Diode

If an electrical voltage between the anode and the cathode is applied, electrons and holes are injected at cathode and anode, respectively. They move towards each other through the organic layers. With a certain probability, an electron and a hole recombine and emit a photon. Hence, the rate of emitted photons is proportional to the electrical current I. Moreover, the light emission happens without remarkable delays. Thus, the number of emitted photons is also proportional to the time for which the electrical current flows through the diode.

By choosing the appropriate organic material, the wavelength of the emitted photon is in the range of the red, green, or blue spectrum. This phenomenon is called electroluminescence. The interested reader is pointed to [10, 11, 12] for further details.

In this thesis, we consider *passive matrix* devices. That is, a display is made up of a grid of diodes such that the cathodes and anodes form vertical and horizontal traces, respectively. So the diodes are arranged in a matrix structure with n rows and m columns. At any crossover between a row and a column in the third dimension there is a diode which works as a pixel, see Figure 2.2. In the case of a color display, each diode corresponds to a red, green, or blue subpixel (RGB) such that the diodes of one column are of one color and the diodes in one row follow the pattern $RGBRGB\cdots$. Since the color of an diode is not mathematically relevant, we will not distinguish between pixel and subpixel, i.e. grayscale and color images.

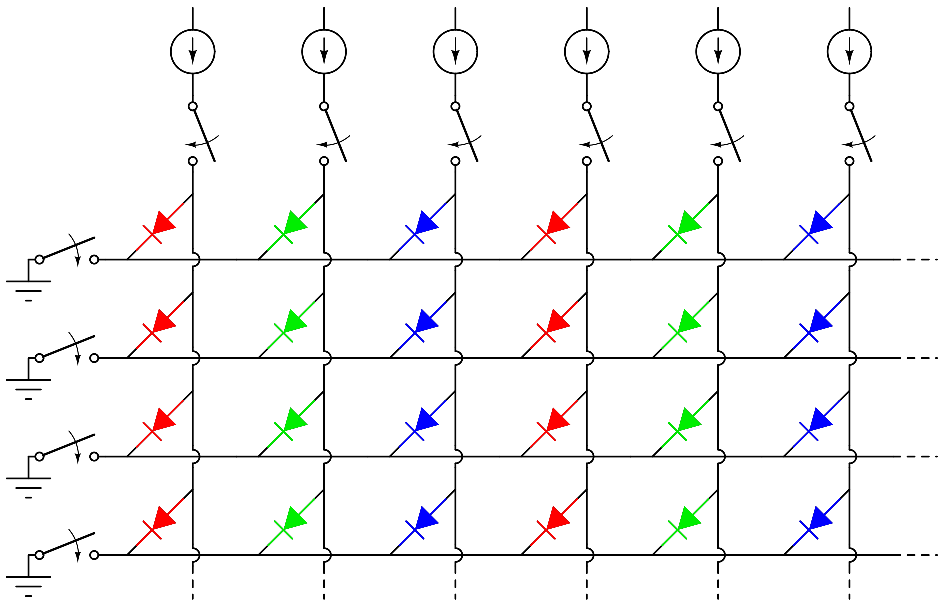


Figure 2.2: Scheme of the electrical circuit of a display.

The image itself is given as an $n \times m$ integer matrix $(r_{ij}) \in \{0, \dots, \varrho\}^{n\times m}$ representing its RGB values. The number ϱ determines the *color depth*, e.g. for 16.7 million colors, we have $\varrho = 255$. If $\varrho = 1$, the image is *monochrome*.

Since there are only $n + m$ contacts available, we can not make the diodes shine in an arbitrary pattern at a given time. This is the main difference to the active matrix devices where this issue is addressed by an additional TFT backplane with a transistor circuitry at every pixel. On the other hand this makes the display more expensive because of higher material costs and yield issues of the fabrication process.

However, at sufficient high framerates, e.g. 50 Hz, it is possible to display arbitrary images using passive matrix displays since the perception of the human eye averages the received amount of light over time. In the next section, we will explain in a simplified way how this works.

2.2 Driving Schemes

Consider the contacts for the rows and columns as switches. If the switch of row i and the switch of column j are closed, the pixel (i, j) shines with a brightness which is proportional to the applied electrical current I which is common to all row-column pairs in our case. Nevertheless, we can make each diode appear at an arbitrary brightness, say r_{ij}, by closing the switches for an appropriate amount of time, say t_{ij}. This method is called *pulse width modulation* (PWM). The relation between r_{ij} and t_{ij} is

$$r_{ij} = I \cdot t_{ij} \tag{2.1}$$

where we consider a dimensionless representation of all physical quantities in this thesis for the sake of simplicity.

As of yet, passive matrix OLED displays are driven in a row-by-row fashion. This means that the switches for the rows are activated one after the other. While row i is active, column j has to be active for the time t_{ij}, so row i can be turned off after $\max\{t_{ij} : j = 1, \dots, m\}$ time units. The time which is required to display the whole image is consequently

$$T = \sum_{i=1}^{n} \max\{t_{ij} : j = 1, \dots, m\}. \tag{2.2}$$

Plugging Equation (2.1) into Equation (2.2) yields

$$T = \frac{1}{I} \sum_{i=1}^{n} \max\{r_{ij} : j = 1, \dots, m\}.$$

To achieve a certain framerate, an image has to be displayed within a certain time-frame T_f. This means, that the amplitude of the electrical current is determined by

$$I = \frac{1}{T_f} \sum_{i=1}^{n} \max\{r_{ij} : j = 1, \dots, m\}.$$

The previous Equations show that the time for which the electrical current flows through the diodes is anti-proportional to its amplitude I. The aforementioned short lifetime of todays OLED displays is mainly due to the high current amplitude which is necessary to display images with a medium and high number of rows.

This means that the peak power which has to be emitted by the diodes of the display is very high while on the other hand, the diodes stay idle most of the time. The high amplitudes of electrical current which alternate with long idle times put the diodes under a lot of stress, which results in a short lifetime. A common notion of *lifetime* is the time T_{50}, i.e. the time of operation until the luminance drops to 50% of its initial value. However, this measure crucially depends on the conditions and the mode of operation as shown in [13]. Moreover, a user-study conducted by Nokia [14] suggests that already a degradation of 10% in the luminance yields annoying visual effects. This is due to different rates of degradation of the red, green, and blue diodes which causes a shift of the perceived colors, and due to the so-called burn-in effect where a often displayed image like the main menu leads to a faster degradation of certain pixels such that this image remains partially visible when displaying other images. Nevertheless, it is clear from [13] that a reduction of the amplitude of the electrical current drastically improves the lifetime of a passive matrix OLED device. Therefore, we will use the relative reduction of the electrical current as an objective measure to quantify the results of our algorithms.

In this thesis, we aim to overcome this problem by a different addressing mechanism. The simple but crucial observation is that, if two rows have the same content, we could drive them simultaneously and thereby we would save half of the time necessary for the two rows. Therefore the value of I could be reduced while the total time T remains within the time-frame T_f.

$$\begin{matrix} 109 & 238 & 28 \\ 112 & 237 & 28 \\ 150 & 234 & 25 \\ 189 & 232 & 22 \\ 227 & 229 & 19 \end{matrix} = \begin{matrix} 0 & 82 & 25 \\ 0 & 82 & 25 \\ 0 & 41 & 22 \\ 0 & 41 & 22 \\ 0 & 0 & 0 \end{matrix} + \begin{matrix} 0 & 0 & 0 \\ 112 & 155 & 3 \\ 112 & 155 & 3 \\ 189 & 191 & 0 \\ 189 & 191 & 0 \end{matrix} + \begin{matrix} 109 & 156 & 3 \\ 0 & 0 & 0 \\ 38 & 38 & 0 \\ 0 & 0 & 0 \\ 38 & 38 & 19 \end{matrix}$$

Figure 2.3: An example decomposition

Consider the example in Figure 2.3. Suppose that $I = 1$. If the image is displayed row-by-row, then the minimum time which is needed to display the image is $238 + 237 + 234 + 232 + 229 = 1170$ time units. But we can do better by a suitable decomposition of the image into three matrices. In the first one every even row is equal to its odd predecessor and in the second one every odd row is equal to its even predecessor. For a mathematically sound notation, we insert zero-rows at the borders such that all matrices are of the same size. The remainder is put into an offset matrix that is driven in the traditional way. By driving the equal rows simultaneously, we require only $82 + 41 + 155 + 191 + 156 + 38 + 38 = 701$ time units. This means that we could reduce the electrical current I by roughly 40%.

We could save even more by driving $3, 4, 5\dots$ rows simultaneously. Of course there is a saturation somewhere, unless the image is totally homogeneous. We will see that the hardware complexity and thus also the production costs of the driver chip increase with the multiline order. Hence, it is not worth to consider more than $k = 4$ simultaneously driven rows. Therefore, we will assume that the multiline order k is a constant where applicable. Moreover, it would be technically demanding to allow an arbitrary number of simultaneously activated rows. Furthermore, we will only combine consecutive rows for the following reasons. A real display is not ideal in the sense that all diodes have the same physical characteristics. However, neighboring diodes are more likely to be similar. Moreover, we will see that combining only consecutive rows yields an elegant mathematical formulation allowing the design of competitive algorithms in practice that do not leave much room for improvement by using non-consecutive multilines.

Since we only need to store the contents of the common part once, a decomposition where up to k consecutive rows are combined can be stored in k matrices $F^{(1)}, \dots, F^{(k)}$ not larger than the original image (see Figure 2.4).

$$F^{(2)} = \begin{pmatrix} 0 & 82 & 25 \\ 112 & 155 & 3 \\ 0 & 41 & 22 \\ 189 & 191 & 0 \end{pmatrix} \qquad F^{(1)} = \begin{pmatrix} 109 & 156 & 3 \\ 0 & 0 & 0 \\ 38 & 38 & 0 \\ 0 & 0 & 0 \\ 38 & 38 & 19 \end{pmatrix}$$

Figure 2.4: Compact representation of the example decomposition of Figure 2.3

2.3 Design Requirements

Before we start with the formal modeling of our problem, we define some goals that we want to achieve with the final design. The primary goal is to design algorithms that are well-suited for being implemented on a chip that drives an OLED device. From an economic point of view, the advantage of passive matrix is that the fabrication is cheaper compared to active matrix. Hence, the cost of the driver shall be as low as possible to keep the CMLA competitive. Therefore, we shall only use fixed precision integers since floating point arithmetic is too expensive. Furthermore, we want to compute a lossless decomposition. We shall develop algorithms that are fully combinatorial, i.e. that use only addition, subtraction, and comparisons, to keep the hardware complexity at a low level. Since the production cost of a chip is mainly determined by its area, our algorithms shall be memory efficient. Nevertheless, we aim for a solution that is capable of video applications. Hence, the algorithm shall offer possibilities for parallelization.

Chapter 3

Theoretical Foundations

In this chapter, we first deduce a formal model for the decomposition problem. It follows an immediate formulation as an integer linear program. The purpose of this formulation is twofold. First, it is the foundation of the subsequent theoretical investigations. Second it permits an early experimental evaluation of the *Consecutive Multiline Addressing* (CMLA) scheme by using standard tools, e.g. the commercial solver CPLEX.

Recall that by our design goals, we shall look for fully combinatorial algorithms. Thus, we examine the combinatorial structure of the linear programming formulation afterwards. This yields a certain network flow formulation which is tied to a special class of graphs. We discuss the structural properties of these graphs as they help us to derive efficient algorithms for our particular problem lateron.

The structure of the linear programming formulation motivates us to consider the related TRANSSHIPMENT problem. We present a generic approach by MAXFLOW/MINCUT, which is well-known from literature. Moreover, we develop a linear time algorithm for this problem on graphs with bounded bandwidth.

Furthermore, both algorithms for the TRANSSHIPMENT problem yield a separation routine for a second linear program with exponentially many constraints but with considerably fewer variables. That is, we linearly project the polyhedron onto the variables appearing in the objective. Thereby, we can solve much larger instances by CPLEX, and we can derive first fully combinatorial heuristics for our application.

On the theory side, this new formulation points to the field of network design so that it motivates a complexity analysis. To this end, we investigate different restrictions and relaxations of our problem to gain further insights that are of help in the development of an approximation algorithm. The design of an PTAS lays the theoretical foundation for a competitive algorithm in practice.

We conclude this chapter with an analysis of the canonical online problem, which helps us to further improve our algorithm.

Before we start, we define some notation that we will use in the remainder of this thesis. Moreover, we review some basics about linear programming and combinatorial optimization.

3.1 Notation and Preliminaries

Let S be a finite set and $f : S \to \mathbb{R}$ be a map. We define for any non-empty subset $U \subseteq S$ that

$$f(U) := \sum_{u \in U} f(u)$$

and furthermore $f(\emptyset) := 0$.

Let $G = (V, A)$ be a directed graph with (finite) nodeset V and arcset $A \subseteq V \times V$. That is, an arc a is an ordered pair of nodes, e.g. $a = (u, v)$ for some $u, v \in V$. We call u the tail of a and v the head of a. For any $X \subseteq V$, we define the sets

$$\begin{aligned} \delta^{out}(X) &:= \{(u,v) \in A : u \in X \wedge v \notin X\} \\ \delta^{in}(X) &:= \{(u,v) \in A : u \notin X \wedge v \in X\} \end{aligned}$$

If X consists of a single element, say i, then we will also use the abbreviation $\delta^{out}(i) := \delta^{out}(\{i\})$. The same holds for $\delta^{in}(\cdot)$. Moreover, for any arc-function $f : A \to \mathbb{R}$ and $(i, j) \in A$, we use only one pair of paranthesis for better readability, i.e. we define $f(i, j) := f((i, j))$.

We now review basic knowledge about linear programming and combinatorial optimization. A reader who is familiar with this field may safely skip to the next section. Since this thesis might be of interest for readers with an electrical engineering background, we will keep the preliminaries as concise as possible to allow an understanding of the key ideas of this chapter. Then again, an encyclopedic reference for these topics is Schrijver's book [15]. We assume that the reader has basic knowledge about linear algebra.

Given a non-zero vector $a \in \mathbb{R}^d$ and scalar $\beta \in \mathbb{R}$, we call the set

$$\{x \in \mathbb{R}^d : a^T x \leq \beta\}$$

a *halfspace* of dimension d. The intersection of finitely many halfspaces is called a *polyhedron*. Every polyhedron P defined by m halfspaces can be represented by a set

$$P := \{x \in \mathbb{R}^d : Ax \leq b\}$$

where the matrix $A \in \mathbb{R}^{m \times d}$ and the vector $b \in \mathbb{R}^m$ denote the inequalities defining the halfspaces. A polyhedron is called *rational* if and only if the matrix A and the vector b can be chosen to be rational or equivalently if and only if they can be chosen to be integer. In the forthcomming, we will only consider rational polyhedra and therefore we will discard the word rational.

Let $P := \{(x, y) \in \mathbb{R}^p \times \mathbb{R}^q : Ax + By \leq b\}$ be a polyhedron. Then the set

$$P' := \{x \in \mathbb{R}^p : \exists y \in \mathbb{R}^q : Ax + By \leq b\}$$

is also a polyhedron and it is called the *linear projection* of P onto the subspace of the x-variables. A well-known general method to compute an explicit representation of P' in terms of new inequalities $Cx \leq d$ is the *Fourier-Motzkin Elimination*. We mention this method just for the sake of completeness. We will present a problem specific method, when we will perform such a projection. The reverse process is called *lifting*. The interested reader is referred to [16].

A *linear program* (LP) is an optimization problem asking to find the maximum of a linear function, say $c^T x$ for some vector $c \in \mathbb{R}^d$, over a given polyhedron P. We call this function the *objective function*. The value

$$\max\{c^T x : x \in P\}$$

is called *objective value*. If it does not exist, then either $P = \emptyset$ or the objective function is unbounded over P. In the former case, we say that the LP is *infeasible*, and in the latter case it is simply called *unbounded*.

Note that maximizing a function $c^T x$ is equivalent to minizing $-c^T x$. Moreover, a polyhedron can also be given with a different notation, e.g.

$$P = \{x \in \mathbb{R}^d : x \geq 0, Ax \geq b\}, \tag{3.1}$$

which may be easily transformed by multiplying the inequalities by -1. Moreover, if we have have an equation, say $a^T x = \beta$, this is equivalent to the intersection of $a^T x \leq \beta$ and $-a^T x \leq -\beta$. Hence, we consider every optimization problem of a linear function subject to a set of linear inequalities and/or equations as a linear program. We summarize the inequalities and equations by the term *constraints*. The matrix A of (3.1) is therefore also called *constraint matrix* whereas constraints like $x \geq 0$ are often called bounds and considered separately. If the constraint matrix of a linear program has only $\{-1, 0, 1\}$-entries, we call the linear program *combinatorial*. A minimization linear program with a non-negative objective vector c and a polyhedron of the form of (3.1) with a non-negative constraint matrix A is called a *covering* problem.

Let $P = \{x \in \mathbb{R}^d : Ax \leq b\}$ be a non-empty polyhedron and let $z \in P$. If a constraint $a^T x \leq \beta$ is fulfilled with equality with respect to z, i.e. $a^T z = \beta$, then we say that the constraint is *tight*. Let $A_z x \leq b_z$ denote the constraints that are tight with respect to z. If $rank(A_z) = d$, i.e. there are dimensionally many linearly independent tight constraints, then we call z a *feasible basic solution* since it is the unique solution of the system $A_z x = b_z$.

Note that a polyhedron may not have a feasible basic solution even if it is non-empty as the example $\{(x, y) \in \mathbb{R}^2 : x \geq 0\}$ shows. However, if it has at least one, then the optimum of any linear objective function $c^T x$ is attained at a basic feasible solution whenever the optimum is finite. In this thesis, we will only consider polyhera containing at least one feasible basic solution if they are non-empty.

A polyhedron P is called *integer* if and only if for each $c \in \mathbb{Z}^d$ the value of $\max\{c^T x : x \in P\}$ is an integer if it is finite. This implies that all feasible basic solutions of an integer polyhedron are also integer. The reverse direction of this statement is only true if there is at least one feasible basic solution.

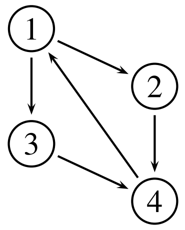
In this thesis, we only consider combinatorial problems, i.e. problems with a $0/\pm 1$ constraint matrix. There is the following relation between a certain class of $0/\pm 1$ matrices and integer polyhedra. Let $A \in \{-1,0,1\}^{m\times d}$ such that the determinant of every quadratic submatrix is in $\{-1,0,1\}$. We say that these matrices are *totally unimodular*. It is well known that if a matrix $A \in \{-1,0,1\}^{m\times d}$ is totally unimodular, then the polyhedron $\{x \in \mathbb{R}^d : Ax \leq b\}$ is integer for any integer vector $b \in \mathbb{Z}^m$. If there is a feasible basic solution $z \in P$, this can be easily seen as follows. Then z is the unique solution to the system $A_z x = b_z$ with at least d linearly independent constraints as defined before. W.l.o.g. we may assume that there are exactly d linearly independent tight constraints such that A_z is a $d \times d$ matrix with determinant ± 1 because it has full rank and A is totally unimodular. Hence by Cramer's rule, its inverse A_z^{-1} is also integer. Moreover, $z = A_z^{-1} b_z \in \mathbb{Z}$ since b is also integer.

Total unimodularity will be relevant for our purpose since we will encounter 0/1 matrices that have the *consecutive ones property*. This means that there is a permutation of the rows and columns such that in each column or in each row appears only one contiguous block of 1s. These matrices are totally unimodular by the following induction. Clearly, a square 0/1 matrix with one row has either determinant 0 or 1. Assume that we have a $n \times n$ submatrix with $n > 1$. Since permutating rows and columns and taking the transpose do not change the absolute value of the determinant, we may assume that the square matrix has one block of consecutive ones in each column. If the first row does not contain a 1 the determinant is 0 and we are done. So, we assume that there is a column for which the block of 1s starts in the first row. Let the shortest of these blocks be (w.l.o.g.) in the first column. Since row operations do not change the determinant, we may subtract the first row from all following rows that also have a 1 in the first column. Hence, the new matrix has only one 1 left in the first column and the submatrix, say A', that we get by deleting the first row and the first column also has the consecutive ones property.

$$\begin{pmatrix} 1 & * & \cdots & * \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{pmatrix}$$

Thus, the determinant is given by the determinant of the submatrix A'. By the induction hypothesis, it is either 0 or ± 1 which proves the claim.

There is another prominent subclass of totally unimodular matrices which is relevant for us. These are matrices with exactly one 1 and one -1 in each column, and the remaining entries are 0. Total unimodularity is easily established by an induction similar to the previous one. Moreover, these matrices are exactly the node-arc-incidence matrices of directed graphs. That is, given a directed graph $G = (V, A)$, we consider the matrix $N \in \{-1,0,1\}^{|V|\times|A|}$ which has a row for each node in V and a column for each arc in A such that there is a 1 in the row of its tail, and a -1 in the row of its head, and 0 otherwise as depicted in the example in Figure 3.1.



$$\begin{pmatrix} 1 & 1 & -1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & -1 \end{pmatrix}$$

Figure 3.1: Example of a graph and its node-arc-incidence matrix

We call a function $f : A \to \mathbb{R}_{\geq 0}$ that assigns a non-negative number to every arc a *network flow* or just a *flow*. Sometimes we also consider a flow as a vector $f \in \mathbb{R}^{|A|}_{\geq 0}$ whenever it is convenient. For any integer vector $d \in \mathbb{Z}^{|V|}$ we may consider the *flow polyhedron* $P := \{f \in \mathbb{R}^{|A|} : 0 \leq f, Nf = d\}$, or if we are additionally given upper bounds $u \in \mathbb{Z}^{|A|}$ for the flow-variables, we consider $P := \{f \in \mathbb{R}^{|A|} : 0 \leq f \leq u, Nf = d\}$. We call these upper bounds *capacities*. Since N is totally unimodular both polyhedra are integer. Moreover, the vector d is called *demand vector*. Its entries are simply called *demands*. We will also use of the equivalent notation of the constraints with functions. That is, for all $i \in V$ we have

$$f(\delta^{out}(i)) - f(\delta^{in}(i)) = d(i)$$

where we switch seemlessly between vectors and functions, e.g. between $d \in \mathbb{Z}^{|V|}$ and $d : V \to \mathbb{Z}$. Though we consider the flow-polyhedron as a subset of $\mathbb{R}^{|A|}$, we may restrict ourselves to feasible basic solutions which are integer.

If we are given a certain flow f that respects the capacities, i.e. $f(a) \leq u(a)$, we define the residual graph G_f that contains for each arc $(s,t) \in A$ the arc (s,t) if $f(s,t) < u(s,t)$ with the residual capacity $u'(s,t) := u(s,t) - f(s,t)$, and the arc (t,s) if $f(s,t) > 0$ with the residual capacity $u'(t,s) := u(t,s) + f(s,t)$. A flow that respects the residual capacities can be added to f without losing non-negativity or violating any capacities. More on flows can be found in [17].

The reason why we are particularly interested in network flows is that there are efficient combinatorial algorithms for linear optimization problems over flow polyhedra. This means that there are algorithms that solve such problems in polynomial time. We assume that the reader has basic knowledge in complexity theory to relate to the terms *NP-complete* and *polynomial time solvable*. If this is not the case, we refer the interested reader to [18] and we give the following rough distinction for the practitioner. If a problem is NP-complete then it is unlikely that we find an efficient algorithm that solves the problem exactly, i.e. we can solve only small instances in practice. If it is polynomial time solvable, then there is an algorithm that solves such a problem within a running time polynomially bounded in the size of the input, and we may hope to come up with an algorithm in practice that exactly solves much larger instances in a timely manner.

We will describe the bounds for the running times and also the space consumptions with the so-called O-notation. Let $f, g : \mathbb{N} \to \mathbb{R}$ functions from the natural numbers to the reals. The function f is in $O(g)$ if there is a constant $c > 0$ and a number n_0 such that $f(n) \leq c \cdot g(n)$ for all natural numbers $n \geq n_0$. We use the notation $f = O(g)$.

Linear programs are polynomial time solvable by the ellipsoid method [19]. However, the problem of finding the optimum integer solution, i.e. the optimum of a linear function, over the set $P \cap \mathbb{Z}^d$ for a general polyhedron $P \subseteq \mathbb{R}^d$ is NP-complete. We call this problem *integer linear programming* (ILP). In this thesis, we will exclusively consider minimization ILPs. That is the task to find $\min\{c^T x : x \in P \cap \mathbb{Z}^d\}$ for some objective vector $c \in \mathbb{Z}^d$ and some polyhedron $P \subseteq \mathbb{R}^d$. The corresponding linear program where we drop the integrality condition is also called the *relaxation*. The ratio of the integer optimum objective value and the optimum objective value of the relaxation is called the *integrality gap*. We have

$$\left| \frac{\min\{c^T x : x \in P \cap \mathbb{Z}^d\}}{\min\{c^T x : x \in P\}} \right| \geq 1$$

since for minimization problems the optimum objective value of the relaxation is never greater than the integer optimum objective value. For non-empty integer polyhedra the integrality gap is 1 for any choice of $c \in \mathbb{Z}^d$ for which minimum is finite. Hence, if we consider only integer polyhedra as input, then linear programming and integer linear programming become equivalent, and hence the latter becomes polynomial time solvable.

As it may be intractable to solve a problem exactly, we shall be content with an approximate solution instead. That is, we wish to find a feasible solution that is not too far from an optimal one. We measure the quality of such an approximate solution, say APX, relative to the optimum objective value, say OPT. Since we will exclusively consider minimization problems, we define the term *approximation ratio* as the quotient $\frac{APX}{OPT}$.

3.2 The Basic Formal Model

In this section, we describe our mathematical model of CMLA. For the sake of clarity, we first explain the case of multiline order $k = 2$. That is, we decompose the image content into two matrices $F^{(1)}$ and $F^{(2)}$ where the former represents the singlelines and the latter the doublelines. We will also refer to this case as the *Doubleline* case. Recall that each pixel is covered by one singleline and two doublelines, one covering also the line above and the other covering also the line below (see Figure 3.2).

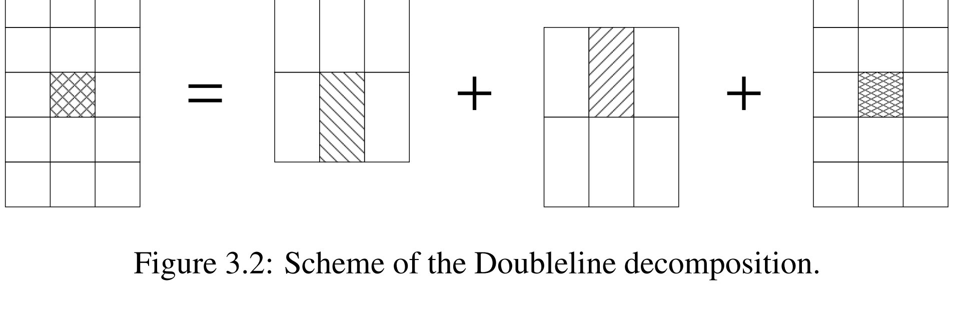


Figure 3.2: Scheme of the Doubleline decomposition.

Given an image $R = (r_{ij}) \in \{0, \ldots, \varrho\}^{n \times m}$, the luminance of each pixel is distributed to the doubleline that also covers its neighbor above, to the doubleline that also covers the neighbor below, and to the singleline that covers this pixel, i.e.

$$r_{ij} = f^{(2)}_{i-1,j} + f^{(2)}_{ij} + f^{(1)}_{ij}$$

for all $i = 2, \ldots, n-1$ and $j = 1, \ldots, m$. For the first and the last row, we have $f^{(2)}_{1j} + f^{(1)}_{1j} = r_{1j}$ and $f^{(2)}_{n-1,j} + f^{(1)}_{nj} = r_{nj}$, respectively.

Since, we can not produce "negative" light, we require non-negativity for the entries in $F^{(1)}$ and $F^{(2)}$. Written in matrix notation, these constraints are of the form

$$R = T^{(2)} \cdot F^{(2)} + T^{(1)} \cdot F^{(1)}$$

where $T^{(\alpha)}$ is the matrix that indicates which row switches are closed in the each step of the driving scheme. That is, each column represents one step and the non-zeros of a column determine which row switches are closed in the corresponding step. Thus, we have

$$T^{(1)} = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & \ddots & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix} \qquad T^{(2)} = \begin{pmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ & 1 & 1 & & & & \\ & & 1 & \ddots & & & \\ & & & \ddots & 1 & & \\ & & & & 1 & 1 & \\ & & & & & 1 & 1 \\ & & & & & & 1 \end{pmatrix}.$$

By this representation, the generalization for combining more than two lines at once becomes straight forward. If we want to drive up to k consecutive lines simultaneously, we get

$$R = \sum_{\alpha=1}^{k} T^{(\alpha)} \cdot F^{(\alpha)} \qquad \text{with} \qquad T^{(\alpha)}_{i,i'} = \begin{cases} 1 & i = i', \ldots, i' + \alpha - 1 \\ 0 & \text{otherwise} \end{cases}$$

We could also consider using matrices of different form than the just defined $T^{(\alpha)}$. But then, we lose a very nice structural property that we will exploit in the next section. However, we will see that using only consecutive lines works well in practice.

Recall that our objective is to minimize the time that is needed to display the image. As said before, each column of the matrices $T^{(\alpha)}$ represents a driving step. The duration of a step, say column i in matrix α, is determined by the maximum entry of row i in the matrix $F^{(\alpha)}$. Hence, we want to minimize the quantity

$$\sum_{\alpha=1}^{k} \sum_{i=1}^{n-\alpha+1} \max\{f^{(\alpha)}_{ij} : j = 1, \ldots, m\}$$

Since the set of feasible decompositions is given by finitely many linear constraints, it constitutes a polyhedron. However, our objective function is not linear as of yet. This issue can be fixed easily by adding auxiliary variables, say $u_i^{(\alpha)}$, which represent the maxima. We substitute each term of the objective by the respective variable and upper bound the variables of each row by adding the constraints

$$f_{ij}^{(\alpha)} \leq u_i^{(\alpha)}$$

for all i, j, and α. Putting all together, e.g. for $k = 2$, we get the integer linear program

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} u_i^{(1)} + \sum_{i=1}^{n-1} u_i^{(2)} & & \\ \text{s.t.} \quad & f_{ij}^{(1)} + f_{i-1,j}^{(2)} + f_{ij}^{(2)} = r_{ij} & \text{for all } i, j & \\ & f_{ij}^{(\alpha)} \leq u_i^{(\alpha)} & \text{for all } i, j, \alpha & \\ & f_{ij}^{(\alpha)} \in \mathbb{Z}_{\geq 0} & \text{for all } i, j, \alpha & \end{aligned} \tag{3.2}$$

Since we have a minimization problem, in every optimal solution the u-variables will be tight at the respective row-maxima. Hence, if in any feasible solution the f-variables are integer then the u-variables will be integer, too. Conversely, if we fix the u-variables to integer values then each basic feasible solution of the f-variables will also be integer because the matrices $T^{(\alpha)}$ have the column-wise consecutive-ones property and hence they are totally unimodular.

However, the complete polyhedron is not integer as the following simple example for the case $k = 2$ shows. Let the image be given by

$$R = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

This yields an optimal objective value of $\frac{5}{2}$ by the decomposition

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

whereas the best integer solution has an objective value of 3. Hence the integrality gap of the linear programming formulation is at least $\frac{6}{5}$.

Recall that one of our design goals is to use only fixed precision arithmetic. Therefore, we have introduced the integrality constraint in the linear programming formulation. Moreover, we are looking for a fully combinatorial algorithm to solve or approximate the problem in practice. Hence, it is immediate to look for a connection to graph problems such as network flows. In fact, the combinatorial structure of our constraints yields a link to that field as we will show in the next section.

3.3 The Network Model

In this section, we exploit the column-wise consecutive-ones property of parts of the constraint matrix, that is the family of $T^{(\alpha)}$ for $\alpha = 1, \ldots, k$. Since these constraints are equations, we can subtract rows from each other without changing the solution space like in Gaussian elimination. By adding the dummy constraint $0 = 0$ at the end and subtracting from each row its predecessor, we transform the constraint matrix into an node-arc-incidence-matrix of a directed acyclic graph. Formally speaking, we define $N_\alpha \in \{-1, 0, 1\}^{n+1 \times n+1-\alpha}$ by

$$N^{(\alpha)}_{i,i'} = T^{(\alpha)}_{i,i'} - T^{(\alpha)}_{i-1,i'}$$

for all $\alpha = 1, \ldots, k$, $i = 2, \ldots, n$, and $i' = 1, \ldots, n + 1 - \alpha$. Moreover, $N^{(\alpha)}_{1,i'} = T^{(\alpha)}_{1,i'}$ and $N^{(\alpha)}_{n+1,i'} = -T^{(\alpha)}_{n,i'}$ for all $i' = 1, \ldots, n + 1 - \alpha$. Hence, the matrices $N^{(\alpha)}$ have exactly one 1 and one -1 in each column. For $n = 5$, e.g., we have

$$N^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \qquad N^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \qquad \cdots$$

Note that the matrices $N^{(\alpha)}$ have the same number of rows. For a given multiline order k, let N be the matrix that we get by a concatenation of the matrices $N^{(\alpha)}$ for $\alpha = 1, \ldots, k$, i.e.

$$N := \begin{pmatrix} N^{(1)} & N^{(2)} & \cdots & N^{(k)} \end{pmatrix}.$$

For the multiline orders $k = 2, 3, 4$, the corresponding graphs, which we call *display graphs*, are depicted in Figure 3.3.

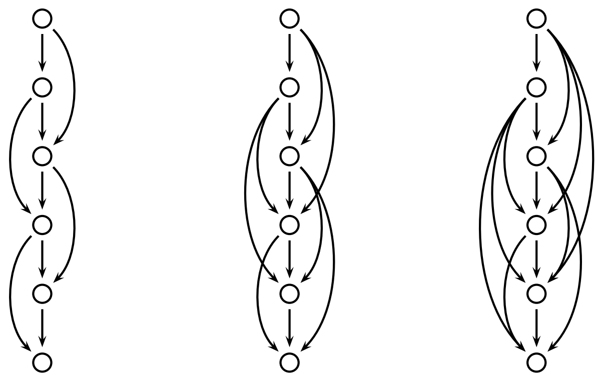


Figure 3.3: Examples for a display graph with $n = 5$, $k = 2, 3, 4$

3.3.1 Properties of the display graph

This subsection is dedicated to the structural properties of the display graph. We start with the graph for the singleline problem, i.e. $k = 1$. So, we need one arc for each row. Hence, let G_n be a path of length n, that is $G_n = (V, A)$ with $V = \{1, \ldots, n{+}1\}$ and $A = \{(i, i{+}1) : i = 1, \ldots, n\}$. For the case $k = 2$, we have to add an arc for each pair of consecutive arcs of G_n. In other words, we connect each pair of nodes i, i' which have a distance of at most 2 in G_n. Hence, the display graph for $k = 2$ is the square of G_n. In general, the display graph for arbitrary k is the k-th power of G_n. Thus, we denote the display graph for given n and k by G_n^k.

Definition 1 *Given a graph $G = (V, A)$, the k-th power of G, denoted by G^k, is a graph on the same nodeset V with an arc from u to v if there is a path of length at most k from u to v in G.*

This construction implies several structural properties. First of all, it is easy to see that the display graph is acyclic. Moreover, it contains a Hamiltonian path. Hence, it has a unique topological order. We will always consider the nodes to be ordered this way in the following. Thus, the terms like *first*, *last*, *smallest* or *largest* node are well-defined.

Moreover, the outdegrees and indegrees of G_n^k are bounded by k. There are also other well-know quantities that depend only on the parameter k, e.g. *cutwidth*, *treewidth*, *pathwidth*, and *bandwidth*. Since they are defined for undirected graphs, we refer to the underlying undirected graph of the display graph.

Definition 2 *Given an undirected graph $G = (V, E)$, the cutwidth of a particular ordering of the nodes, say $(v_1, \ldots, v_n)$, is the maximum number of edges in a cut $\delta(\{v_1, \ldots, v_i\})$ for $i = 1, \ldots, n$. The cutwidth of G is the minimum cutwidth taken over all orderings of V.*

Lemma 1 *A display graph G_n^k has a cutwidth of $O(k^2)$.*

Proof: We may use the canonical ordering of a display graph G_n^k to bound its cutwidth. In this ordering, the width of a cut in the underlying undirected graph is equivalent to the number of outgoing arcs. Moreover, we have

$$|\delta^{out}(\{1, \ldots, i\})| \leq k + k - 1 + \cdots + 1 = \frac{1}{2}k(k+1) = O(k^2)$$

for all $i = 1, \ldots, n$. ■

Definition 3 *A tree decomposition of a graph $G = (V, E)$ is a tree $T = (V_T, E_T)$ and a subset $V_t \subseteq V$ associated with each node $t \in V_T$ such that the following statements hold*

1. *Every node of G belongs to at least one V_t.*

2. *For every edge $\{u, v\} \in E$, there is a $t \in V_T$ with $u, v \in V_t$.*

3. *Let $r, s, t \in V_T$ be such that s lies on the path from r to t in T. Then, $V_r \cap V_t \subseteq V_s$.*

The width of a tree decomposition is the maximum cardinality of a set V_t minus 1. The treewidth is the minimum width ranging over all tree decompositions. A path decomposition is a tree decomposition where T is a path. The pathwidth is defined analogously.

It is shown in [20] that the cutwidth is an upper bound on the pathwidth. Moreover, it is easy to see that twice the cutwidth is also an upper bound on the maximum degree of G. Hence if the cutwidth is bounded by a constant then the pathwidth and the degrees are also bounded. Conversely, in [21] it is shown that the cutwidth is at most the product of the maximum degree and the pathwidth. Thus bounded degree and bounded pathwidth also imply bounded cutwidth.

Lemma 2 *We may assume w.l.o.g. that $k \le n$ since otherwise the statement is trivially fulfilled by choosing $V_T = \{V\}$. The pathwidth of a display graph G_n^k is bounded by k.*

Proof: Consider the decomposition $T = (V_T, E_T)$ with

$$\begin{aligned} V_T &:= \{1, \dots, n-k+1\} \\ E_T &:= \{(i, i+1) : i = 1, \dots, n-k\} \\ P_i &:= \{v \in V : v = i, \dots, i+k\} \quad \text{for all } i \in V_T \end{aligned}$$

It is indeed a path decomposition since

1. each node i of G_n^k is contained in P_i if $i \le n-k$, or in P_{n-k+1} otherwise,

2. for each arc $(u, v) \in A$ the target node v is also contained in P_u if $u \le n-k$, or otherwise both are contained in P_{n-k+1}, and

3. if $r, s, t \in V_T$ with $r \le s \le t$ then $P_r \cap P_t$ is either the empty set if $r + k < t$, or else it is {t,...,r+k}. In both cases the intersection is contained in P_s.

Since the size of each P_i is $k+1$, the pathwidth is bounded by k. ■

In the following, we show the relation between display graphs G_n^k and graphs with the bandwidth k.

Definition 4 *Given an undirected graph $G = (V, E)$, the bandwidth of a particular numbering of the nodes, that is a one-to-one function $f : V \to \{1, \dots, |V|\}$, is given by*

$$\min\{|f(u) - f(v)| : \{u, v\} \in E\}.$$

The bandwidth of G is the minimum bandwidth taken over all numberings.

Lemma 3 *The bandwidth of a display graph G_n^k is bounded by k and conversely any undirected graph G' with $n + 1$ nodes and bandwidth k is a subgraph of the underlying undirected graph of G_n^k.*

Proof: It is easy to see that by construction the bandwidth of the canonical ordering is bounded by k. To show the second part of the lemma, we consider $G' = (V, E')$ such that $V = \{1, \dots, n+1\}$ and the bandwidth of the identity is k. Let $\{u, v\} \in E'$ be an arbitrary edge. W.l.o.g. we may assume that $u < v$. Because of the bandwidth condition, we have $v \le u + k$. Thus, for each edge in E' there is a corresponding arc in G_n^k. $\blacksquare$

Note that G_n^n is the complete directed acyclic graph with $n + 1$ nodes. But also G_n^k with $k < n$ exhibits a special structure which we call *partially complete*. This property is formally defined as follows.

Definition 5 *We call a directed acyclic graph $G = (V, A)$* partially complete*, if and only if there is a topological order of V such that for any three nodes $i, j, k \in V$ with $i < j < k$ the following two implications hold.*

$$(i, j) \in A \wedge (i, k) \in A \;\Rightarrow\; (j, k) \in A \qquad \textit{and} \tag{3.3}$$

$$(i, k) \in A \wedge (j, k) \in A \;\Rightarrow\; (i, j) \in A \tag{3.4}$$

3.3.2 From images to demands

As of yet, we have only transformed the constraint matrix, but we have to apply the same transformation also to the right-hand sides of the equations. Hence, for an image $R = (r_{ij}) \in \mathbb{Z}_{\ge 0}^{n \times m}$, we define $D = (d_{ij}) \in \mathbb{Z}^{n+1 \times m}$ by

$$d_{ij} = \begin{cases} r_{1j} & \text{if } i = 1 \\ r_{ij} - r_{i-1,j} & \text{if } i = 2, \dots, n \\ -r_{nj} & \text{if } i = n + 1 \end{cases}$$

for all $j = 1, \dots, m$ such that the constraints are now of the form

$$D = \sum_{\alpha=1}^{k} N^{(\alpha)} \cdot F^{(\alpha)}$$

and the problem can be understood as follows.

Problem 1 (CMLA) *Given a display graph G_n^k and a family of demand functions $d_j : V \to \mathbb{Z}$ for each $j = 1, \dots, m$, assign integer capacities $u : A \to \mathbb{Z}_{\ge 0}$ to the arcs such that for each j there exists a feasible flow $f_j : A \to \mathbb{Z}_{\ge 0}$ that satisfies the corresponding demands d_j and respects the capacities u. Among all feasible assignments of capacities, we look for the one that minimizes $u(A)$.*

Henceforth, we call the f-variables *flow-variables* and the u-variables *capacities*. Moreover, the flow-variables for each $j = 1, \ldots, m$ are only coupled by the capacities. Hence, if we fix the capacities, we can compute the flow-variables for each j independently. Since the flow-variables do not appear in the objective, any feasible solution is sufficient. The problem then amounts to m independent TRANSSHIPMENT problems. We will define this problem in the next section and we will discuss a generic solution for general graphs and a custom-tailored one for display graphs G_n^k. We will assume for time being that the capacities are given. In the section thereafter, we will consider the problem of minimizing the capacities.

3.4 The Transshipment Problem

In general, the TRANSSHIPMENT problem is stated as follows.

Problem 2 *Given a directed graph $G = (V, A)$, a demand function $d : V \to \mathbb{Z}$, and capacities $u : A \to \mathbb{Z}_{\geq 0}$, compute a flow $f : A \to \mathbb{Z}$ with*

$$f(\delta^{out}(i)) - f(\delta^{in}(i)) = d(i)$$

for all $i \in V$ and

$$0 \leq f(a) \leq u(a)$$

for all $a \in A$ or assert that no such flow exists.

3.4.1 The generic approach

We may solve this problem by a MAXFLOW computation (cf. chapter 11 of [15]). The construction is as follows. We introduce a *supersource* s and a *supersink* t to the graph. The supersource is connected to all vertices $i \in V$ with $d(i) > 0$. We assign the capacities $u(s, i) := d(i)$ to these newly created arcs (s, i). Analogously, the vertices $i \in V$ with $d(i) < 0$ are connected to the supersink t where we set $u(i, t) := -d(i)$.

For a feasible TRANSSHIPMENT, the flow must be tight at every arc leaving the supersource or entering the supersink, respectively. Hence, it is necessary that the value of the MAXFLOW, say φ, is given by

$$\varphi = u(\delta^{out}(s)) = u(\delta^{in}(t)) \tag{3.5}$$

Note this implies $d(V) = 0$ which can be checked easily. Moreover, it is trivially fullfilled by the construction of the demands in our application. Therefore, we always assume $d(V) = 0$ in the following.

To assert the case if no feasible solution exists, we can use the well-known MAXFLOW/MINCUT theorem.

Theorem 1 (MaxFlow/MinCut) *For a directed graph $G = (V, A)$, a capacity function $u : A \to \mathbb{Z}_{\geq 0}$ and two vertices $s, t \in V$, the value of the maximum $s - t$-flow is equal to the value of the minimum $s - t$-cut.*

See, e.g., chapter 10 of [15] for a proof. So, after the MAXFLOW computation, we check Equation (3.5). If $\varphi < u(\delta^{out}(s))$, then there is an $s - t$-cut, say $X\dot{\cup}\{s\}$ where $X \subset V - \{t\}$, in the extended graph with

$$u(\delta^{out}(X\dot{\cup}\{s\})) = \varphi < u(\delta^{out}(s))$$

certifying the infeasibility of the TRANSSHIPMENT instance. Translated back to the original graph this is equivalent to

$$u(\delta^{out}(X)) < d(X)$$

Conversely, if no such $X \subset V$ exists, i.e.

$$u(\delta^{out}(X)) \geq d(X) \qquad \text{for all } X \subset V \tag{3.6}$$

holds, then the given TRANSSHIPMENT instance is feasible. We refer to these inequalities as *cut inequalities* in the following.

So, the goal of the TRANSSHIPMENT problem is two-fold. If we are given feasible capacities, we wish to compute corresponding flow-variables. Otherwise, we wish to find the bottleneck, i.e. a cut inequality which is not satisfied, since finding violated inequalities for a given assignment to the variables is a key idea for solving linear programs. It is well known [22] that the *linear optimization* problem over a given polyhedron is polynomial time equivalent to the *separation* problem for this polyhedron. Also some of our heuristics (see Section 4.2) rely on the solution of the separation problem for the cut inequalities (3.6). In our setting, the separation problem is the following:

Problem 3 *Given a capacity assignment $u \geq 0$ for a display graph $G_n^k = (V, A)$ and demands $d : V \to \mathbb{Z}$, determine whether u is feasible and if not, compute a subset $X \subset V$ such that $u(\delta^{out}(X)) < d(X)$.*

Algorithmically, we perform a MAXFLOW computation and check whether the capacities are feasible as mentioned before. If they are not feasible, we examine the residual graph by a breadth-first search and identify the connected component containing the supersource and get the respective cut inequality.

There are several well-known MAXFLOW algorithms, e.g. Dinits' blocking flow method [23] or Goldberg's push-relabel method [24], which work on general graphs (see [25] for a survey). Moreover, there is the concept of *capacity scaling* [26] which makes use of the following idea.

Consider the bit representation of the capacities u. In the i-th iteration, we solve the MAXFLOW problem with respect to the i most significant bits of u based on the solution of the previous iteration. That is, we wish to find a maximum flow f' subject to the capacities $u' := \lfloor u/2^i \rfloor$ provided that we already have a maximum flow f'' subject to the capacities $u'' := \lfloor u/2^{i+1} \rfloor$. We observe first that $2 \cdot f''$ is also a maximum flow subject to $2 \cdot u''$. Moreover, u' and $2 \cdot u''$ differ by at most 1 per arc. Hence, we need at most $|A|$ paths to augment $2 \cdot f''$ to f' since we may increment the capacities arc by arc.

Since in some of our heuristics, we iteratively increase the capacities, we may use the previous observation to avoid the scaling from scratch each time. In particular for some of our strategies, we increase the capacity of one arc by some integer constant c in each iteration.

Definition 6 *Given two capacity functions $u, u' : A \to \mathbb{Z}_{\geq 0}$ for some graph $G = (V, A)$, we call them* capacity adjacent*, if they differ exactly at one arc, i.e.*

$$|\{a \in A : u(a) - u'(a) \neq 0\}| = 1$$

Theorem 2 *Given two capacity adjacent assignments $0 \leq u \leq u'$, say with*

$$u'(\bar{a}) - u(\bar{a}) = c$$

for one particular arc $\bar{a} \in A$ and some constant $c > 0$, and suppose that one is given a maximum flow f subject to u, then the separation problem for u' can be solved in linear time. More precisely one can compute a maximum flow $f' \leq u'$ and a minimum cut $X' \subset V$ with respect to u' in linear time.

Proof: Note that in this case there only exist at most c augmenting paths in the residual network and moreover these paths have to take the arc $\bar{a}$. Since each of these paths can be found in linear time by depth-first search, the complete update takes only linear time. As mentioned before, a minimum cut $X' \subset V$ can be found within the same timebound. ■

We will use this approach in practice since we can benefit from capacity adjacent updates and in case we do not adhere to this strategy we can easily fall back to the blocking flow method. We will present an evaluation in Section 4.2. The methods mentioned in this section are designed for general graphs. In the next section, we take the special structure of the display graph into account to solve the TRANSSHIPMENT problem.

3.4.2 A linear time algorithm for bounded bandwidth

Since the multiline order k is relatively small, it is reasonable to examine the TRANSSHIPMENT problem of G_n^k when k is a constant. Thus, the cutwidth, bandwidth, pathwidth, and treewidth are bounded as shown in Section 3.3.1. We will use the bounded bandwidth of the display graph G_n^k as the key feature to derive a fully combinatorial linear time algorithm.

We first show how to solve the separation problem for fixed k in linear time. That is the problem of finding a violated inequality, i.e. a violated cut in our case. Afterwards, we will extend this partial result, which we already presented in [7], to find also a feasible flow, if one exists, within the same timebound.

Theorem 3 *The separation problem for the cut inequalities* (3.6) *can be solved by a fully combinatorial algorithm in linear time for fixed k.*

Proof: It is sufficient to find a subset $X \subset V$ such that $u(\delta^{out}(X)) - d(X)$ is as small as possible. If the minimum is negative, we have found a violated inequality. Otherwise, all inequalities are fulfilled. We reduce this problem to a shortest path computation in an auxiliary directed acyclic graph.

In order to find such a subset X, we partition the vertices $V = \{1, \dots, n+1\}$ into consecutive blocks $B_1, \dots, B_\ell$ of size k where $\ell := \lceil (n+1)/k \rceil$. That is,

$$B_i := \{(i-1) \cdot k + 1, (i-1) \cdot k + 2, \dots, i \cdot k\} \quad \text{for } i = 1, \dots, \ell - 1$$

and $B_\ell := \{(\ell - 1) \cdot k + 1, \dots, n+1\}$. This partition is similar to the path decomposition in the proof of Lemma 2, in fact, the blocks constitute a minimal subset of the P_i such that each node is covered.

Recall that the arcs are of the form (i, i'), where $i' \leq i + k$. Hence, the arcs, which have their tails in block B_i, have their heads either in the same block or in B_{i+1}.

We now consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where the vertex set $\mathcal{V}$ contains all subsets of the sets B_i and there is an edge (S_1, S_2) if there exists an index i such that $S_1 \subseteq B_i$ and $S_2 \subseteq B_{i+1}$ as shown in Figure 3.4. Note that the empty set is a subset of each block. Thus, we introduce the vertices Z_i which are associated to the empty subsets of the respective blocks B_i.

Furthermore $\mathcal{G}$ has an additional vertex s and an additional vertex t together with edges (s, S) for each $S \subseteq B_1$ and (S, t) for each $S \subseteq B_\ell$. For the sake of clarity, we will use the terms *vertex* and *edge* for the graph $\mathcal{G}$ and the terms *node* and *arc* when we refer to the display graph G_n^k. We call the graph $\mathcal{G}$ *separation graph* in the following.

As illustrated in Figure 3.4, a path from s to t in $\mathcal{G}$ specifies a subset $X \subseteq V$ of the display graph in a natural way (and also vice versa): Given a path from s to t, take the union of the subsets represented by the inner vertices of the path.

It remains to define edge weights in $\mathcal{G}$ such that the weight of such a path is exactly $u(\delta^{out}(X)) - d(X)$. For this consider an edge (S_1, S_2) with head and tail different from s and t The weight of this edge is defined as

$$u(\delta^{out}(S_1) - \delta^{in}(S_2)) - \sum_{i \in S_2} d(i).$$

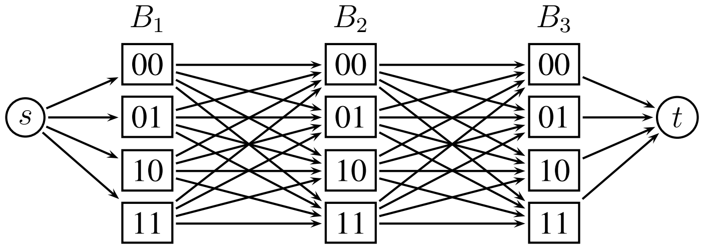


Figure 3.4: Solving the separation problem by a shortest path computation in a DAG (here $k = 2$). Each rectangle represents a subset of the corresponding block and the label indicates its characteristic vector. Hence, concatenating the labels of a path from s to t gives the characteristic vector of a cut X.

That is, from the capacities of the arcs, which have their tails in S_1 and their heads neither in S_1 nor in S_2, we subtract the demands of the nodes in the set S_2. The weight of an edge (s, S) is defined as $-\sum_{i \in S} d(i)$ and the weight of an edge (S, t) is $u(\delta^{out}(S))$.

In this way, the weight of a path from s to t in $\mathcal{G}$ is equal to $u(\delta^{out}(X)) - d(X)$, where X is the set which is represented by the path. Thus, the separation problem can be reduced to a shortest path problem in the directed acyclic graph $\mathcal{G}$ with roughly $2^k \cdot (n+1)/k = O(n)$ vertices, if k is fixed. Since then the degree is also bounded, we can solve the separation problem by a fully combinatorial algorithm in linear time. Moreover, the construction can be computed within the same time bound using only additions and subtractions. ■

We have shown that we can assert in linear time whether a feasible solution exists if k is fixed. Hence, it is natural to ask whether it is possible to certify this also by a feasible flow.

Theorem 4 *The* TRANSSHIPMENT *problem on a display graph G_n^k can be solved by a fully combinatorial algorithm in $O(n)$ if k is fixed.*

Proof: We consider the same construction of $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ as in the previous proof. We already know that a shortest path computation of linear time tells us whether the instance is feasible or not. Thereby, we can assign a distance label to each vertex of $\mathcal{V}$, say $dist(\cdot)$, which denotes the length of the shortest path from s to this vertex. Hence, if $dist(t) < 0$, then there is no feasible flow as already stated in the previous proof. So, we assume in the following that $dist(t) \geq 0$. We now reduce our instance by fixing the incoming flow of the last node of G_n^k. To this end, we consider the last block B_ℓ, i.e. $n+1 \in B_\ell$. We first fix the flow on the arc $(n, n+1)$. Therefore, let $S \subset B_\ell$ such that $(n, n+1) \in \delta^{out}(S)$ and $\Delta := dist(S) + u(\delta^{out}(S))$ is minimum. Note that we would still get a feasible solution, if we replace $u(n, n+1)$ by $u'(n, n+1) := \max\{0, u(n, n+1) - \Delta\}$. Hence, we may route that amount of flow over the arc $(n, n+1)$. That is, we set $f(n, n+1) = u'(n, n+1)$, modify $d(n)$ and $d(n+1)$ accordingly and set the capacity $u(n, n+1)$ to 0. According to this changes, we also modify the weights of the edges of $\mathcal{G}$ and update the distance labels. This involves a constant amount of work since the changes only affect the last two blocks of $\mathcal{G}$. We repeat this for all incoming arcs of node $n+1$. Since the instance remains feasible, we end up with $0 = u(\delta^{in}(n+1)) \geq -d(n+1) \geq 0$. Hence, for each set $S \subseteq B_\ell$ with $n+1 \in S$, we have $u(\delta^{out}(S)) = u(\delta^{out}(S - \{n+1\}))$ and $d(S) = d(S - \{n+1\})$. Thus, we may delete the node $n+1$ from the display graph and the aforementioned vertices S from $\mathcal{G}$. If for block B_ℓ only the vertex Z_ℓ associated with the empty subset remains, then Z_ℓ becomes the new vertex t and we delete B_ℓ. We perform the same procedure until only one node of the display graph is left. In total we have n iterations each taking $O(1)$ time. Moreover, we use only addition, subtractions, and comparisons. ■

The previous theorem applies not only to display graphs but to all graphs for which the underlying undirected graph has bounded bandwidth by using Lemma 3. Note that it is NP-complete to decide whether the bandwidth of a general graph is bounded by a given constant greater than 2 [27]. So, we require that we are also given a numbering of the nodes with bounded bandwidth. We can drop this condition only for those cases where we can either find a numbering in linear time, e.g. if the bandwidth is not more than 2 as also shown in [27], or if there is a linear time approximation algorithm with a constant factor guarantee, e.g. if we are furthermore restricted to circular-arc graphs [28].

Corollary 4a *The* TRANSSHIPMENT *problem can be solved by a fully combinatorial algorithm in linear time provided that the bandwidth of the underlying undirected graph is bounded and the nodes are numbered accordingly.*

Proof: Let an instance be given by a graph $G' = (V, A')$ with $V = \{1, \ldots, n+1\}$ such that for some constant k and for each arc $(u, v) \in A$ we have $|u - v| \leq k$, a demand function $d : V \to \mathbb{Z}$, and capacities $u : A' \to \mathbb{Z}_{\geq 0}$. By Lemma 3, we know that the underlying undirected graph of G' is embedded in the underlying undirected graph of $G_n^k = (V, A)$. For each arc $(s, t) \in A$ of the display graph that is not present in G', i.e. $(s, t) \notin A'$, we set $u(s, t)$ to 0. If $(s, t) \in A'$ with $s > t$, then we modify $d(s)$ to $d(s) + u(s, t)$ and $d(t)$ to $d(t) - u(s, t)$. Furthermore, we augment the capacity of the reverse arc $u(t, s)$ to $u(t, s) + u(s, t)$. This is equivalent to considering the residual graph of a flow $\bar{f}$ saturating all backward arcs. We call this flow *preflow*. Note that this construction can be done in linear time. Afterwards, we solve the problem on G_n^k in $O(n)$ time and add $\bar{f}$ to this solution. Hence, we get a fully combinatorial algorithm as only additions, subtractions, and comparisons are involved. ■

Furthermore, we can also solve the MINCUT problem on such a graph in linear time by overestimating the value of the minimum cut and applying the separation routine. This yields the minimum cut and hence also value of the maximum flow. Thereby computing the corresponding flow-variables reduces to a TRANSSHIPMENT problem.

Corollary 4b *The* MAXFLOW *problem and the* MINCUT *problem can be solved by a fully combinatorial algorithm in linear time provided that the bandwidth of the underlying undirected graph is bounded and the nodes are numbered accordingly.*

Proof: Let an instance be given by a graph $G = (V, A)$ with $V = \{1, \ldots, n + 1\}$ such that for each arc $(u, v) \in A$ we have $|u - v| \leq k$ for some constant k. Moreover let s, t be the source and the sink, respectively. Clearly, $U := u(\delta^{out}(s))$ is an upper bound on the value of the MAXFLOW. We define the demands $d : V \to \mathbb{Z}$ for a TRANSSHIPMENT problem as follows.

$$d(v) := \begin{cases} U & \text{if } v = s \\ -U & \text{if } v = t \\ 0 & \text{otherwise} \end{cases}$$

Then, we perform the same transformation as in the previous proof to get an instance for G_n^k. We solve the separation problem for this instance yielding $\Delta := dist(t) \leq 0$ and a corresponding cut $X \subset V$. That is, we have found a MINCUT with the value $U + \Delta$. Since this operations only take linear time in total, we proved the part of the claim for the MINCUT. Because of the MAXFLOW/MINCUT theorem, we know also that the value the MAXFLOW is $\varphi := U + \Delta$. We modify the demands accordingly, i.e. we set $d(s)$ to φ and $d(t)$ to $-\varphi$. The solution of this TRANSSHIPMENT instance is feasible and yields the desired MAXFLOW in linear time by the previous corollary. Clearly, the algorithm remains fully combinatorial. ■

We have seen that we can implicitly enumerate the exponentially many constraints in linear time if k is fixed. But the constant scales with 2^k. From a practical point of view, it is advisable to keep this constant as low as possible by dismissing dominated cuts beforehand. That is, e.g., if a general graph is not weakly connected then the problem decomposes into independent subproblems and the cuts on one componet do not interfere with the cuts in the others. We extend this idea and get the following result.

Definition 7 *A cut $X \subset V$ in a weakly connected graph $G = (V, A)$ is called* simple*, if and only if X and $V - X$ are weakly connected in G*

Lemma 4 *If a cut $X \subset V$ is not simple, then the corresponding inequalities $u(\delta^{out}(X)) \geq d(X)$ are dominated.*

Proof: Let $X \subset V$ be a cut that is not simple. Hence, there is a partition of $X = X_1 \dot{\cup} \cdots \dot{\cup} X_p$ with $p > 1$ such that $\delta^{out}(X_i) \cap \delta^{in}(X_j) = \emptyset$ for all distinct i and j as illustrated in the Venn-diagram of Figure 3.5 where the set X is represented by the dashed rectangle, and its complement by the rectangle on the right. The partition is shown by the circles whose leaving arcs have their heads in the complement of X.

Hence, the set of outgoing arcs of X partitions into subsets $\delta^{out}(X_i)$ for $i = 1, \ldots, p$, i.e.

$$u(\delta^{out}(X)) = \sum_{i=1}^{p} u(\delta^{out}(X_i)) \geq \sum_{i=1}^{p} d(X_i) = d(X).$$

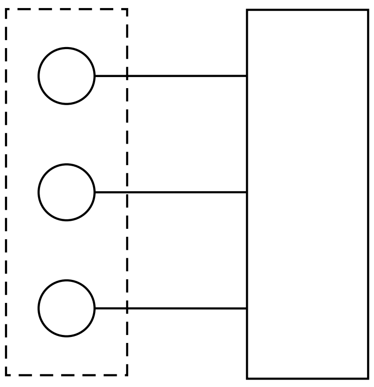


Figure 3.5: Venn-diagram illustrating the mutually disjoint sets

Hence, the inequality associated with the cut X is dominated by the cuts corresponding to the said partition. By taking the partition maximal, we may restrict ourselves to cuts whose vertices are weakly connected in G.

So let X be such a cut with a partition of $V - X = Y_1 \dot{\cup} \cdots \dot{\cup} Y_p$ with $p > 1$ and $\delta^{out}(Y_i) \cap \delta^{in}(Y_j) = \emptyset$ for all distinct i and j. This setting can also be illustrated by the Venn-diagram of Figure 3.5 where the X is the rectangle on the right and the partition of the complement is represented by the circles. Note that then the incoming arcs of the sets Y_i are a partition of the outgoing arcs of X, i.e.

$$u(\delta^{out}(X)) = \sum_{i=1}^{p} u(\delta^{in}(Y_i)) = \sum_{i=1}^{p} u(\delta^{out}(V - Y_i)) \geq \sum_{i=1}^{p} -d(Y_i) = d(X)$$

since we may assume that $d(V) = 0$. It remains to show that the sets $V - Y_i$ are weakly connected in G. This holds since G is weakly connected and since by our assumption there is no connection between Y_i and Y_j within $V - X$ for some i and j. Hence, the inequalities arising from simple cuts define the respective polyhedron. ■

Note that using this result we may prune some of the arcs and eliminate nodes of degree 2 in $\mathcal{G}$ to achieve a speedup in practice. Moreover, for the special case $k = 2$ the number of simple cuts is polynomial in n, in fact, it is in $O(n^2)$.

There is a somewhat general result that implies a linear time algorithm for the MAXFLOW problem on graphs with bounded treewidth in literature [29]. Though the construction of a MAXFLOW instance from a TRANSSHIPMENT problem does not maintain bounded bandwidth and bounded cutwidth in general, because of the unbounded degree of the additional nodes, the pathwidth and treewidth increase by at most 2. To see this, we consider a tree decomposition $T = (V_T, E_T)$ of the original graph, and add the supersource s and the supersink t to every V_i for all $i \in V_T$ while E_T remains unchanged. This yields a tree decomposition for the modified graph as the new nodes clearly belong to at least one V_i for some $i \in V_T$, and the additional edges $\{s, v\}$ and $\{v, t\}$ for each original node v are contained in some V_i since v belongs to at least one V_i, and for each $u, v, w \in V_t$ such that v lies on a path from u to w in T, the inclusion $V_u \cap V_w \subseteq V_v$ is maintained since the new nodes s and t are contained in each of the three sets V_u, V_v, V_w.

However, the authors of [29] use the heavy machinery of *integer linear programming in fixed dimension* in their arguments. On this note, our results of this subsection can be considered as a refinement to achieve fully combinatorial algorithms for TRANSSHIPMENT, MAXFLOW, and MINCUT on graphs with bounded bandwidth as desired for our application.

We have seen in this section how to derive the linear projection of the polyhedron of our integer linear program onto the space of the capacity variables using the MAXFLOW/MINCUT theorem, efficient ways to solve the separation problem for the this projection, and provided a feasible solution for this projection how to lift back such a solution to the space of the original variables. In the next section, we will analyze the projection and relate it to the field of network design problems.

3.5 The Network Design Problem

In the previous section, we have seen that a set of capacities is feasible, if and only if the cut inequalities hold. Hence, the set of feasible capacities is determined by a polyhedron. This yields an alternative formulation for the CMLA problem only containing the capacities as follows.

$$\begin{aligned} \min\ & \sum_{a \in A} u(a) \\ \text{s.t.}\ & u(\delta^{out}(X)) \geq d_j(X) && \text{for all } X \subset V, \text{ for all } j \\ & u \in \mathbb{Z}_{\geq 0}^{|A|} \end{aligned}$$

Such cut covering problems belong to the class of network design problems. Prominent members of this class are the *Directed Steiner Network* (DSN) problem and the *Directed Steiner Tree* (DST) problem. In fact, there is a connection between CMLA for monochrome images and the DSN problem, which we will investigate in the next subsection to gain insights into complexity of CMLA.

3.5.1 Monochrome images and Directed Steiner Networks

A *monochrome image* is an image $R \in \{0,1\}^{n \times m}$. That is, we consider diodes of a certain color that are either on (1) or off (0) without a gray-scale inbetween. Hence, we consider demands with $d_j(i) \in \{-1, 0, 1\}$. The following example shows the transformation of such an image into the demand matrix by subtracting from each row its predecessor.

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 \end{pmatrix} \tag{3.7}$$

Note that each column is a $0, \pm 1$-vector. Furthermore it is easy to see that the occurrences of 1 and -1 in each column alternate and that each 1 is succeeded by a -1 and each -1 is preceded by a 1 disregarding the zeros inbetween. Moreover, the two matrices on the right of (3.7) have the same set of feasible capacities which are subsets of arcs such that the pairs of nodes $(1,2)$, $(3,4)$, $(2,4)$, and $(1,4)$ are connected in the corresponding subgraph. Therefore, we assume w.l.o.g. that each column yields exactly one such pair of nodes to which we also refer as a *commodity* in the following. In general, the problem of optimally decomposing monochrome images can be understood as follows.

Problem 4 *Given commodities (s_j, t_j), $j = 1, \ldots, m$ with $s_j < t_j$ for each j and a display graph G_n^k, select a minimum number of arcs of G such that there exists a path from s_j to t_j for each $j = 1, \ldots, m$.*

The nodes s_j are called *sources* and the nodes t_j are called *sinks*. Furthermore, if a node is neither a source nor a sink, we call it *Steiner*. The selection of arcs of G is given by a function $u : A \to \{0,1\}$, where $u(a) = 1$ if the arc a is selected and $u(a) = 0$ otherwise. This problem is a restriction of the well-known DirectedSteinerNetwork (DSN) problem to display graphs.

Problem 5 (DSN) *Given a directed graph $G = (V, A)$, a length function $\ell : A \to \mathbb{Z}_{\geq 0}$, and a set of commodities $D = \{(s_1, t_1), \ldots, (s_m, t_m)\}$, determine the shortest set $N \subseteq A$ with respect to the length function ℓ such that s_j is connected to t_j in the graph $H = (V, N)$ for all $j = 1, \ldots, m$.*

By fixing a node r and setting all $s_j = r$, we get the DirectedSteinerTree (DST) problem. Note that DST is NP-complete, even if restricted to acyclic graphs and uniform objective, i.e. $\ell(a) = 1$ for all $a \in A$, by the following reduction to SetCover (problem SP5 in [18]). Given an universe U and a set system $\mathcal{S} \subseteq 2^U$, we construct an instance of DST with a directed acyclic graph as follows. The vertex set consists of a root node r, nodes for each set $S \in \mathcal{S}$, and nodes for each $e \in U$ which we consider as terminals. The root r is connected to every node $S \in \mathcal{S}$ with an arc pointing away from r. Each vertex $S \in \mathcal{S}$ has arcs from S to each $e \in S$. This construction is illustrated in Figure 3.6.

Since in a feasible solution of this DST problem each terminal is connected to r by a path via a node S, we immediately have a SetCover of the universe U by some sets of $\mathcal{S}$. If W is the number of arcs of an optimal solution to the DST problem, then $W - |U|$ is the minimum number of sets needed to cover U.

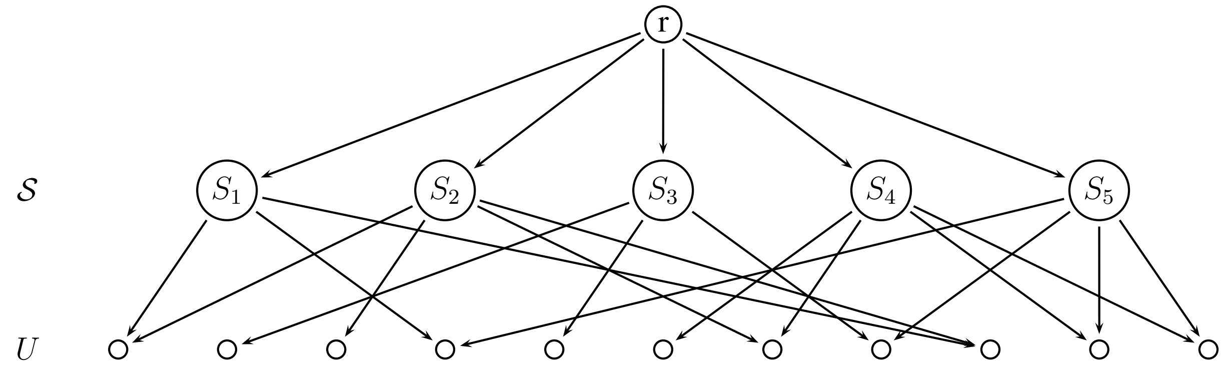


Figure 3.6: The construction of a DST problem from a SETCOVER instance

It is natural to ask whether the problem remains NP-complete for G_n^n, i.e. the complete directed acyclic graph. We will consider the weighted case first. In fact, if we use a complete directed acyclic graph in the same way as in the previous reduction and give every arc connecting an element with its set weight 0 and all other arcs weight 1, then a DST solution yields a SETCOVER with the same weight. So it is crucial to consider the uniform case where we get the following result.

Theorem 5 *Given a set D of m commodities (s_j, t_j) with $s_j < t_j$ for each $j = 1, \dots, m$, the DSN problem can be solved in polynomial time on a unweighted complete directed acyclic graph.*

Proof: Since the underlying graph is complete, directed, and acyclic, there is a unique topological order of the nodes. Moreover, only the portion between the first and the last terminal is relevant. Because of the uniform weight of the arcs, it is sufficient to consider only the terminal nodes. So, let

$$V := \{s_1, \dots, s_m\} \cup \{t_1, \dots, t_m\}$$

Furthermore, let E be the set of undirected edges corresponding to D. By a breadth-first search, we identify the connected components, say $K_1, \dots, K_c \subseteq V$, of the graph (V, E). Note all of these components are spanned by c disjoint directed paths in the underlying complete graph. Moreover, each commodity $(s_j, t_j) \in D$ is connected since s_j and t_j lie in the same component. Let A be the union of the arcs of those paths, then $|A| = |V| - c$.

Assume that there is a set A' with $|A'| < |V| - c$ such that each commodity is connected. Then, the number of weakly connected components with respect to A' is greater than c. But this implies that there is a commodity, say (s, t), such that s and t lie in different components and hence A' is not a feasible solution which proves the optimality of A. Furthermore, all the mentioned operations can be carried out in $O(m)$ time. ■

So we may always chose the optimum solution in such a way that it consists of disjoint paths. Or in other words, we may chose among all optimal solutions the one for which the indegrees and outdegrees are bounded by 1. It is straight forward that this also holds for partially complete acyclic graphs.

Lemma 5 *Given a partially complete acyclic graph $G = (V, A)$ and capacities $u : A \to \{0, 1\}$ which are feasible for an arbitrary DSN instance on G, then there exists a feasible solution u' with the at most the same total weight, i.e. $u'(A) \leq u(A)$ and*

$$u'(\delta^{out}(i)) \leq 1 \qquad \text{as well as} \qquad u'(\delta^{in}(i)) \leq 1 \tag{3.8}$$

for all nodes $i \in V$.

Proof: We may assume w.l.o.g. that G is connected as the problem would decompose into independent subproblems otherwise. Since the graph is acyclic and partially complete, it contains a Hamiltonian path. Hence, there is a unique topological order of V. Let $r \in V$ be the first node with $u(\delta^{out}(r)) > 1$. Let

$$N := \{i \in V : (r, i) \in A \wedge u(r, i) > 0\}$$

and s be the smallest node in N. We define $u'(r, s) := 1$ and for all $t \in N - \{s\}$ we set $u'(r, t) := 0$ and $u'(s, t) := 1$ as shown in Figure 3.7.

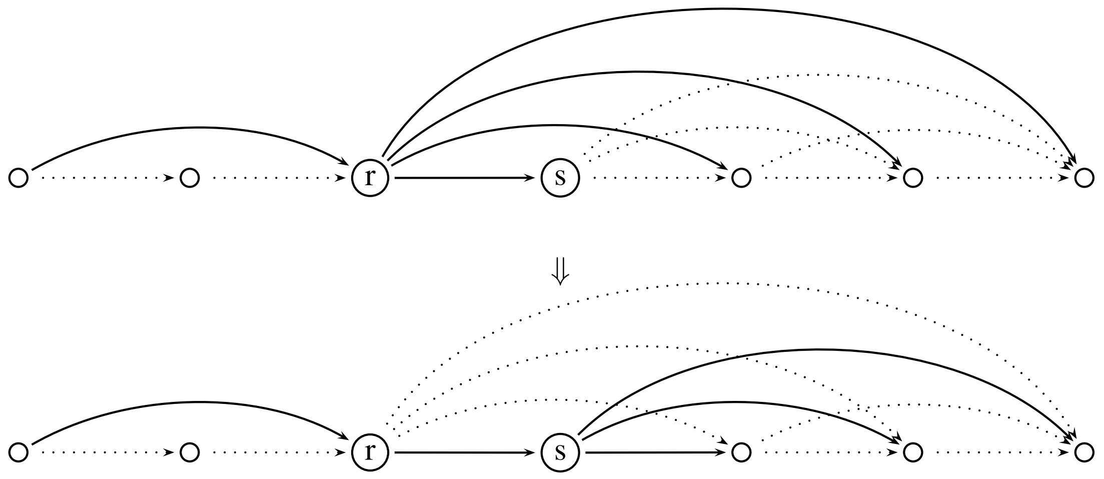


Figure 3.7: Reducing the outdegree while maintaining the connectivity.

For all other arcs $(i, j) \in A$, the capacities remain the same, i.e. $u'(i, j) := u(i, j)$. Clearly, $u'(A) \leq u(A)$ and all connectivities are preserved. Note that by this construction $u'(\delta^{out}(r)) \leq 1$ holds. We repeat this procedure until $u(\delta^{out}(i)) \leq 1$ holds for all nodes $i \in V$. We apply it analogously for the incoming arcs of all nodes in reverse order. The whole procedure terminates in $O(|V|)$ steps since no outdegree is increased by the modifications of the indegrees. ■

We call the Inequalities (3.8) *degree condition*. We will generalize them later to unbounded demands. It is easy to see that we may simplify each instance such that the degree condition holds. That is, the subgraph induced by the commodities is already a collection of disjoint paths. The following Lemma states that we can achieve this in linear time.

Lemma 6 *Given a (partially) complete acyclic graph $G = (V, A)$ and set D of commodities. We can find in $O(|D|)$ time an alternative set of commodities D' consisting only of disjoint paths such that all solutions to the new DSN instance are also feasible for the original one and the optimum objective value remains unchanged.*

Proof: As in the proof of Theorem 5, we can perform a breadth-first search on the undirected graph H induced by D to identify the connected components. This takes $O(|D|)$ time as it is sufficient to consider only the nodes contained in D. The new set of commodities D' is then union of the arcs corresponding to the paths spanning each connected component. ■

In [8], we showed that the *Doubleline Addressing* problem for monochrome images is solvable in polynomial time. In fact, we presented an algorithm that runs in linear time if the input is given in the matrix representation for CMLA. However, it has quadratic running time, if the commodities are represented as pairs. We can improve this time bound to a linear one as follows.

Theorem 6 *Given a display graph $G_n^2 = (V, A)$ and a set of m commodities (s_j, t_j) for $j = 1, \ldots, m$ then the minimum cardinality subset of A that connects all commodities can be computed in $O(n + m)$.*

Proof: It is easy to see that a display graph satisfies the conditions of the previous lemma. Hence, we may assume w.l.o.g. that every node is the source of at most one commodity and the sink of at most one commodity. We will follow a dynamic programming approach to show the desired running time bound. For node i, we call a commodity (s, t) *open*, if $s < i \leq t$. Moreover, we maintain for each node a list L of open commodities in semi-circular order with respect to the source node. For node i this list is ordered as follows: Commodities out of $i - 2, i - 4, \ldots$, commodities out of $i - 3, i - 1$. Being at node i, there are two choices for each open commodity: Either it passes through node i or is routed through arc $(i - 1, i + 1)$ as depicted in Figure 3.8.

Since we are only interested in solutions that form disjoint paths, we consider only such solutions as feasible. Hence, routing a commodity through $(i - 1, i + 1)$ implies that all commodities following it in L have to use this arc, too. Conversely, if a source is connected to i then all preceding sources in L are also connected to i. Hence, each contiguous partition of L into two segments defines a partial solution where the outgoing arcs of the nodes $< i$ are already selected.

Note that the optimal solution has an objective value less than the number of nodes. Moreover, the cost of a partial solution at node i is always less or equal than i. In fact, the cost is i minus the number of nodes not having an outgoing arc.

We define a state as a pair of a pointer into L denoting a partition (L_1, L_2) and the number of nodes before i not having an outgoing arc. Note that the arc $(i - 1, i + 1)$ is included in the partial solution if and only if L_2 is non-empty.

At the beginning, L is empty and there is one state with value 0.

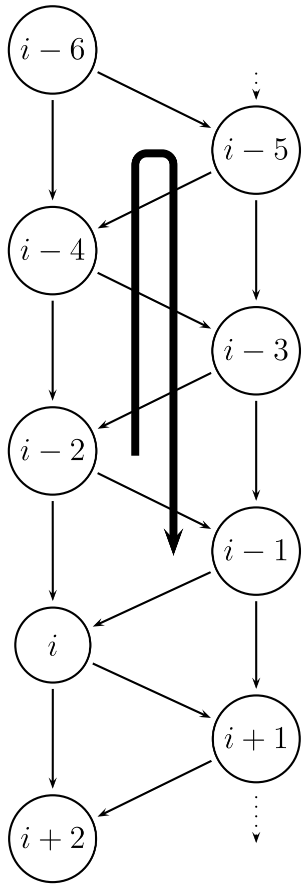


Figure 3.8: Semi-circular ordering of the open commodities in G_n^2. The drawing is stretched in its width for better visualization.

The update step consists of two phases. In the first phase, we delete all partial solutions that turn out to be infeasible. If node i is target of a commodity, say (s, i), we declare all solutions infeasible that do not connect s to i. This means, we delete all states where s belongs to L_2. Afterwards, we remove the commodity (s, i) from L. Note that this was the only commodity ending in i by our assumption. Since there are $O(n)$ states in total and each state is deleted only once, the total cost of this phase is in $O(n)$.

In the second phase, we distinguish two cases. If there is a commodity starting at i then we insert it at the beginning of L and get the new list L'. Moreover, we create a new state corresponding to the partition $(\emptyset, L')$ which means that we use arc $(i, i+1)$ as L' is prepared for proceeding to node $i+1$ where we consider the open commodities in reversed order. Since this new state uses an additional arc, it gets same value as the state $(L, \emptyset)$.

If no commodity starts at node i, the list L remains unchanged. If the state corresponding to $(\emptyset, L)$ still exists, then it is possible to route all remaining open commodities through the arc $(i-1, i+1)$. Hence there is no need for an outgoing arc of node i in this state. Thus, we increment the value of this state by one.

The cost of the second phase in one step is clearly in $O(1)$. Hence, the total cost of the entire update step remains in $O(n)$. Because of the preprocessing step, we get a total running time of $O(n+m)$. At the end there is only one remaining state. The objective value of the corresponding solution is given by n minus its value. We determine the selection of the arcs by tracing back the execution of the dynamic program. ■

We conjecture that the DSN problem is solvable in polynomial time for display graphs G_n^k because of their special structure allowing the restriction to solutions forming disjoint paths. Anyway, we are more interested in the case beyond $\{-1, 0, 1\}$ demands.

3.5.2 Unbounded demands

Unfortunately, the previous algorithm for the case $k = 2$ does not generalize to unbounded demands. So, we recall the formulation of our problem as an integer linear program containing only the capacities.

$$\begin{aligned} \min \quad & \sum_{a \in A} u(a) & \\ \text{s.t.} \quad & u(\delta^{out}(X)) \geq d_j(X) & \text{for all } X \subset V, \text{ for all } j \\ & u \in \mathbb{Z}_{\geq 0}^{|A|} & \end{aligned} \tag{3.9}$$

By our separation routine and the ellipsoid method [19], we can solve the LP relaxation in polynomial time. Since we have a combinatorial problem here, we may even solve the LP in strongly polynomial time by a result of Tardos [30].

Given such a (fractional) solution of the LP, we immediately get an approximative solution to the integer program by rounding up each fractional capacity. Though, we are within an additive error not greater than the number of variables then, we could round down instead of rounding up and solve or approximate the remaining 0/1 integer program with a possibly better solution for the whole problem.

One approach for getting an approximation is the so-called *separation and augmentation* idea. That is, we maintain an assignment to the capacities, iteratively solve the separation problem, and augment one or more capacities that are contained in a violated cut. We repeat this procedure until the assignment gets feasible. We elaborate more on this procedure in the next chapter.

However, as mentioned before solving (or approximating) linear programs involving fractional numbers is not really what we want. Of course, we can initialize all the capacities with zeros and augment by integers only, e.g. incrementing only one capacity by 1 to avoid overshooting the mark too far. But this means that we need as many iterations as the final objective value. It is natural to ask whether there is a way inbetween that allows us to stick with integers and to use a fully combinatorial algorithm.

Indeed, if we follow an idea known from *Lagrangian duality* and restrict ourselves to *easy* constraints, i.e. constraints describing an integer polyhedron, we accomplish the first goal of remaining integer. If we only consider constraints permitting fully combinatorial algorithms (e.g. flow problems or their duals) then we could also achieve the second goal.

Having solved such an easy subproblem, we can initialize our heuristics with the thereby computed solution. The next theorem identifies such an easy subset of the constraints.

Theorem 7 *The linear program* (3.9) *restricted to the set system*

$$\mathcal{C} = \bigcup_{i=1}^{n} \{X_i, Y_i, X_i \cap Y_i, X_i \cup Y_i : X_i = \{1, \dots, i\}, Y_i = \{i, i+2, i+3, \dots, n+1\}\}$$

has an integer optimal solution for $k = 2$ which can be found in linear time by a fully combinatorial algorithm.

Proof: We order the variables $u_i^{(\alpha)}$ lexicographically, that is with respect to tail and head of the corresponding arcs. The constraints corresponding to the cuts of $\mathcal{C}$ are

$$\begin{aligned}
u(\delta^{out}(X_i)) &= u_{i-1}^{(2)} + u_i^{(1)} + u_i^{(2)} \\
u(\delta^{out}(X_i \cup Y_i)) &= u_{i-1}^{(2)} + u_i^{(1)} \\
u(\delta^{out}(X_i \cap Y_i)) &= u_i^{(1)} + u_i^{(2)} \\
u(\delta^{out}(Y_i)) &= u_i^{(1)}
\end{aligned}$$

Hence, the constraint matrix has the consecutive ones property in every row. Since our objective is the all-ones vector, we can solve the problem by a shortest path computation in an directed acyclic graph of $O(n)$ nodes and $|\mathcal{C}| = O(n)$ edges. To this end, we apply a unimodular transformation to the variables. That is, we define a new set of variables y as follows.

$$\begin{aligned}
y(0) &:= 0 \\
y(1) &:= u_1^{(1)} \\
y(2i) &:= u_i^{(2)} - u_i^{(1)} && \text{for } i = 1, \dots, n-1 \\
y(2i+1) &:= u_{i+1}^{(1)} - u_i^{(2)} && \text{for } i = 1, \dots, n-1
\end{aligned}$$

With the new variables, we get $\geq$ constraints containing exactly one y-variable with coefficient 1 and one with -1 in each row. We may solve such a system of difference constraints by a shortest path computation (cf. section 25.5 of [31]). The costs of the edges are given by the negated right-hand sides of the respective constraints. Clearly, there is only one dominating right-hand side per constraint among all $j = 1, \dots, m$. It is easy to see that we can provide the costs for the edges in a preprocessing step taking $O(n \cdot m)$ time which is linear in the size of the demand matrix. The costs of the shortest path from the first to the last node yield the negated optimal objective value subject to the subsystem $\mathcal{C}$. The corresponding capacities can be computed from the difference of the distance labels of the nodes by reversing the transformation. All these operations take $O(n \cdot m)$ time in total. ■

Note that this consecutive ones property does not hold for $k > 2$. However, we can restore it by adding the missing capacities on the left-hand side of the inequalities and the corresponding lower bounds on the right-hand side giving (weaker) valid inequalities yielding an approximate solution.

Moreover, the previous Theorem gives us a lower bound on the objective value. But the variables of such a solution are not lower bounds for the variables with respect to the complete set of cut inequalities. For this purpose, it is only save to use the constraints of the form $u_i^{(1)} \geq d_j(Y_i)$.

Note that these statements hold for arbitrary k. This is due to the fact that those constraints by design contain only variables of arcs that are close to each other in the canonical order. Recall that, for the the linear time separation, we enforced and exploited such a locality by bounding k and by partitioning the nodes into blocks of size k.

In the following, we will again consider the situation if k is a constant and extend the segmentation idea. Thereby, we gain a lot of freedom for the algorithms operating on the segments of constant size though we still get a linear timebound for the complete instance. We first demonstrate this idea with the following simple result yielding a polynomial time factor 2-approximation if k is fixed. Afterwards, we extend this approach to a PTAS.

Theorem 8 *Given a display graph $G_n^k = (V, A)$ where k is a constant, and demands $d_j : V \to \mathbb{Z}$ for $j = 1, \dots, m$, there is a polynomial time approximation algorithm with approximation factor* 2.

Proof: We divide the display graph G_n^k into segments of the form G_k^k such that last node of the first segment coincides with the first node of the second segment, and so on. We call these common nodes *breakpoints* (see Figure 3.9). Moreover, we omit all arcs that point from one segment to the next one.

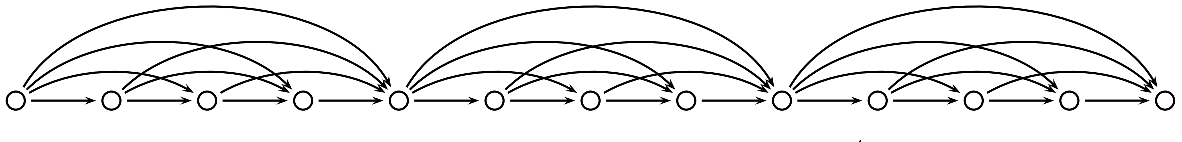


Figure 3.9: The nodes 5 and 9 are the breakpoints when splitting G_{12}^4 into segments of size 4.

Since there are no arcs that skip one or more segments, all those remaining arcs are within their respective segments. Let G' be the first segment. We define for each $j = 1, \dots, m$

$$d'_j(i) = \begin{cases} d_j(i) & \text{for } i = 1, \dots, k \\ d_j(\{k+1, \dots, n+1\}) & \text{for } i = k+1 \end{cases}$$

which means that we contract the remainder of the display graph to the node $k+1$. Since the network design problem with respect to G' has a constant number of variables, we may solve the corresponding ILP in polynomial time [32]. Afterwards, we fix the capacities of this part and continue with the next segment. Thereby we end up with a feasible solution in polynomial time. Since the partial solutions do not influence each other, we have found an optimal solution to the problem without the omitted arcs. Note that for each omitted arc (u, v) there remain two arcs, say (u, w) and (w, v) where w is the breakpoint of the two segments. Hence, we are within two times the optimal solution for the original problem. ■

We now extend this idea towards a polynomial time approximation scheme (PTAS). That is for every fixed $\varepsilon > 0$, we have an algorithm that runs in polynomial time and achieves an $(1+\varepsilon)$-approximation.

The obvious idea is to increase the size of the subproblems depending on the constant ε. Though this approach yields a polynomial time algorithm, it is easy to construct instances such that the optimum objective value is dominated by the omitted arcs and hence the $(1+\varepsilon)$-approximation guarantee does not hold. This means that if we fix the breakpoints in advance, say $s+1$, $2s+1$, $3s+1$, and so on, then we may construct a bad instance with $d_j(h \cdot s) = M$ and $d_j(h \cdot s+2) = -M$ for every natural number h such that $h \cdot s + 2 \in V$. For a sufficiently large M, the approximation ratio exceeds the $(1+\varepsilon)$ bound for every fixed ε.

We may circumvent this obstacle by dynamically breaking the graph into segments depending on the instance. So, we have to identify good breakpoints. We will show that it is sufficient to keep the size of the segments constant (except the first and the last one which may be smaller). Hence, the breakpoints are determined by the first one. That is, if i is the first breakpoint and s is the size of the segments, then the remaining breakpoints are $i+s$, $i+2s$,$i+3s$, and so on. Before we prove this claim, we first generalize the degree condition to unbounded demands. Recall that $r_{ij} = d_j(\{1, \ldots, i\})$ and let $\overline{r_i} := \max\{r_{ij} : j = 1, \ldots, m\}$.

Lemma 7 *Given a feasible solution u, then there exists a feasible solution u' with*

$$u'(\delta^{out}(i)) \leq \overline{r_i} \qquad \textit{and}$$
$$u'(\delta^{in}(i)) \leq \overline{r_{i-1}}$$

with at most the same total weight.

Proof: Recall that $f_j(\delta^{out}(\{1, \ldots, i\})) = r_{ij}$. Hence, $f_j(\delta^{out}(i)) \leq r_{ij}$ because of the non-negativity of the flow-variables. Analogously,

$$f_j(\delta^{in}(\{i, \ldots, n+1\})) = f_j(\delta^{out}(\{1, \ldots, i-1\})) = r_{i-1,j}$$

and hence $f_j(\delta^{in}(i)) \leq r_{i-1,j}$. W.l.o.g. we may assume that we have $u(i, i') \leq \overline{r_i}$ for each arc $(i, i') \in A$ since we may always choose the capacities in such a way that they are tight to the flow for at least one j. Let $i \in V$ such that $u(\delta^{out}(i)) > \overline{r_i}$. Moreover, let $(i,t) \in \delta^{out}(i)$ with t maximal such that

$$\sum_{i'=i+1}^{t} u(i,i') \leq \overline{r_i}.$$

Then we set $u'(i,i') := u(i,i')$ for all arcs $(i,i') \in \delta^{out}(i)$ with $i' < t$. Furthermore, we set

$$u'(i,t) := \overline{r_i} - \sum_{i'=i+1}^{t-1} u(i,i')$$

and $u(i,i') := 0$ for all arcs $(i,i') \in \delta^{out}(i)$ with $i' > t$. All other capacities remain the same. This transformation tightly fullfills the degree condition for the outgoing arcs of i. To restore the connectivity of the new capacities, we distribute the old capacities of the outgoing arcs (i,i') with $i' \geq t$ as follows. Let N be a multiset corresponding to the neighborhood of the outgoing arcs of i according to their capacities, i.e.

$$N := \{i' \in V : (i,i') \in A, u'(i,i') > 0\}$$

where the multiplicity of every element i' is determined by $u'(i,i')$ and hence the cardinality of N is $\overline{r_i}$. If $u(i,t) > u'(i,t)$, say $u(i,t) = u'(i,t) + \Delta$, then we pick Δ elements of N, say $v_1, \ldots, v_\Delta$, and increment $u'(v_h, t)$ by 1 for each $h = 1, \ldots, \Delta$. Note that it is always possible to pick Δ elements of N since $\Delta \leq \overline{r_i}$. Moreover, the flow of the arc (i,t) in each transshipment with respect to the capacities u can be rerouted since the flow through node i is also bounded by $\overline{r_i}$. Hence, we may apply this procedure to every arc (i,i') with $i' > t$ and restore the connectivity. Moreover, we do not change the total weight of the capacities. By repeating this for every node from top to bottom, we end up with $u'(\delta^{out}(i)) \leq \overline{r_i}$ for every $i \in V$. Note that $u'(\delta^{in}(i)) = u(\delta^{in}(i))$ and hence we may apply the same procedure analogously for the incoming arcs of every node from bottom to top which proves the claim. ■

Note that all operations in the previous proof can be carried out in polynomial time.

Theorem 9 *There is a polynomial time approximation scheme for any display graph $G_n^k = (V,A)$ where k is a constant.*

Proof: Let $s \geq k$ be the size of the segments in which we partition the instance (except for the first and the last one which may be smaller). As mentioned before, we have to dynamically define our breakpoints. That is, we consider each node in $\{1, \ldots, s\}$ to be the first breakpoint,i.e. if i is the first breakpoint then the following ones are $i+s$, $i+2s$, $i+3s$, and so on. We first establish an upper bound on the additional costs that we have to pay at one breakpoint. Let node i be a breakpoint and $\bar{u}$ be an optimal integer solution. By breaking at node i, we omit the arcs $B := \delta^{out}(\{1, \ldots, i\}) - \delta^{out}(i)$. Because of the degree condition, we may assume

$$\bar{u}(B) \leq \sum_{i'=i-k+1}^{i-1} \bar{u}(\delta^{out}(i)) \leq \sum_{i'=i-k+1}^{i-1} \overline{r_{i'}}$$

which is also an upper bound on the additional costs that may occur by breaking those arcs. Let $cost(i)$ denote the additional costs that are caused by breaking at the nodes $1 + i + h \cdot s$ for $h = 0, \ldots, H$ where H is the maximum such that $1 + i + H \cdot s \in V$. We added the shift by 1 for the ease of notation since breaking at node 1 yields the same result as breaking first at node $s + 1$ but the breakpoint at node 1 does not contribute to the additional costs. Hence

$$cost(i) \leq \sum_{h=0}^{H} \sum_{i'=i-k+2}^{i} \overline{r_{i'+h\cdot s}}$$

It is easy to see that $\frac{1}{k}\sum_{i=1}^{n} \overline{r_i}$ is a lower bound on the optimum objective value, say OPT.

Hence,

$$\begin{aligned} OPT &\geq \frac{1}{k}\sum_{i=1}^{n}\overline{r_i} \\ &\geq \frac{1}{k}\sum_{i=1}^{s}\sum_{h=0}^{H}\overline{r_{i+h\cdot s}} \\ &= \frac{1}{k(k-1)}\sum_{i=1}^{s}\sum_{h=0}^{H}\sum_{i'=i-k+2}^{i}\overline{r_{i'+h\cdot s}} \\ &\geq \frac{1}{k(k-1)}\sum_{i=1}^{s} cost(i) \\ &\geq \frac{s}{k(k-1)}\underbrace{\min\{cost(i) : i = 1, \ldots, s\}}_{mincost}. \end{aligned}$$

By solving the problem for all s choices for the first breakpoint and selecting the solution with the minimum total objective value, say APX, we get the one with the minimum additional cost, i.e.

$$APX = OPT + mincost \leq (1 + \frac{k(k-1)}{s}) \cdot OPT.$$

Thus, for any arbitrary but fixed $\varepsilon > 0$, there is an integer constant s such that $APX \leq (1 + \varepsilon) \cdot OPT$ which proves the claim. ■

The algorithms presented in this section have the following key property in common which is due to the special structure of the display graph. They use only local information to take their decisions. That is in every step only a segment of constant size of the display graph is considered. Thereby, all those algorithms have a running time which is linear in the image size and polynomially bounded in the encoding length of the image data. This is a crucial improvement over the separation and augmentation approach since there the running time depends on the objective value which is not polynomially bounded by the bitsize of the input numbers in general.

In practice we want to look at each of the $n \cdot m$ pixels as few times as possible. Since we do not expect that we can solve the CMLA problem exactly in $O(n \cdot m)$ time, we leave the complexity status open for CMLA with unbounded demands and focus our research on approximation algorithms that achieve that time bound by design. To this end, we establish the following paradigm.

> The algorithm shall pass only once over the input for computing feasible capacities.

We will discuss and evaluate its realization in Section 4.3. We will see there that a combination of the ideas of this section will yield a competitive algorithm that outperforms the separation and augmentation approach by several orders of magnitudes with respect to running time at almost equal reduction ratios for real world images. Moreover, the theoretical investigation of the related online problem that we will discuss in the next section will lead to furher improvements on the cases where the first implementation of the new algorithm fell short compared to the segmentation and augmentation approach.

3.6 The Online Problem

In this section, we investigate a setting that forces a single pass algorithm by design. That is, we restrict the look-ahead to a certain number of rows. What follows is an online version of our problem where we have to fix the capacities of the outgoing arcs of a node only based on the knowledge of the following c rows. For the theoretical analysis, we will consider the case $k = 2$ and monochrome images. At the end of this section, we will elaborate in particular on the look-ahead of 3 rows as it is used in the practical implementation.

The canonical online algorithm uses the exact algorithm of Section 3.5.1 as follows. We solve the instance of the known c rows to optimality. According to that solution, we fix the outgoing arcs of the first node. After updating the instance and reading the next row, we repeat these steps until we reach the end.

In the following, we will first give a lower bound on the competitive ratio of any algorithm in that online setting. The competitive ratio of an online algorithm is the worst-case performance compared to an offline algorithm. That is the quotient of the objective value achieved by the online algorithm and the optimum solution of an exact offline algorithm. Furthermore, we will analyze the competitive ratio of the aforementioned approach. Before we state the theorem, it is helpful to have a look at following example where the adversary starts with the image in the middle and then reveals the fourth row according to the arc we have selected for the first node.

$$\begin{pmatrix} \cancel{1} & \cancel{0} & \cancel{0} \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \xleftarrow{\text{singleline}} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ \star & \star & \star \end{pmatrix} \xrightarrow{\text{doubleline}} \begin{pmatrix} \cancel{1} & \cancel{0} & \cancel{0} \\ \cancel{1} & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

The optimal value in both cases is 3. But after making the choice for the first row, the adversary forces us to pay 4.

Theorem 10 *Any online algorithm that fixes the outgoing arcs of node i without knowing the rows $i + c$ and beyond, has a competitive ratio of at least $\frac{c+1}{c}$.*

Proof: An adversary reveals the first c nodes of an instance with the commodities (i, t_i) for all $i = 1, \ldots, c$ where $t_i \in \{c + 1, c + 2\}$ is chosen later depending on the arc the algorithm picks following the idea shown in (3.10). If the algorithm selects the singleline arc, then the adversary sets t_1 to the odd node. Otherwise, it is set to the even node. All other t_i are set such that the commodities (i, t_i) are even. The optimal solution of the residual problem is c. Hence, the achieved objective value is $c + 1$ whereas the opposite choice for the first arc would yield an optimal solution of value c. ■

Theorem 11 *There is an online algorithm with competitive ratio $\frac{c+1}{c}$.*

Proof: We first compare the optimal algorithm running on the complete instance with the one running on the first c rows. Let t be the first node without a selected outgoing arc in an optimal solution that contains the doubleline arc leaving the first node. If no such node exists, then in every feasible solution the singleline arc has to be picked for leaving node 1. This also holds for the instance restricted to the first c nodes. Note that the choice of the first arc only depends on the first t rows. Hence, if $t \leq c$, then the online algorithm makes the same decision on the outgoing arcs of the first node as the optimal one. Otherwise, if $t > c$, the restricted algorithm uses t singleline arcs whereas $t - 1$ arcs are used in this part of the optimal solution. This yields a ratio of

$$\frac{t}{t-1} = 1 + \frac{1}{t-1} \leq 1 + \frac{1}{c}$$

since $t - 1 \geq c$. Note that the remainder of the optimal solution is independent from the previous part. Hence, we can partition the optimal solution accordingly, say $OPT = \sum_{i=1}^{p} OPT_i$, and repeat the same argument for each part. This yields

$$APX \leq \sum_{i=1}^{p} \frac{c+1}{c} OPT_i = \frac{c+1}{c} OPT$$

which proofs the claim. ■

3.6.1 A compact $4/3$-approximation

In this subsection, we unroll the generic algorithm for the case $c = 3$ and give a compact set of rules for the selection of the capacities. The choice of $c = 3$ is of particular interest since we have used it in the implementation of the single pass paradigm. These rules can be generalized and integrated into the said implementation to decompose colored images, i.e. instances with unbounded demands. In Section 4.3, we will see that these rules will improve the reduction ratio in practice. They are described as follows and depicted in Figure 3.10.

mlacompact We consider the first three nodes. If the first node is not a source, we skip it without selecting any outgoing arc. Assume it is a source in the following. If the corresponding sink is node 2, i.e. we have the commodity $(1, 2)$ as depicted in Figure 3.10, we select the singleline arc. If node 2 is a *Steiner node* (Figure 3.10b), i.e. it is neither a source nor a sink, then we select the doubleline arc. If node 2 is a source and node 3 is either the corresponding sink or Steiner (Figure 3.10c/d), we select the singleline arc. Otherwise, we select the doubleline arc (Figure 3.10e).

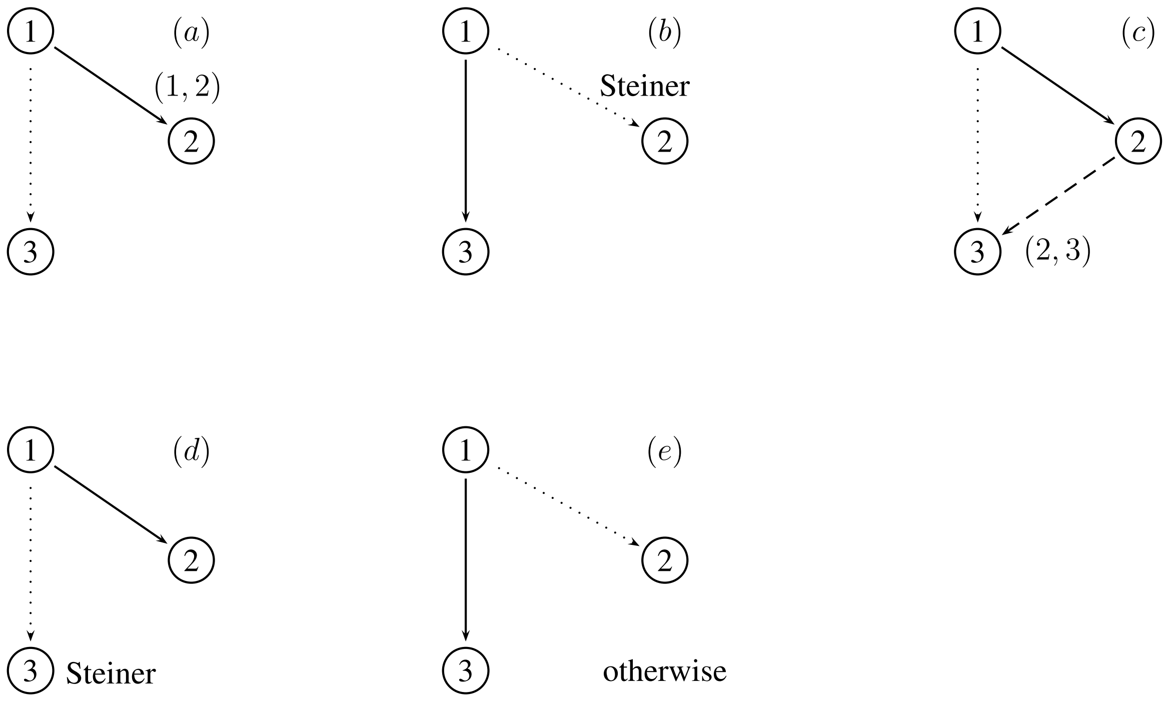


Figure 3.10: The rules for the algorithm `mlacompact`

Lemma 8 *The algorithm* `mlacompact` *achieves a competitive ratio of* $4/3$.

Proof: The interesting cases are if node 1 and 2 are sources and their corresponding sinks are not revealed yet. Assume first that node 3 is Steiner. We show that we can transform every feasible solution such that the Steiner node is isolated (see Figure 3.11). Consider a feasible solution where the arc between node 1 and 3 is picked. Since node 2 is a source the arc between 2 and 4 is also selected. Moreover there must be an arc between node 3 and 5. We can reconnect the tail of latter to node 4 and the head of the outgoing arc of node 1 to node 2. Thereby, we do not change the number of arcs and the routing remains feasible. If node 3 is a source instead, the demand of node 1 may go piggyback with the demand of node 2 or with the one of node 3. Since the first three nodes are sources, each of them has an outgoing arc in every feasible solution. If an adversary reveals that our decision to take the doubleline arc for node 1 was wrong, we need one surplus arc to fix it. Until the adversary does not force us to change the parity, i.e. choose an singleline arc, we do not use more arcs than optimal. Moreover, if we are forced to take such an arc, the problem decomposes into independent subproblems. Thereby, we use at most one surplus arc by three necessary ones and hence get a ratio of $4/3$. ■

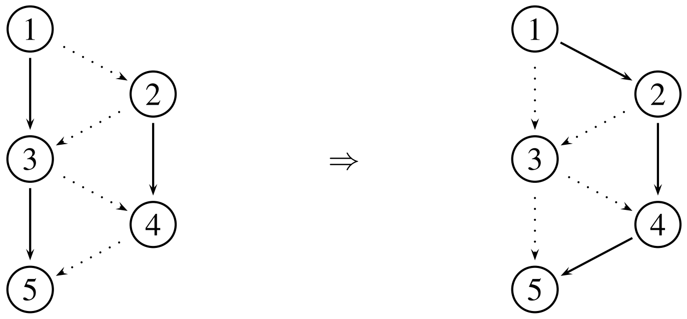


Figure 3.11: Isolating a Steiner node

Chapter 4

The Engineering Process

In this chapter, we describe the engineering process which leads from a first integer linear programming model to an efficient fully combinatorial approximation algorithm producing near optimal solutions in real-time. To this end, we present the different stages of the development with their experimental evaluations and the considerations that motivated certain theoretical investigations and improvements of practical performance.

For the experimental evaluation, we will preliminarily consider a limited test set of instances to get first benchmarks. To this end, we will use a test set reflecting a use-case of the application as such instances should be preferred over artificial ones, e.g. randomly generated. When we will reach the point where we will have an implementation presumably coming into commercial operation, we will diversify and extend the testsets.

Thereby, we can steadily measure the progress with respect to multiple criteria. In our case, these criteria are the achieved objective value, the running time, and the complexity of the hardware implementation. Whereas the objective value is universal, the running time strongly depends on the testing environment. Therefore, we will consider them mostly from a qualitative point of view, especially the absolute running times on ordinary PC hardware.

The outline of this chapter is as follows. First, we evaluate the basic (integer) linear programming formulation with the commercial solver CPLEX. Afterwards, we analyze the separation and augmentation approach. Taking the previously found issues into account, we present a very competitive algorithm which adheres to the *Single Pass Paradigm* and which is well suited for being implemented in the hardware of a driver chip. Some of the engineering details remain undisclosed as they are subject to a commercial implementation of our driving scheme.

4.1 At a First Glance

At the beginning of the engineering process, we have to deal with the selection of the first test set and the representation of the data. We have chosen to represent the image data by the `ppm`-

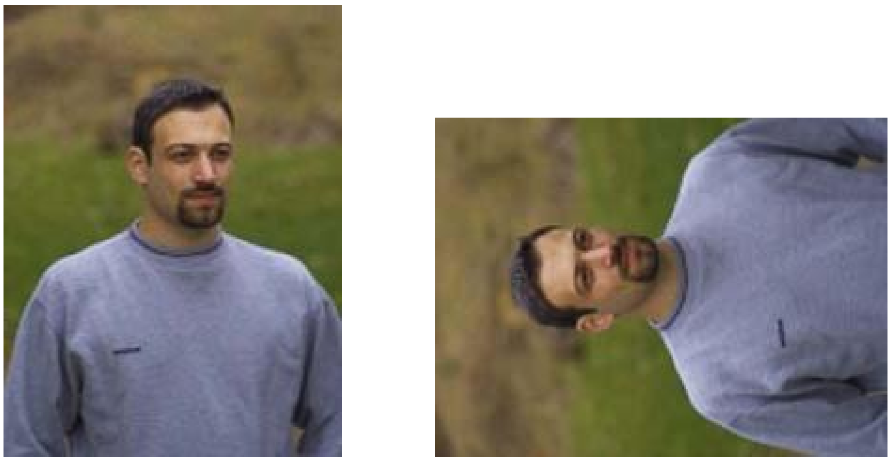


Figure 4.1: The images are flipped such that the smaller dimension becomes the number of rows.

format[1] in its plain text version. Though this choice involves more hard-disk space, the following advantage of having a human readable representation prevail: We can easily implement input and output routines to read the instance and write the decomposition. Thereby, we may analyze input and output images to examine the decomposition algorithms.

We implemented a converter to generate the basic integer linear programs in the `lp`-format [33] from an instance given as a `ppm`-file. For a given image of n rows, m columns, and multiline order k, the basic linear program contains roughly $n \cdot m \cdot k$ flow-variables, roughly $n \cdot k$ capacity-variables, roughly $n \cdot m \cdot k$ constraints for bounding the flow by the capacities, and $n \cdot m$ equality constraints to ensure a lossless decomposition.

To get a rough idea on the size of the images and the corresponding linear programs, we present the numbers of a typical display on a handheld device with QQVGA (read quater-quater-VGA) resolution, i.e. $n = 120$ rows each containing 160 RGB pixels ($m = 480$). Hence, we get linear programs with 227994 variables, 285120 constraints, and 1021440 non-zeros in the constraint matrix.

As a preliminary testset, we used portrait photos as they reflect a common use-case for mobile phones with a built-in digital camera. We scaled them down to different numbers of rows, such as $n = 30, 60, 90, 120$ while keeping the aspect ratio fixed, i.e. $m = 4n$. The notion of rows and columns is discretionary. It is advisable from a technical point of view to choose the smallest of the display dimensions as the number of rows and the larger one as the number of columns and placing the RGB subpixels in the same row. This can be easily seen on the basis of our objective where we sum over the rows and take the maximum over the columns. Hence, we flipped all test image accordingly as depicted in Figure 4.1.

We generated the corresponding `lp`-files to solve them with the commercial CPLEX interactive optimizer. As a representative, we will present the computational results for the portrait of the

[1]`http://netpbm.sourceforge.net/doc/ppm.html` as of October 30th, 2007

author (see Figure 4.1). For the multiline order $k = 2$, the optimum objective values for the different number of rows are shown in Table 4.1.

number of rows	30	60	90	120
original values	5473	11364	17367	23131
objective values	2976	5899	8860.5	11688
reduction ratios	0.54	0.52	0.51	0.51

Table 4.1: Computational results for the flipped image of Figure 4.1 and multiline order $k = 2$.

The running times are depicted in Figure 4.2. Here, we restrict ourselves to this particular image since the results with respect to the running times are similar to the other tested instances.

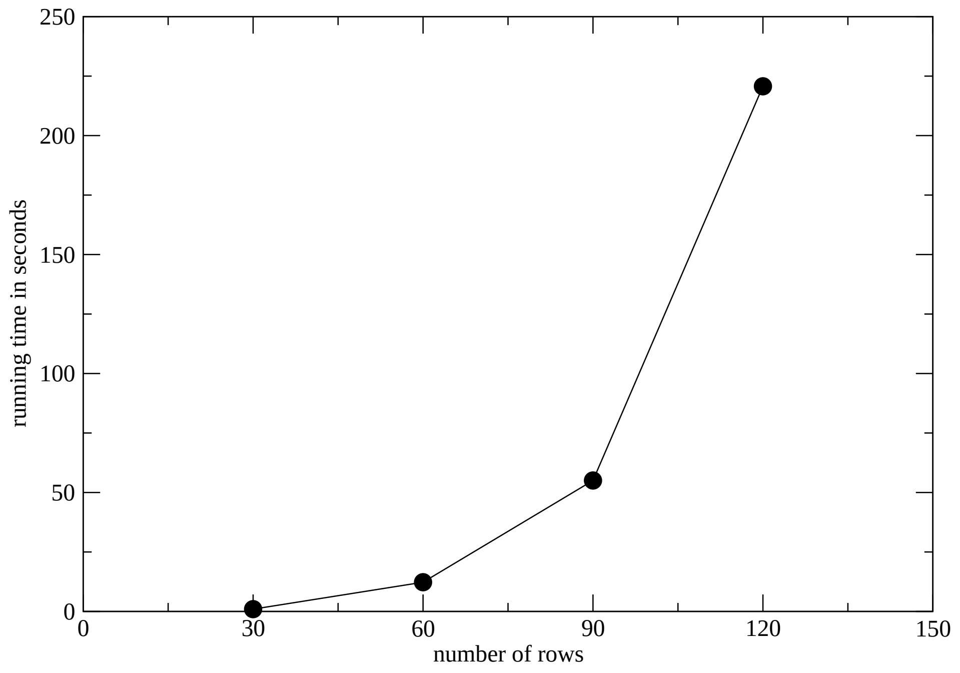


Figure 4.2: Running times to solve the CMLA2 linear program corresponding to the inset image with `baropt` of the CPLEX interactive optimizer 10.0.0. For the original computations, an older version and a different machine have been used. However, we present the results of this version because it is available on the same machine as used for the other measurements of this chapter.

As mentioned before, we had been more interested in the qualitative results then in quantitative ones at this early point of the engineering process. Therefore, we did not compute the optimum integral solution but the optimum of the linear programming relaxation since the (fractional) LP solution is a lower bound on the objective value of the optimal integer solution which is within an additive error of $k \cdot n$. This accuracy is sufficient to assess the improvement in lifetime. As mentioned in Chapter 2, the lifetime strongly depends on the physical conditions like temperature and luminance. To get a more objective measure of the effect of Multiline Addressing, we compare the objective values to Singleline Addressing, i.e. to the sum of the row-maxima of the original image (*original value* for short). We define the reduction ratio as the quotient of objective value and original value. As a rule of thumb, a reduction ratio of $\frac{1}{q}$ increases the lifetime by at least a factor of q. These early estimates of the reduction ratios ranged roughly from 51% to 75% for Doubleline Addressing and from 26% to 45% for $k = 4$. Hence, the expected increase in lifetime of passive matrix OLED devices has been promising so that it supported the decision to pursuit the project.

As one can see in Figure 4.2, the running time for solving the linear program rapidly increases with the number of rows for the Doubleline case. For $k = 4$, the running times grow even faster. They raise from about 3 seconds for $n = 30$ to about 400 seconds for $n = 90$. Therefore, we apply a simple but effective trick by cutting the instances into segments, e.g. we cut the image with 90 rows into three pieces of size 30 each. Thereby, the running time drops to about 100 seconds and the objective value increases from 5491 to 5763.3, i.e. by about 5%. Note that the number of columns is not effected by this segmentation. As one can see in Figure 4.3, the effect on the running time is well described by a linear relation and the objective values are nicely fitted by $5368 \cdot (1 + \frac{2.18}{s})$ where s is the segment size. This result has motivated to use the segmentation idea for proving the existence of the PTAS where we have got a similar dependence of the bound on the approximation value on the segment size. We will reconsider this idea to save memory in Section 4.4.

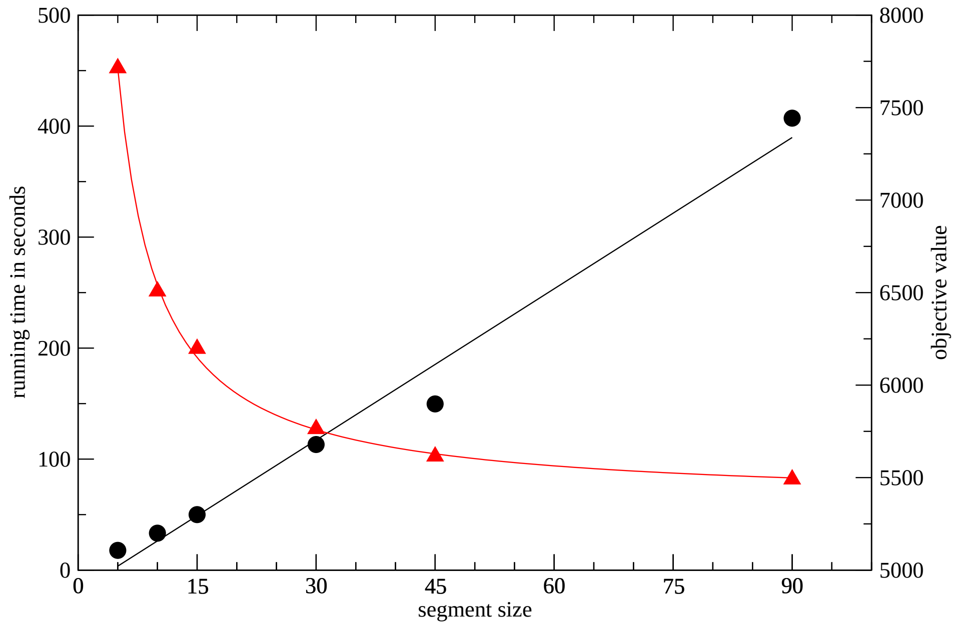


Figure 4.3: Effect of the segmentation on running time (black circles) and objective value (red triangles).

The concise network formulation by itself has not given any advantage with respect to the solution time of the LP solver, since it originates from the basic LP formulation just by simple row operations and most importantly the number of variables remains the same.[2] This drastically changes with the network design formulation as we consider a projection to the space of the capacity variables which has dimension of roughly $k \cdot n$. Note that thereby the number of constraints is no longer polynomially bounded by n and m in general. But this fact does not constitute an obstacle to an efficient solution since we already know from our theoretical investigation that there is an efficient separation routine. Moreover, for $k = 2$ Lemma 4 tells us that only $O(n^2)$ cut inequalities are non-dominated. Thereby, it becomes feasible to generate the `lp`-files for programs with $n = 180$ and $m = 720$ and even compute the best integer solution within several seconds for $k = 2$.

While this indicates that we are on the right track, it is clear that we can not implement an ILP solver on a chip that drives an OLED display, e.g. in a mobile device, at reasonable cost. Therefore, we are looking for fully combinatorial heuristics. In the next section, we will present and evaluate such heuristics, which achieve this design goal, based on the generic routine to solve the TRANSSHIPMENT problem by a MAXFLOW/MINCUT computation.

4.2 Separation and Augmentation

We first describe the general framework of our separation and augmentation heuristics. In addition to the following explanation, we illustrate the general framework with pseudo-code in Figure 4.4.

We start with an initial assignment of the capacities returned by some function INITIALIZE. We will discuss it later. Consider it to be a function that simply returns the zero vector for now. Afterwards, we iterate until we have found a feasible solution. In each iteration, we first solve the separation problem by a MAXFLOW/MINCUT computation. Depending on the outcome of the separation, we either return the feasible solution, or we augment one or several capacities according to the chosen strategy. When we only augment one capacity variable, we may benefit from the efficient treatment of capacity adjacent problems as discussed in Section 3.4.1. Thereby, time for each iteration is linear in the input size. In practice, the performance is even better since on average the augmenting paths are rather short. Moreover, we have to consider only the columns that have this arc in their current MINCUT since otherwise the additional capacity would not have any effect.

One issue of this approach is that we maintain the flow-variables f. On ordinary PC hardware the memory consumption is negligible and we have to compute the decomposition of the image, which is represented by the flow-variables, at the end in any case. But we need roughly k times more memory than the input size which makes the implementation on a chip more expensive, as the area of the chip increases accordingly and amounts to a considerable share. We address this important issue separately in Section 4.4.

[2]It should be mentioned that we have also thought of the adaption of known minimum cost network flow algorithms to our special objective. But this did not lead to satisfactory results and hence this direction of research has been dismissed early.

```
Input: $G = (V, A)$, $d = (d_1, \ldots, d_m)$
Output: $f = (f_1, \ldots, f_m)$
$f = 0$
$u = \textsc{Initialize}(d)$
while($f$ is not feasible){
    $f = \textsc{MaxFlow}(G, d, u)$
    $C = \textsc{MinCut}(G, d, u)$
    foreach($a \in A$){
        $u(a) = u(a) + \Delta u(a, C)$
    }
}
return$f$
```

Figure 4.4: The framework for the approximation heuristics for general directed graphs $G = (V, A)$. We are asked to compute a family of flow-variables f_j that satisfy the respective demands $d_j : V \to \mathbb{Z}$ for each $j = 1, \ldots, m$. The variables u denote the capacities, and the notion of the function Δu covers several variants of our heuristics for augmenting said capacities. Note that it is sufficient to return the flow-variables as the capacity of an arc a is then implicitly given by the maximum flow $f_j(a)$ running over this arc, where the maximum is taken over all $j = 1, \ldots, m$.

After having found a violated cut, the question arises which capacity variables to augment. Unlike in the framework of Garg and Könemann [34] where all variables of the inequality would be multiplied with a constant, we select only one variable to augment for the following reason. We shall use only fixed precision datatypes for an economical implementation. Thus, we consider the variables to be integer all the time. By selecting only one variable, we gain a finer control since we do not have to deal with cases, e.g., where the violation of an inequality is less than the number of variables contained in the corresponding cut. Though this means more iterations, we benefit from the fast solution of capacity adjacent problems as mentioned before.

Our strategy is to augment the most prospective variable that is in any of the cuts of the different columns that are not feasible yet. That is, we compute the most violated inequalities for every d_j with $j = 1, \ldots, m$. We measure the potential impact of a capacity by the number of different columns, i.e. cuts, it appears in. This is equivalent to summing up all the inequalities over all columns which gives us a valid violated inequality too. The *basic greedy* approach selects the capacity variable having the highest potential impact, i.e. we increase the capacity with the highest coefficient in the sum of the inequalities. A slightly different variant of this approach takes only the variables into account which appear in the most violated cut inequality (referred to as *max-column greedy update*), i.e. the cut which attains the minimum in the separation problem where the minimum is taken over all columns.

Since we maintain the integrality of the variables throughout the algorithm, we have to increase the variables at least by 1 in each iteration. With this value, the running time is then proportional to the size of the display and the difference between the achieved objective value and the one from the initial solution. By adding a greater constant or a certain fraction of the previous capacity like in the framework of Garg and Könemann, the running time would improve, however this would be on the expense of the approximation ratio. Moreover, this approach deals with fractional arithmetic in the first place. Though it would be possible to use bitshifts instead of multiplications, this would require a higher intermediate presicion of the numbertypes to compensate the coarser augmentation of the capacities. A higher intermediate precision could be easily achieved by scaling the constraints, i.e. the demands, but this would also increase the costs of a hardware implementation. Hence, we consider this option as a less-than-ideal solution. We will see in the next section that there is also a fully combinatorial algorithm which gets by with a limited number of iterations but which does not do so on the expense of the objective value.

As a testset we used the portraits of 197 employees of the MPI. While the original images have a resolution of 180×240, we scaled them down to $n = 60, 90, 120, 150$ keeping the aspect ratio constant such that we have $m = 4n$ for all images.

As initialization routines, we tested three variants that we already mentioned in Section 3.5.2. First of all, there is the trivial initialization by the *all-zero vector*. We may disregard this variant here as it may be replaced seemlessly by the *save lower bounds*. Furthermore, we will consider the initialization with the solution of the system of *easy constraints*.

We present the objective values for the multiline orders $k = 2, 4, 6$ in Figure 4.5 where we initialize the capacities by the easy constraints (ec), used a blocking flow algorithm (bf) for the separation, performed the capacity augmentation by the maximum column greedy update method (mcgu).

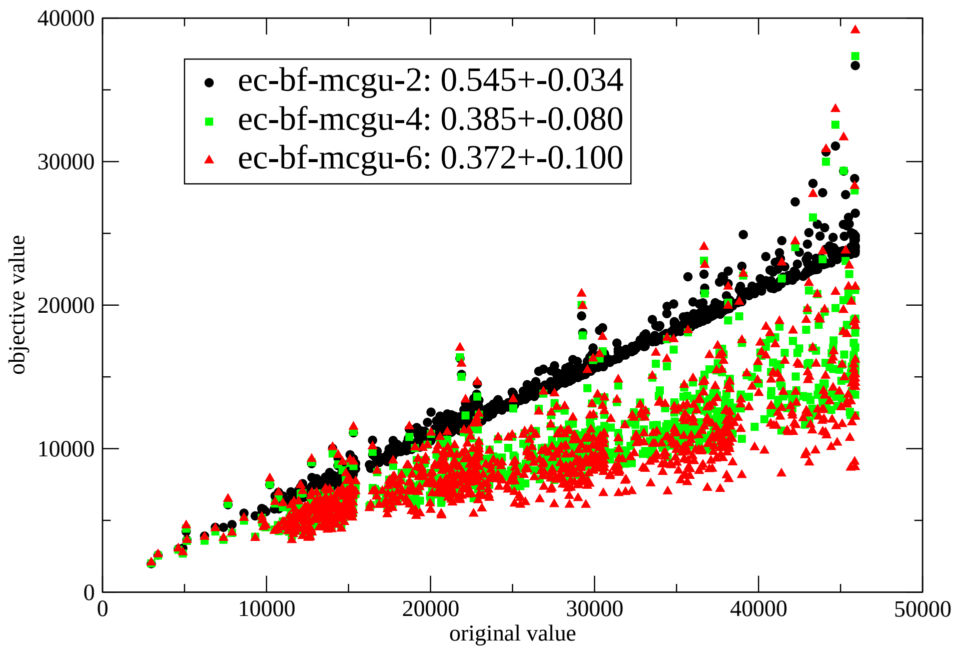


Figure 4.5: The objective values for $k = 2, 4, 6$. The inset shows the average reduction ratios and their standard deviations respectively.

For $k = 2$, the average reduction ratio is roughly 0.545 with a standard deviation of 0.034. The green squares are the results for $k = 4$. Here the average ratio is 0.385 ± 0.065. The red triangles represent $k = 6$ with a ratio of 0.372 ± 0.100. The theoretical lower bound for these ratios being $1/k$. We also implemented diverse other strategies for the capacity update, e.g. book-keeping about the variables that have been already present in recent cuts and thus are considered with a greater weight. However, it turned out that the max-column greedy update (`mcgu`) performend best on average. Furthermore, we are in particular interested in the average reduction ratio since this measure reflects best the benefit of CMLA in everyday usage. We will discuss this in more detail at the end of the next section.

In Figure 4.6, one can see the dependence of the running time on the input size together with the distribution of the instances. We connected the median running time to guide the eye. The fit of these medians with respect to a power function yields $t = 21\mu s \cdot n^{2.23}$ which is almost linear in the number of pixels.

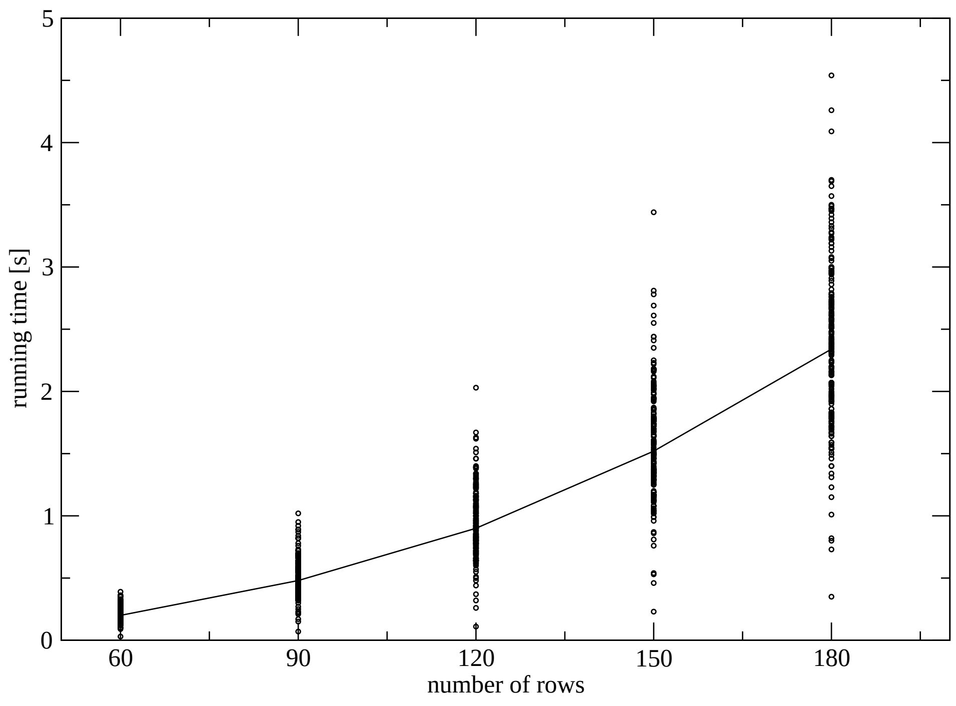


Figure 4.6: The dependence of the running time on n in case of $k = 2$.

Not using the easy constraints but safe lower bounds yields slightly better ratios as depicted in the inset of the diagram of Figure 4.7. Moreover, the safe lower bounds resolve an issue of the easy constraint initialization where we have a few instances with worse objective value for higher multiline oder as one can see by comparing Figure 4.5 and Figure 4.7.

But the running times grow faster with the number of pixels as one can see in Figure 4.8 where the three top curves belong to the median running times using save lower bounds with $k = 2, 4, 6$ respectively. Whereas the median running times using the easy constraint initialization and $k = 2, 4, 6$ yield the three lower curves.

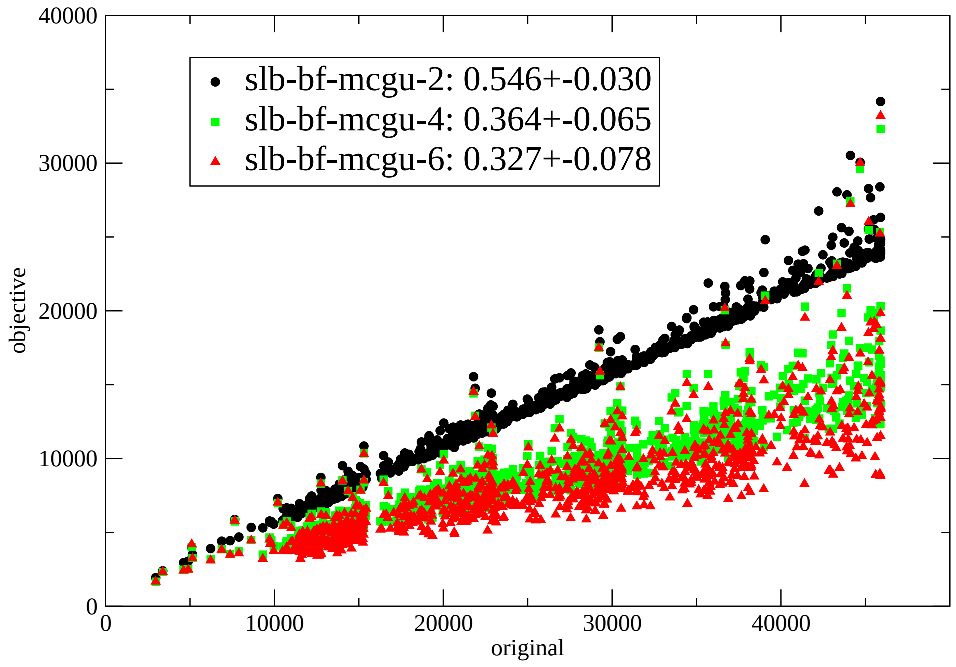


Figure 4.7: Using safe lower bounds for initialization yields better objective values.

In practice we have a display of fixed size and the requirement on the running time to be below the perception of a human eye. According to this time constraint, we have to design our algorithm such that the cost of the integrated circuit which implements it is minimized. Parallelization, e.g. working on several columns concurrently, and reusing results of certain computations, e.g. shortest path trees of previous iterations, decrease the running time but increase the complexity and memory usage of the circuit and hence the production costs.

The important difference between both approaches of Figure 4.8 is the number of augmentations which is thereby identified as a big threat to the running time. In fact, the running time is not polynomially bounded in the bitsize of the input because the number of augmentations is proportional to the objective value.

We will leave it at the presentation of these results for the separation and augmentation approach, since they are sufficient to illustrate the following findings. First, we have shown that we can achieve reduction ratios not far from the theoretical lower bound of $\frac{1}{k}$ with fully combinatorial heuristics. Second, the main weakness of this approach is the number of iterations.

Substituting the blocking flow routine for the MAXFLOW computation by some other method like push-relabel would hardly change the number of iterations (only by chance, if the MINCUTS are not unique). The same holds for the linear time TRANSSHIPMENT method. We will see that this routine plays a more important role in the next section, when we compute feasible capacities already in the initialization. Moreover, the separation and augmentation approach will be rendered obsolete by the following results.

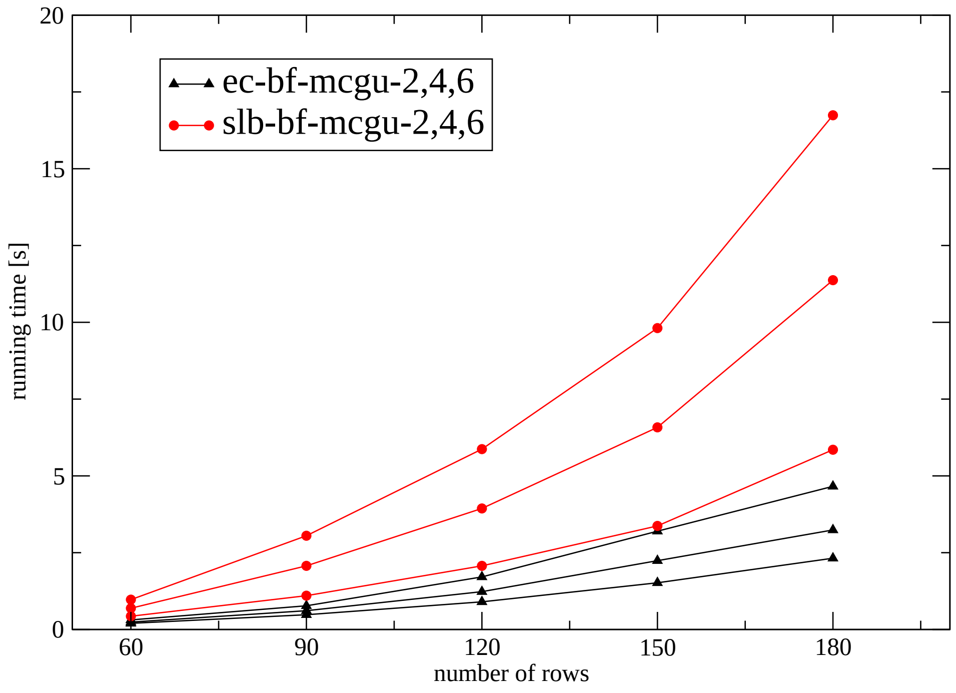


Figure 4.8: Using safe lower bounds for initialization yields worse running times.

4.3 The Breakthrough in the Doubleline Case

Recall that we intend to develop an algorithm that finds a decomposition in real-time while keeping it simplistic enough such that it can be implemented on a chip with a low hardware complexity. Hence, we are looking for a linear algorithm that takes only a few clock cycles per pixel in a hardware implementation. We established the single pass paradigm in Section 3.5.2 based on theoretical considerations. However, this is also advisable from a practical point of view since we can not perform an arbitrary number of random accesses to the memory in one clock cycle at reasonable cost

In the previous section, we have seen that the main bottleneck of our heuristics is the number of iterations. Moreover by using the linear time solution of the easy constraints, we have been able to cut the running time considerably. Our theoretical investigations have shown that we may exploit the special structure of the graph to get linear time algorithms for the separation and the TRANSSHIPMENT problem. Furthermore, we derived a PTAS that takes its decision in some local sense.

Hence, it is natural to ask whether we can combine the idea of the easy constraint solution and the linear time separation to compute feasible capacities immediately in the first pass. Recall that we can find the easy constraint solution by a shortest path computation on a directed acyclic graph. To this end, we have ordered the capacity variables lexicographically by tail and head. That is, we first consider the outgoing arcs of node 1, then the ourgoing arcs of node 2, and so on. Thereby, we can formulate a dynamic program that determines one capacity after the other. Moreover, we may run the separation concurrently and immediately detect if a constraint is violated, whose variables have already been fixed. More than this, we can decide on the fly to which extent we have to augment a capacity from the easy constraint solution such that a feasible solution to the whole ILP remains possible. Put differently, when we consider an arc, say a, then all capacities of the previous arcs have already been fixed. If there are any unsatisfied cut inequalities containing $u(a)$ as the last variable, then we set $u(a)$ to the largest violation of all these cut inequalities. Hence, we fulfill all of these inequalities and obtain a feasible solution at the end.

With respect to memory requirements which become an issue in the hardware implementation, in this case it is sufficient to store the distance labels for the vertices in the separation graph for each $j = 1, \ldots, m$. Moreover, we only need the labels of one block at a time. Hence, the extra space is only of size $O(m)$ which is negligible compared to the size of the input which is given by $n \cdot m$ integral numbers. However, we may reuse the distance labels for the the computation of the flow-variables.

Though the idea is generic for arbitrary but fixed k, we need specialized implementations for each k to make the most efficient code out of each case. Moreover, from an engineering perspective, it is advisable to focus on the simplest non-trivial case first and generalize the applied concepts afterwards. Therefore, we have first implemented the single pass paradigm for the case $k = 2$.

We have restricted the look-ahead to 3 rows in our implementation. An evaluation of a first version on the `mpi` testset was performed before the theoretical investigation of the online problem in the monochrome case. Compared to the separation and augmentation approach, the running times dropped by about two orders of magnitudes due to the single pass capacity computation and the subsequent solution of the TRANSSHIPMENT problem in linear time.

In Figure 4.9, we show the running times for each instance. The red dots at top represent the old measurements of *ec-bf-mcgu-2*. Whereas the dots at the bottom correspond to the new algorithm. We connected the measurements by lines to guide the eye. Note the logarithmic scale of the time axis. One can see that the new algorithm is two orders of magnitudes faster than the old one. Comparing the mean running times, the improvement is more than a factor of 300 between *ec-bf-mcgu-2* and the new algorithm. Moreover, the variance decreases drastically. This is due to the fact that the running time of the old algorithm depends strongly on the input data, i.e. on the unary encoding length, while it scales only with the size of the binary encoding length in the new one.

Also the results with respect to the objective values have been very promising. But on some instances the objective values fell short of the ones achieved by separation and augmentation (see Figure 4.10). This issue offered an additional motivation to consider the online problem.

In order to support the theoretical investigations, we generated all monochrome instances for $n = 5$ and $n = 6$. Such an exhaustive search on monochrome instances of a few lines are possible since we already know from Lemma 6 that it is sufficient to consider at most n commodities per instance. We thereby identified the instances for which we got sub-optimal results. This led not only to the theoretical results like the lower bound $\frac{4}{3}$ for the approximation factor but also to

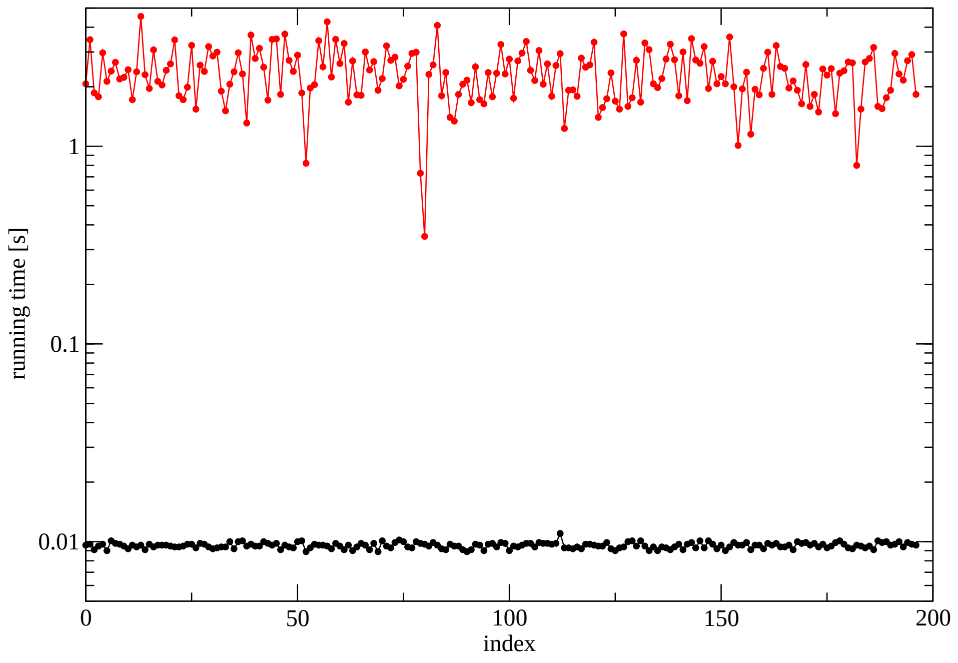


Figure 4.9: Comparison of the running times of ec-bf-mcgu-2 (top) and the first implementation of the single pass paradigm (botton).

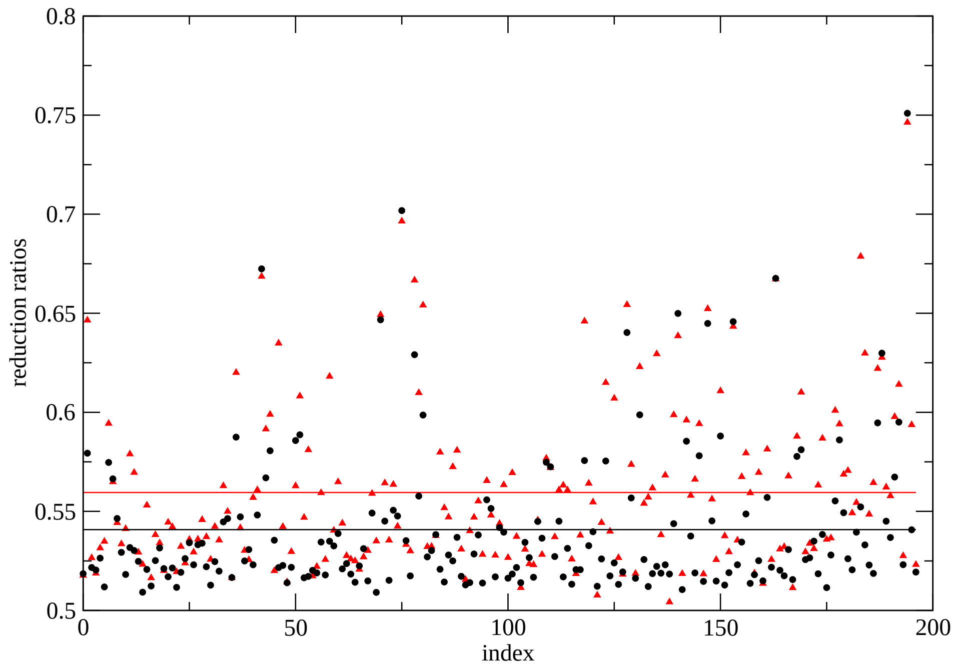


Figure 4.10: Comparison of the reduction ratios of ec-bf-mcgu-2 (black dots) and the first implementation of the single pass paradigm (red triangles). The horizontal lines show the set averages which are roughly 54% and 56%, respectively.

the algorithm `mlacompact` which incorporates the rules of Subsection 3.6.1 into the previous implementation.

As one can see in Figure 4.11, this yields near optimal solutions in practice. To support this claim, we use the exact solution that we gained by solving the network design formulation for doubleline addressing with the commercial solver CPLEX as mentioned in Section 4.1. Thereby, we obtain the optimal solutions and are able to compare the per-instance approximation ratios of *ec-bf-mcgu-2* and `mlacompact`.

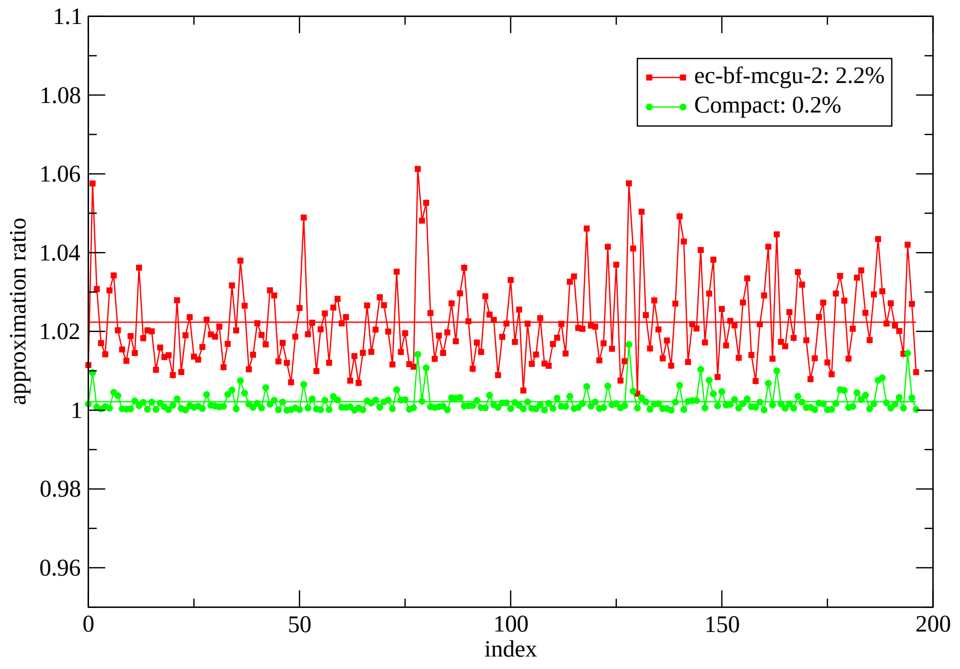


Figure 4.11: Results of the old and the new doubleline algorithms normalized to the corresponding optimal solution. The horizontal lines indicate the respective means.

As one can see, the quality of the approximation of the new algorithm is not worse than for the separation and augmentation approach. In fact, on average it is even considerably better. Recall that the objective of our optimization problem is proportional to the electrical current and therefore has a direct impact on the lifetime of such a passive matrix OLED display. The average gap of 0.2% shows that we have found an algorithm that does not leave much room for improvement with respect to the necessary electrical current to display real world images using the doubleline addressing scheme.

As of yet, we only focused on the `mpi` testset to evaluate the algorithms in practice. This was sufficient so far as they led to clear improvements. We have thereby reached the point in the design process where an economical hardware implementation becomes possible. Therefore, we now discuss the applicability of consecutive Multiline Addressing to a broader set of images. The corresponding measurements have been performed as a part of a bachelor's thesis [35]. To be self-contained, we summarize and discuss the results in the following.

Based on the results for human faces, it is natural to ask for photographs in general. It turned out that on a variety of over 3000 pictures, `mlacompact` achieves a reduction of the electrical current to 56% on the average with a mean per instance approximation ratio of 1.003 compared to the optimal solution provided by CPLEX. The results for two exemplary music videos are even better with a reduction to 51% which is only a factor of 1.002 away from the optimum.

We explain this behavior by the fact that the content of photos is rather smooth, e.g. they are not dominated by sharp edges as in artificial images like cliparts and text by bitmap fonts. This is in agreement with the results on a testset of wallpapers for mobile phones with a mean reduction to 63% and approximation factor of 1.005 averaged over about 4500 samples. We observed that diagonal lines, in particular if the width is only one pixel and the contrast to the neighborhood is high, e.g. as in letters like in Figure 4.12, constitute an obstacle to Multiline Addressing.

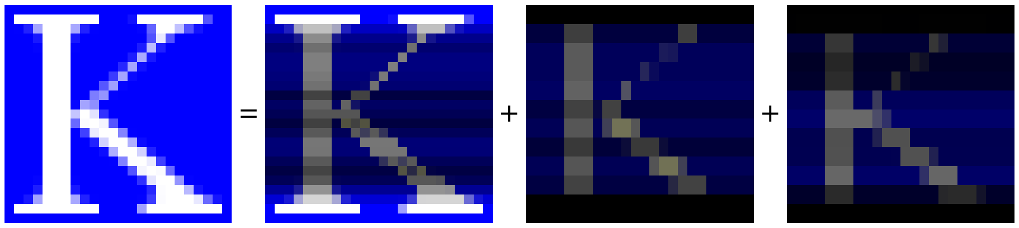


Figure 4.12: Diagonal lines form an obstacle to CMLA. The images have 23 lines and the reduction ratio is about 77%. Without the antialiasing it would be even worse.

Considering videos, it becomes clear why we are in particular interested in the average reduction ratio. The diodes are more or less uniformly exposed to the stress, and the degradation is a long drawn process. Moreover the worst-case reduction ratio for any algorithm is 1 since an image that is represented by a diagonal matrix can not be decomposed into multilines. Note that this also holds for non-consecutive multiline addressing. On the other hand, we advise designers of user interfaces for CMLA driven OLED devices to bear in mind that they may delay the so-called burn-in effect of frequent steady images by smoothing sharp diagonal edges, e.g. by antialiasing.

As mentioned before, we claim that an economical hardware implementation becomes possible. The hardware complexity of the logics to implement `mlacompact` is only a few thousand gates for QQVGA displays. This yields a sufficiently small area on a silicon wafer that is necessary to be in business on the highly competitive display market. However, if we use the same amount of RAM as we did on the PC, that is to store the complete output, the area will increase significantly. We will address this issue in the following.

4.4 Memory

In this section, we will discuss some approaches to reduce the space requirement. Basically, we may sacrifice a fraction of the achieved objective value to save memory, or we make a tradeoff between the speed of the computation and the memory requirement. To get an idea where we may save memory, we will first explain the interaction of a multiline decomposition algorithm with the other components of the OLED display driver.

Recall that the objective value, i.e. the sum of the capacities, determines the amplitude of the electrical current that is applied to the columns of the display. Hence, we have to compute this value before we start displaying any content. On the other hand, we can benefit from this since we do not need to store any flow-variables during the capacity computation. Hence, deviding the decomposition problem into a network design problem and TRANSSHIPMENT problems is also preferred from the application point of view. We illustrate the data flow in Figure 4.13.

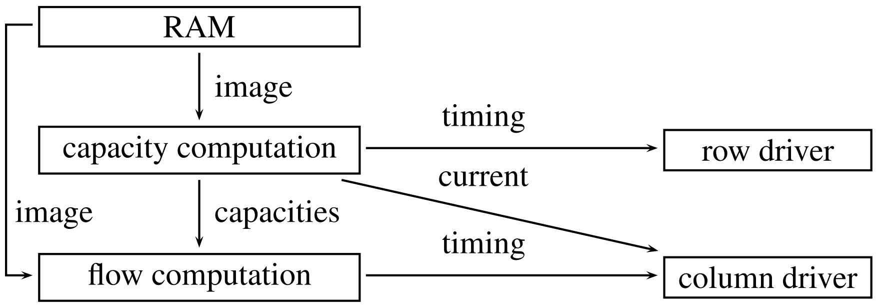


Figure 4.13: Data flow of the decomposition

Recall that the capacities coincide with the timings of the row switches and the flow-variables determine the timing for the columns. Hence, we need the flow values of one (multi)line at hand when the corresponding row switch is closed. This allows us to solve the TRANSSHIPMENT problem *on the fly* to some extent and thus save memory for storing the flow-variables of the whole image.

To this end, we recall the linear time solution of the TRANSSHIPMENT problem. There we first compute the distance labels of the separation graph in one pass and in a second backward pass, we compute the flow-variables. Note that the solution of the TRANSSHIPMENT problem is independent for each column. Hence, we only need to store the $O(n)$ distance labels to compute the flow variales of one column.

Nevertheless, we have to process the flow-variables line by line and not column-wise. To work around this issue, we may apply an idea similar to the segmentation used in the PTAS and at the beginning of this chapter. That is, we cut the image into segments of size s and consider each subproblem independently. Thereby, we only need a framebuffer of size $O(s \cdot m)$ for the flow-variables since we can display the segments one after the other.

Note that in general the objective values increases because we omit some of the multilines. This approach has been evaluated in [35]. To be self-contained, we present the results for the doubleline case and the `mpi` testset in Figure 4.14. As one can see, the reduction factor increases linearly with the number of omitted arcs.

By this approach which we call `mlafixed`, the memory requirement is $m \cdot n + O(n + s \cdot m)$ where the first term is the size of the input and the second contains the space for storing the capacities and the flow-variables in O-notation hiding a constant only depending on the multiline order k.

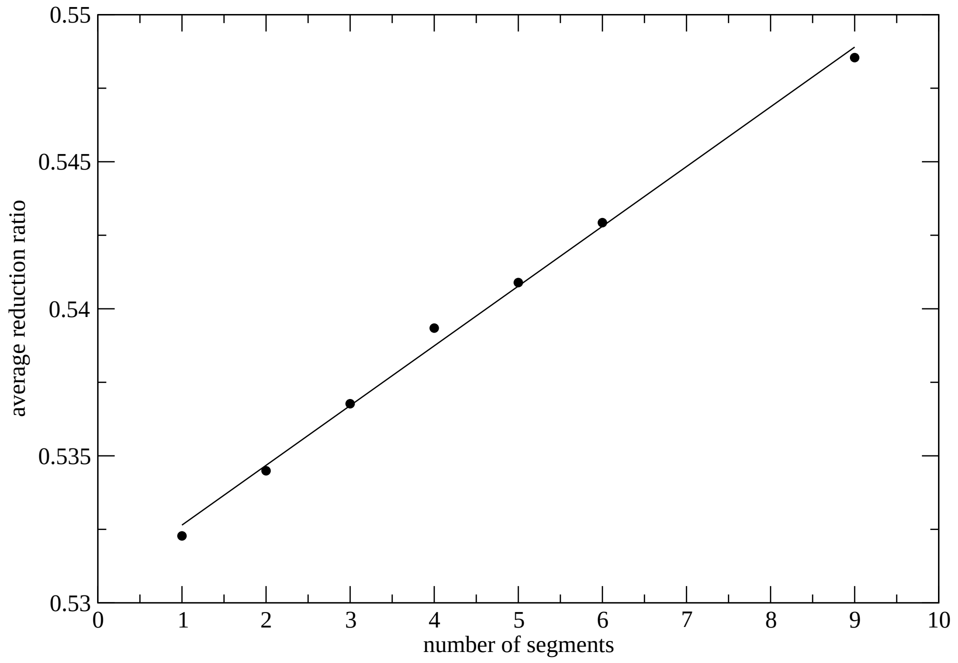


Figure 4.14: The dependence of the number on segments on the reduction ratio for the `mpi` testset. The slope of the regression line is 0.2% which means that for each additional segment the reduction ratio increases by this amount.

It is natural to ask whether this is the best possible performance with respect to the reduction ratio that we can achieve within the that bound on the memory. Indeed, we can improve as we will show by means of the Doubleline case in the following.

The key feature of `mlafixed` is that we omit the arcs that *overlap* the breakpoints (see Figure 4.15). In other words, we implicitly set the capacities of these arcs to 0. Thereby, the incoming flow of a breakpoint node from the segment above is determined as well as the outgoing flow of the breakpoint when we consider the segment below. This artificial bottleneck makes the flow computation of both segments independent from each other.

Hence, we can start from the backward computation of the flow-variables from each breakpoint as well as the computation of the distance labels in the separation graph in forward direction. But essentially, we could achieve the same if we would set the capacity of an overlapping arc to a positive value and determined the corresponding flow-variables in a predefined way.

That is, we implicitly fix the flow values of the overlapping arcs during the capacity computation adhering to the following two conditions. The flow values of a overlapping arcs depends only on local information at the breakpoint, and its choice still permits a feasible solution to the whole problem.

Thereby, we can easily recover the flow values of the overlapping arcs and start the backwards computation from the breakpoints since the instance also decomposes into independent subprob-

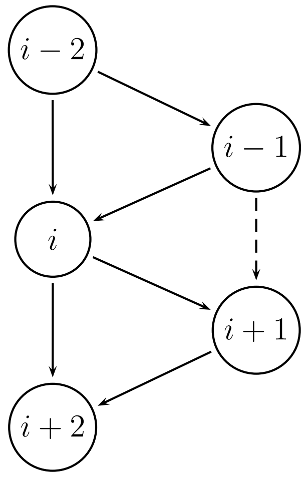


Figure 4.15: The dashed arc $(i-1, i+1)$ overlaps the breakpoint i.

lems. Moreover, we can restart the computation of the distance labels at the respective breakpoint when the preceding segment is displayed.

We call this approach `mlaoverlap` and present a comparison to `mlafixed` with respect to the reduction ratio in Figure 4.16. The linear behavior of the average reduction ratios with respect to the number of segments persists. However, the slope of 0.07% is only about a third of `mlafixed`.

The running times of both approaches are very close to the running times of `mlacompact` [35]. Moreover, the memory requirement for `mlaoverlap` is also $n \cdot m + O(n + s \cdot m)$. Hence, `mlaoverlap` should be preferred over `mlafixed`. However, it is considerably easier to adopt `mlafixed` to higher multiline orders.

We have seen that we can reduce the memory requirements by sacrificing some fraction of the reduction ratio. However, we may also make a tradeoff between speed and memory. In the previous two approaches, we imposed additional constraints on the capacities such that we did not have to look at the complete graph G_n^k to solve the TRANSSHIPMENT problem for a particular column. This is indeed necessary as the following example for $k = 2$ shows. We consider the image

$$\begin{pmatrix}
1 & 2 & & & & & \\
1 & 1 & 1 & & & & \\
1 & & 1 & 1 & & & \\
1 & & & 1 & & & \\
\vdots & & & & \ddots & & \\
1 & & & & & 1 & \\
1 & & & & & 1 & 1 \\
x & & & & & & 1
\end{pmatrix}$$

with $x \in \{0, 1\}$. For either choice of x, the optimum capacities are $u^{(1)} = (1, 0, \cdots, 0)^T$ and

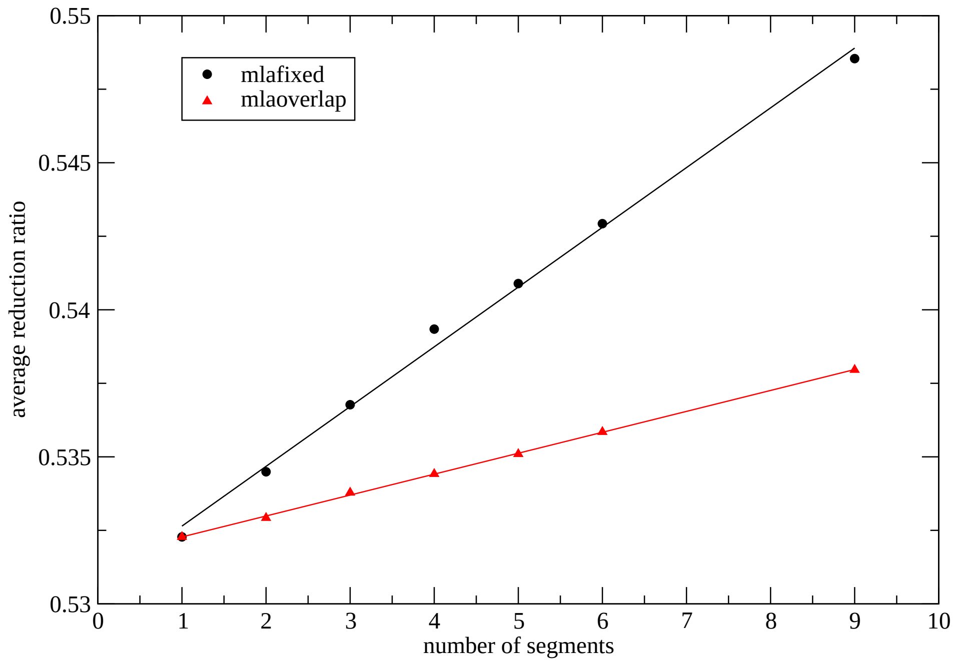


Figure 4.16: Comparison of `mlafixed` and `mlaoverlap` on the `mpi` testset for $k = 2$. The slope of the regression line of the latter is only 0.07% which is about a third of the slope of the former.

$u^{(2)} = (1, \cdots, 1)^T$. But considering the TRANSSHIPMENT problem for the first column, there is only one feasible way for each choice of x to route the outgoing flow of the first node. Hence, we have to scan over the whole graph before we are able to take a decision on the routing.

The naive way to remain within the memory bound of the previous approaches would be to compute the distance labels that are necessary for the last segment and discard the previous ones, then compute the flow variable for this segment and repeat the procedure until all segments have been displayed. This would involve a running time growing quadratically in n. But we may cache some of the distance labels so that we do not have to start from the beginning each time but at the nearest cached one instead. hence, we get the opposite extremal case if we cache all distance lables. This would allow a faster computation of the flow-variables than with `mlacompact` but we would not save any memory as the size of the cache would be in $\Theta(n \cdot m)$.

Let s be the number of rows that fit into the framebutter as for the previous two approaches. Hence, we compute and display the flow-variables in $\lceil \frac{n}{s} \rceil$ phases, i.e. the image is again divided into segments. By choosing the cached distance labels such that they coincide with the break-points of the segments, we can compute the flow-variables of the segments after another without any overhead. The running time thereby remains the same as for `mlacompact`. Note that we need $O(\frac{n}{s} \cdot m)$ additional memory when we cache the distance labels in this way. Hence, the optimum choice for s is in $\Theta(\sqrt{n})$ to minimize the memory consumption and to achieve the same reduction ratio and roughly the same running time as in `mlacompact`. The total requirement with respect to memory is then $n \cdot m + O(n + m \cdot \sqrt{n})$.

4.5 Beyond Doubleline Addressing

As of yet, we have presented competitive heuristics only for the Doubleline case, i.e. $k = 2$. We have also implemented the single pass paradigm for CMLA3. After some engineering similar to the Doubleline case, we achieved the following results depicted in Figure 4.17.

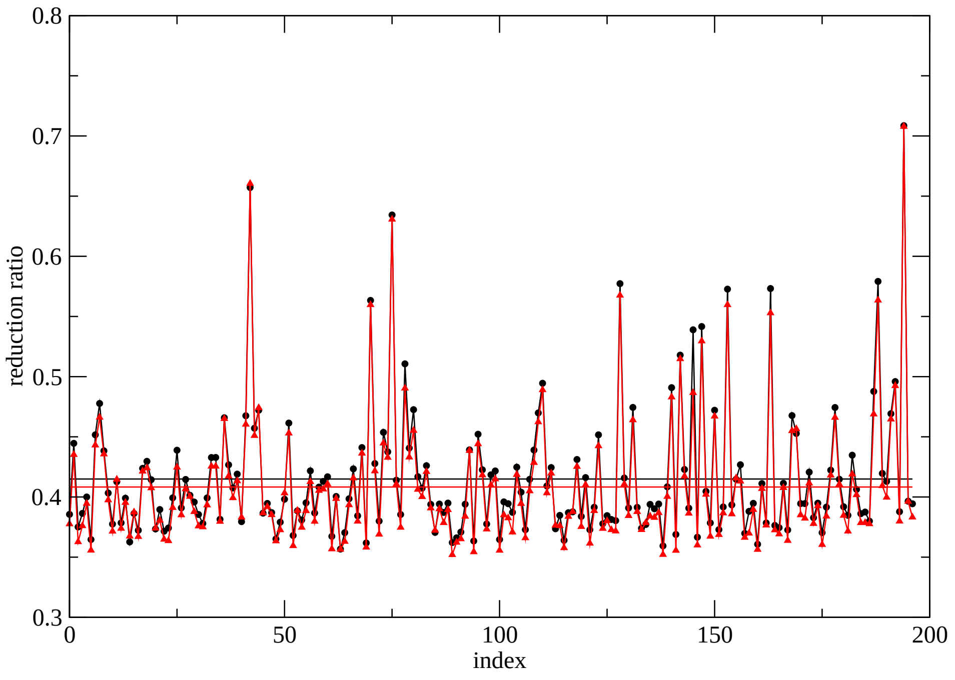


Figure 4.17: Comparison of the reduction ratios of separation and augmentation approach (red triangles) and the single pass paradigm (black dots) for $k = 3$. The horizontal line represent the average ratios of about 41.5% and 40.8%,respectively.

In principal, it is possible to extend the approach to $k > 4$. However, the effort to derive a concise dynamic programming formulation increases considerably. It is possible to use linear projection methods like Fourier-Motzkin elimination for this task, but it remains tedious. Moreover, the hardware complexity with respect to the size of the logics and the necessary amount of memory grows significantly. On the other hand, the improvement with respect to the reduction ratio becomes smaller when moving from multiline order k to $k + 1$.

However, there is a special case of CMLA4 where we discard the tripplelines as it allows a simple extension of our algorithm for CMLA2 by a cascading strategy. To this end, we first apply `mlacompact` to get a Doubleline decomposition of an instance R into $F^{(1)}$ and $F^{(2)}$. Then we separately consider the odd and the even lines of $F^{(2)}$, say $F^{(odd)}$ and $F^{(even)}$ respectively, as the input of a further Doubleline problem. The doublelines of the second stage then cover four lines of the original images (see Figure 4.18).

We call this approach `mlacascading`. It yields a slightly worse average approximation ratio than `ec-bf-mcgu-4` (by a relative factor of 1.02 as depicted in Figure 4.19). But this is

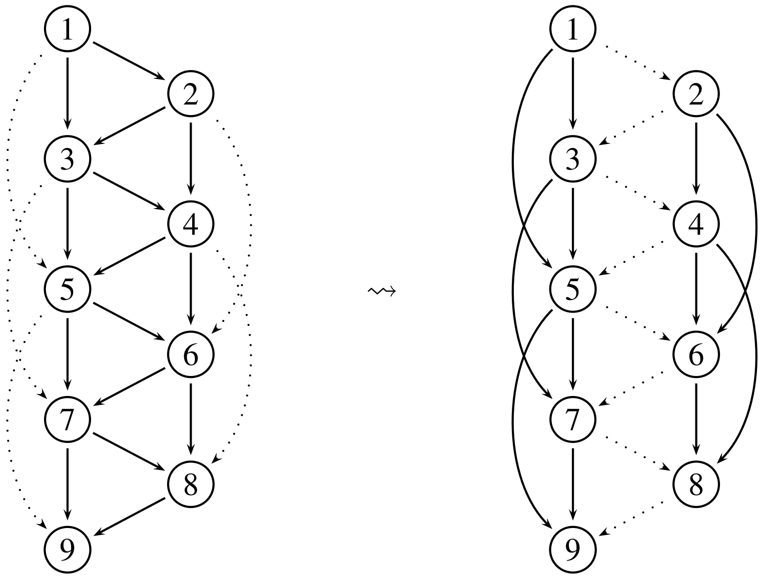


Figure 4.18: The Cascading scheme. First, we solve the CMLA2 problem on the left, then we split the instance and solve two independent CMLA2 problems

more than compensated by the improvement with respect to the running time by a factor of about 180. Moreover, it is *not possible* that `mlacascading` yields a worse objective value than `mlacompact` on the same instance whereas this behavior occurred on some instances concerning *ec-bf-mcgu-2* and *ec-bf-mcgu-4*.

Theoretically, we could apply a third phase to combine 8, 4, and 2 lines, but the additional effort is not justified by the small gain with respect to the reduction ratio as shown in [35]. Furthermore, it is possible to apply the same concepts as for `mlafixed` and `mlaoverlap` to the cascading approach. However, this makes only sense to a some extent as the improvement in the reduction ratio will be nullified by the loss due to the segmentation.

We can also extend the Doubleline Addressing in a different direction by dropping the requirement to combine consecutive lines. This would help on some pathological examples, e.g. if we are given a checkerboard as in the following image.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

In this case, we can not reduce the electrical current by CMLA. But by combining the lines

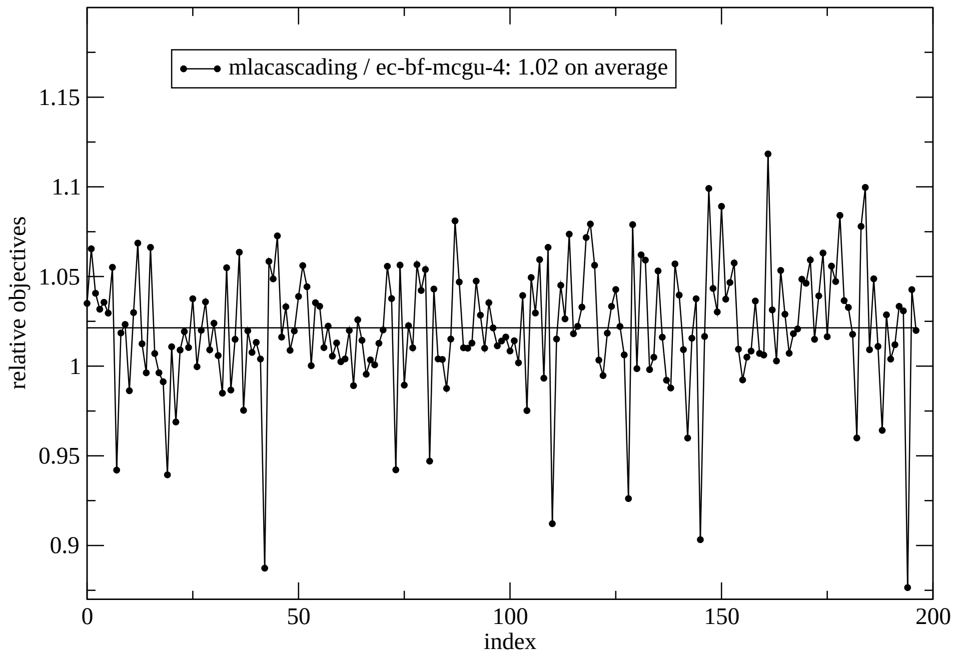


Figure 4.19: The relative objective of `mlacascading` with respect to *ec-bf-mcgu-4*. The horizontal line indicates the average ratio of 1.02

$(1, 3)$, $(2, 4)$, $(5, 7)$, and $(6, 8)$, we achieve a reduction ratio of 50%. If we allow the algorithm to permute the lines before the CMLA decomposition, we could work around this issue. However, this would involve considerably greater effort to perform such a step in real-time. Moreover, this would only pay off for specialized applications since we are already very close to the theoretical lower bound of $\frac{1}{2}$ for Doubleline Addressing on real-world images. Note that this lower bound also holds for non-consecutive Doubleline Addressing.

Chapter 5

Conclusion

In this thesis, we study an optimization problem arising in the field of OLED displays. Briefly, there are two different technologies called active matrix and passive matrix. The latter is the cheaper one but such displays suffer from insufficient lifetime. Our goal is to develop an algorithm that increases the lifetime of passive matrix devices, which is essential to achieve mass market penetration. We can extend the lifetime of a standard passive matrix OLED display with our so-called *Consecutive Multiline Addressing* scheme (CMLA). To this end, we need to compute a decomposition of the image, which we want to display such that the stress on the diodes is minimized.

We apply an algorithm engineering approach consisting of concurrent and interacting theoretical analysis and practical evaluation. We capture this process in three categories: the technical background, the theoretical foundations, and the engineering process itself. The former defines the setting of our application and thus it is the basis of the latter two categories. Though the engineering process lies on top of the theoretical foundations, there is a tight interaction between both since the practical evaluations motivate certain directions of theoretical research and conversely theoretical results facilitate the development of suitable algorithms.

We briefly describe the technical or electrical engineering background of the application to the extent that is necessary to relate to the mathematical model of the decomposition problem. That is, we identify certain numerical properties of a given image and determine their relation to the amplitude of the electrical current, which has a major impact on the lifetime of an OLED device. Thereby, we also explain the effectiveness of our CMLA scheme. Moreover, we define the design goals to which an algorithm finally should adhere to allow an economical implementation. Our algorithm is suitable for being implemented at low cost on a chip that drives a passive matrix OLED display, e.g. for a mobile phone. To this end, our algorithm is fully combinatorial, i.e. it only uses operations like addition, subtraction, and comparison, and furthermore the overhead in terms of memory usage is a small fraction of the input size. Moreover, it reacts in realtime and achieves substantial improvements in terms of lifetime. We emphasize that despite meeting all these design goals, our algorithm nevertheless allows for an economical implementation.

On top of the technical background, we formulate the mathematical model of the decomposition problem. We derive an integer linear program where the objective is proportional to the amplitude of the electrical current and the constraints assure a lossless decomposition. The restriction to integer solutions is due to the fact that we should only use fixed precision arithmetic for an economical hardware implementation. However, we may consider the relaxation for a first evaluation of CMLA on real-world images. Thereby, we obtain a first benchmark that promises an advantage in the application if an economical algorithm nearly achieves the same results.

Moreover, we show that there is an equivalent formulation using network flows. That is, we identify a subset of the variables as flow-variables and a much smaller subset considered as variable capacities. We thus establish the foundation for our first fully combinatorial heuristics to compute good feasible solutions not far away from the optimum in practice. This approach is based on iterative separation and augmentation of the capacities until a feasible solution is found. Though this already works well in practice, the experiments show that the bottleneck is the possibly large number of iterations. Hence, further theoretical research to overcome this issue was necessary.

This combinatorial point of view also intuitively enables us to divide our problem into a network design problem, which contains only the variable capacities, and independent TRANSSHIPMENT problems. For the latter, we exploit the special structure of the graphs that appear in our application to come up with a fully combinatorial linear time algorithm for graphs with bounded bandwidth and accordingly numbered nodes. This result is not only of theoretical interest, but also yields very efficient implementations in practice. That is, we first solve the network design problem for the variable capacities and then we lift the feasible solution to the original set of variables by solving TRANSSHIPMENT problems. Hence, it remains to develop efficient (approximation) algorithms for the network design problem, i.e. algorithms running in linear time and ideally passing only once over the input.

To this end, we analyze the complexity of the network design problem under diverse assumptions on the input, e.g. $\{$-1,0,1$\}$-demands. We identify a subset of constraints that allow an exact solution by a fully combinatorial linear-time algorithm. We show the existence of a polynomial-time approximation scheme for unbounded demands on our special graphs. This helps us to better understand the structure of the problem, in particular for the simplest non-trivial case of doubleline addressing. We thereby come up with an approximation algorithm for this special case that passes only once over the graph and still achieves solutions comparable to the separation and augmentation approach.

To further improve the per instance approximation ratios, we investigate the canonical online problem since the new approach also fits in this setting. On the theoretical side, we present an algorithm with tight competitive ratio in the considered special case. We thereby identify situations in which the first implementation of our new approach takes suboptimal decisions. By integrating the gained insights into the algorithm, we finally come up with a fully combinatorial linear time algorithm achieving near optimal solutions for Doubleline addressing in practice. The same ideas can also be ported to higher multiline orders k as indicated by the computational results for $k = 3$.

Furthermore, we present several approaches to cut the memory requirement. We thereby achieve all our design goals with algorithms that are suitable for being implemented in hardware to drive a passive matrix OLED device. An international integrated circuit supplier has invested in CMLA to bring chips containing our algorithms on the market.

Bibliography

[1] T. Tsujimura, W. Zhu, S. Mizukoshi, N. Mori, K. Miwa, S. Ono, Y. Maekawa, K. Kawabe, and M. Kohno, “Advancements and Outlook of High Performance Active-Matrix OLED Displays,” in *SID 2007 International Symposium Digest of Technical Papers* (J. Morreale, ed.), vol. XXXVIII, (Long Beach, USA), pp. 84–88, Society for Information Display, May 2007.

[2] C. Xu, J. Wahl, F. Eisenbrand, A. Karrenbauer, K. M. Soh, and C. Hitzelberger, “Verfahren zur Ansteuerung von Matrixanzeigen,” *Patent 10 2005 063 159, Germany, pending*, 2005.

[3] E. Smith, P. Routley, and C. Foden, “Processing digital data using non-negative matrix factorization,” *patent GB 2421604A, pending*, 2005.

[4] E. C. Smith, “Total Matrix Addressing (TMA^{TM}),” in *SID 2007 International Symposium Digest of Technical Papers* (J. Morreale, ed.), vol. XXXVIII, (Long Beach, USA), pp. 93–96, Society for Information Display, May 2007.

[5] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, pp. 111–126, 1994.

[6] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” *Advances in Neural Information Processing Systems*, vol. 13, 2001.

[7] F. Eisenbrand, A. Karrenbauer, M. Skutella, and C. Xu, “Multiline Addressing by Network Flow,” in *Algorithms - ESA 2006 : 14th Annual European Symposium* (Y. Azar and T. Erlebach, eds.), vol. 4168 of *Lecture Notes in Computer Science*, (Zürich, Switzerland), pp. 744–755, Springer, 2006.

[8] F. Eisenbrand, A. Karrenbauer, and C. Xu, “Algorithms for longer OLED Lifetime,” in *WEA 2007* (C. Demetrescu, ed.), vol. 4525 of *Lecture Notes in Computer Science*, (Rome, Italy), pp. 338–351, Springer, June 2007.

[9] C. Xu, A. Karrenbauer, K. M. Soh, and J. Wahl, “A New Addressing Scheme for PM OLED Display,” in *SID 2007 International Symposium Digest of Technical Papers* (J. Morreale, ed.), vol. XXXVIII, (Long Beach, USA), pp. 97–100, Society for Information Display, May 2007.

[10] C. W. Tang and S. A. V. Slyke, “Organic electroluminescent diodes,” *Appl. Phys. Lett.*, vol. 51, pp. 913 – 915, 1987.

[11] J. H. Burroughes, D. D. C. Bradley, A. R. Brown, R. N. Marks, K. Mackay, R. H. Friend, P. L. Burns, and A. B. Holmes, “Light-emitting diodes based on conjugated polymers,” *Nature*, vol. 347, pp. 539 – 541, 1990.

[12] R. H. Friend, R. W. Gymer, A. B. Holmes, J. H. Burroughes, R. N. Marks, C. Taliani, D. D. C. Bradley, D. A. D. Santos, J. L. Brédas, M. Lögdlund, and W. R. Salaneck, “Electroluminescence in conjugated polymers,” *Nature*, vol. 397, pp. 121 – 128, 1999.

[13] K. M. Soh, C. Xu, and C. Hitzelberger, “Dependence of OLED Display Degradation on Driving Conditions,” *Proceedings of SID Mid Europe Chapter Fall Meeting*, 2006.

[14] A. Lääperi, I. Hyytiänen, T. Mustonen, and S. Kallio, “OLED Lifetime Issues in Mobile Phone Industry,” in *SID 2007 International Symposium Digest of Technical Papers* (J. Morreale, ed.), vol. XXXVIII, (Long Beach, USA), pp. 1183–1187, Society for Information Display, May 2007.

[15] A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*, vol. 24 of *Algorithms and Combinatorics*. Springer Verlag, 2003.

[16] A. Schrijver, *Theory of linear and integer programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[17] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Englewood Cliffs, NJ: Prentice Hall Inc., 1993.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[19] L. Khachiyan, “A polynomial algorithm in linear programming,” *Doklady Akademii Nauk SSSR*, vol. 244, pp. 1093–1097, 1979.

[20] H. L. Bodlaender, “Classes of graphs with bounded tree-width,” *Bulletin of EATCS*, pp. 116 – 128, 1988.

[21] F. R. K. Chung and P. D. Seymour, “Graphs with small bandwidth and cutwidth,” *Discrete Math.*, vol. 75, pp. 113 – 119, 1989.

[22] M. Grötschel, L. Lovász, and A. Schrijver, “The ellipsoid method and its consequences in combinatorial optimization,” *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.

[23] E. A. Dinits, “Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation,” *Soviet Math. Dokl.*, vol. 11, pp. 1277 – 1280, 1970.

[24] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 136 – 146, ACM Press, 1986.

[25] A. V. Goldberg, "Recent Developments in Maximum Flow Algorithms (Invited Lecture)," in *Algorithm Theory - SWAT '98, 6th Scandinavian Workshop on Algorithm Theory, Stockholm, Sweden, July, 8-10, 1998, Proceedings*, vol. 1432 of *Lecture Notes in Computer Science*, pp. 1–10, Springer, 1998.

[26] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[27] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth, "Complexity results for bandwidth minimization," *SIAM Journal on Applied Mathematics*, vol. 34, no. 3, pp. 477–495, 1978.

[28] D. Kratsch and L. Stewart, "Approximating bandwidth by mixing layouts of interval graphs," *SIAM J. Discrete Math.*, vol. 15, no. 4, pp. 435–449, 2002.

[29] T. Hagerup, J. Katajainen, N. Nishimura, and P. Ragde, "Characterizations of k-terminal flow networks and computing network flows in partial k-trees," in *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, (Philadelphia, PA, USA), pp. 641–649, Society for Industrial and Applied Mathematics, 1995.

[30] E. Tardos, "A strongly polynomial algorithm to solve combinatorial linear programs," *Oper. Res.*, vol. 34, no. 2, pp. 250–256, 1986.

[31] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. The MIT Press, 1990.

[32] H. W. Lenstra, "Integer programming with a fixed number of variables," *Mathematics of Operations Research*, vol. 8, no. 4, pp. 538 – 548, 1983.

[33] ILOG S.A., *ILOG CPLEX 10.0 File Formats*, 2006.

[34] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *FOCS*, pp. 300 – 309, 1998.

[35] T. Jung, "Testing multiline addressing algorithms for passive matrix OLED displays." Universität des Saarlandes, April 2007. Bachelor's Thesis.