Technische Universität Berlin

Algorithms for Detecting Block Structures in Matrices

Diplomarbeit

am Fachgebiet Kombinatorische Optimierung und Graphenalgorithmen
Prof. Dr. Marco Lübbecke
Institut für Mathematik
Fakultät II — Mathematik und Naturwissenschaften
Technische Universität Berlin

vorgelegt von

Michael Bastubbe

am

16. September 2011

Die selbständige und eigenhändige Anfertigung
versichere ich an Eides statt.

Berlin, den 16. September 2011

Acknowledgments

At first, I wish to thank my parents and my brother for their support and encouragement during my studies.

I would like to express my gratitude towards my supervisor Prof. Dr. Marco E. Lübbecke. His encouragement and support were crucial to the success of this thesis. I am truly grateful for the great degree of freedom I enjoyed in exploring my ideas. This way I could discover how doing mathematics works and made the creation of this thesis a unique experience.

Finally, I want to thank my friends and fellow students for making the creation of this thesis a much more enjoyable process. It motivated me a lot to spend vast working hours and many helpful discussions with Stefan Müller. I wish to thank Alexander Richter, Sebastian Schenker, Marlen Schwengfelder and Martin Trapp for reading earlier versions of this thesis and many valueable suggestions. I thoroughly enjoyed working with you.

All remaining errors were overlooked by myself during the final review.

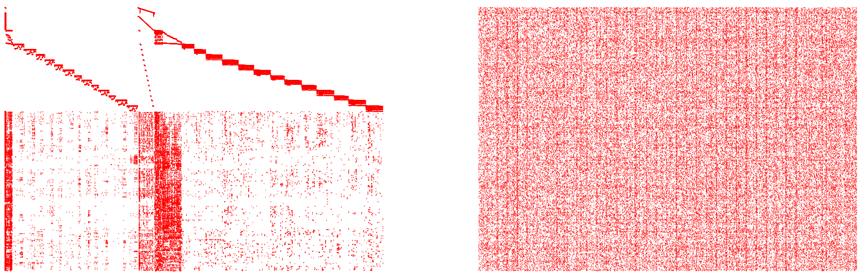
Contents

1 Introduction

What do society, emotions, public transport, and a desktop have in common? – All of them can drown in chaos. It does not matter whether we consider anarchy, hate and love, train cancellation, or an inspiring working environment, there is a notion for almost every object to describe its state as being disordered.

Occasionally chaos seems to be abhorred in mathematical optimization. In order to mention two examples from practice: Local train schedules are coordinated to minimize the changing time and communication networks are designed such that capacity and survivability constraints are satisfied while costs are minimized. The coordination of single components in a complex system cannot only safe money and time, it sometimes also decreases the perceived chaos. For many people a priceless feeling.

The mathematical problems behind these real-life applications can be formulated as a mixed integer program(MIP). Mixed Integer Programming is a powerful tool to model large-scale combinatorial optimization problems. The coefficient matrix used by this approach encodes the problem data and hence includes much information about the structure of the problem. Consider for example the coefficient matrix of the mixed integer programming instance msc98-ip.mps, modeling an telecommunication network design problem, taken from MIPLIB 2010 [30] in Figure 1.1a. This matrix has 15850 rows and 21143 columns with 92918 nonzero entries. All nonzero entries are marked with a red dot.



(a) original (b) randomly permuted

Figure 1.1: Coefficient matrix of msc98-ip.mps

At first sight, it looks like this matrix is divided into two parts. About the first third of the matrix seems to have a special structure, but the remaining matrix appears rather random. Roughly speaking, this is due to the fact that the nonzero entries of the right and left upper part are arranged blockwise. Rows and columns belonging to one block have no nonzero entries in columns and rows, respectively, that belongs to another block (at least, if one only considers the left or right upper third of the matrix). This block structure reveals that there are disjoint sets of rows and columns that are somehow connected with each other, caused by the structure of the problem. The first crucial question arises:

Is it possible to exploit block structures in the coefficient matrix of a MIP? (1.1)

We have not stated what we exactly mean by "block structures" and therefore we should ask a better question:

How should a coefficient matrix of a MIP look like, such that the structure of its nonzero entries can be exploited to improve the performance? (1.2)

To answer this question we should take a look at the algorithms to solve Mixed Integer Programs. The standard method to solve a MIP, like msc98-ip.mps, is the so-called branch-and-bound algorithm. This algorithm, in its original form, does not exploit possible structure of the nonzero entries in the coefficient matrix. Hence, the order of the rows and columns does not matter from a theoretical point of view. In Figure 1.1b the coefficient matrix of msc98-ip.mps is displayed once more, but the rows and columns are randomly permuted this time. This matrix is probably one more notion for chaos to most people and thus, at least, it seems that information is lost by not exploiting the structure of the matrix.

However, in this thesis we want to study two kind of block forms: The *k-arrowhead form* and the *bordered k-block diagonal form*. Throughout this introduction, we will refer to them as "block forms". Instead of giving a proper definition here, we illustrate them by the coefficient matrix of msc98-ip.mps: Its rows and columns are permuted[1] such that it is in 24-arrowhead form in Figure 1.2a and in bordered 6-block diagonal form in Figure 1.2b.

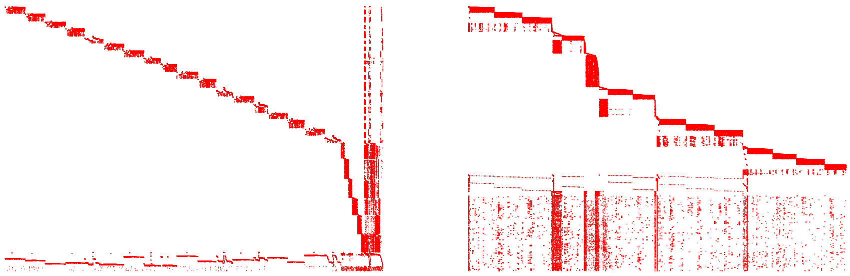
Fortunately, algorithmic approaches exploiting block structures in coefficient matrices of MIPs were recently developed [18] and are still emerging. Moreover, there are further mathematical problems for which it can be exploited that the coefficient matrix is in one of these forms. We will sketch how this can be done for solving two well-known problems from linear algebra. However, it is not our purpose to answer questions 1.1 and 1.2 in detail, rather we wish to investigate the problem of finding permutations of the rows and columns of a matrix such that we obtain matrices in k-arrowhead form or bordered k-block diagonal form.

In this context the next question arises:

How can permutations yielding a matrix in one of both block forms can be characterized? (1.3)

To answer Question 1.3, we are going to introduce the concept of a *k-decomposition* of a matrix that is essentially a partition of the rows and columns. In this way, a permutation

[1] These permutations were found by the matrix decomposition tool *Decomp* implemented in the course of this thesis.



(a) 24-arrowhead form (b) 6-bordered block diagonal form

Figure 1.2: Coefficient matrix of msc98-ip.mps

of the rows and columns can be encoded. On the one hand, we will present sufficient conditions for a k-decomposition to yield a matrix in one of the block forms with k blocks. On the other hand, it will turn out that every permutation that yields a matrix in one of the block forms with k blocks, can be expressed by a k-decomposition up to the order of the rows and columns inside the blocks.

We use this characterization to formulate two optimization problems, one for each block form, namely MinAf and MinBf, whose objective functions are motivated by the applications introduced in Section 2.2. Naturally, more questions appear:

How can we solve MinAf and MinBf? (1.4)

Can we hope for a polynomial time algorithm to solve them? In other words, are these problems $\mathcal{NP}$-hard[2]? (1.5)

We will find out that both problems can be solved in polynomial time for fixed objective function value. However, they are $\mathcal{NP}$-hard in general, and hence we are going to investigate two kind of algorithms.

At first, we are interested in finding heuristics, that obtain feasible solutions for a reasonable number of instances of acceptable quality in moderate time. In fact, we will introduce for each block form two algorithmic approaches. These approaches have many structural similarities. All of them include the solution of a graph partitioning problem in a special graph or hypergraph. We will only touch a few aspects of graph partitioning in this thesis.

Secondly, we want to state an integer program that solves this problems exactly. It will turn out that this model has two major flaws: On one hand, the model is highly symmetric[3]. We will introduce two types of constraints that reduce symmetry. On the other hand, the LP-relaxation will turn out to be weak. We are going to introduce another integer program that is based on the old one, but has exponentially many variables. A

[3]Roughly speaking, an integer program is symmetric if its variables can be permuted without changing the structure of the problem. For a thorough treatment, we refer the reader to [34].

column generation approach is presented to solve the LP-relaxation of the new model. However, it is not our purpose to indicate a branch-and-price algorithm to solve the integer program, rather we will restrict our attention to the LP-relaxation. In order to solve it, we introduce the pricing problem and show that it can be solved in polynomial time under some conditions. Moreover, we present an integer program that solves the pricing problem in general.

Most approaches are implemented and tested in the course of this thesis. We are going to compare the heuristic approaches by quality. In order to do so, we will define alternative quality measures that are motivated by applications and study extensively which approach performs best with respect to one of the quality measures. Furthermore, we will compare our results with some former experiments of Ferris and Horn [17]. Moreover, we study which instances from practice (namely the Netlib and MIPLIB 2010 [30]) have a coefficient matrix that can be permuted to k-arrowhead or bordered k-block diagonal form for different k by our implementation. Furthermore, we study the performance and limitation of the first integer program. Finally, we present our computational results for the LP-relaxation of the second model.

Outline

The remaining part of this diploma thesis is structured as follows. Chapter 2 contains the mathematical background. It provides definitions of the block forms and exemplifies by some applications from mathematics how they can be exploited. It develops the concept of a *k-decomposition* that can be used to characterize permutations of the rows and columns that yield a matrix in a block form. The definitions of the corresponding mathematical optimization problems MinAf and MinBf are provided. It also defines three quality measures motivated by the applications. It ends with a brief review of the relevant literature. Chapter 3 contains an analysis of the complexity of both problems. A polynomial time algorithm for fixed objective function value is presented for each problem. Then it provides proofs that some special cases of MinAf and MinBf are $\mathcal{NP}$-hard. Chapter 4 gives an introduction to graph partitioning. It then introduces two heuristics for each problem based on solving different graph partitioning problems. It also provides relevant examples of failed runs for every heuristic. Chapter 5 deals with exact solving methods for MinAf and MinBf. It introduces two integer programs that can be used to solve MinAf and can easily be adapted to solve MinBf. It deals with the pricing problem that has to be solved to handle the second model that has exponentially many variables. Chapter 6 includes a documentation and analysis of our computational experiments for the heuristics and the exact solving methods.

Finally, we want to point out that we do not share the opinion that chaos is abhorred in mathematical optimization and that we do not abhor it either; instead we follow the words of the renowned Dutch graphic artist M.C. Escher:

"We adore chaos because we love to produce order."

2 Background

In the introduction, we have already seen two matrices whose nonzero entries are arranged in one of the proposed block forms. Now we will give a proper definition of these two forms. We develop the concept of *k-decomposition* that can be used to characterize permutations of the rows and columns yielding such forms. Furthermore, we present optimization problems for each form that are motivated by applications from mathematics.

To be more precise, the first chapter consists of five sections. Initially, we are going to introduce and illustrate basic definitions in Section 2.1. In particular, we will introduce two block structures for matrices, namely the *k-arrowhead form* and the *bordered k-block diagonal form*. Secondly, we will show potential advantages offered by matrices in such forms offer. This will be done by presenting some applications in Section 2.2. In the next section we introduce the concept of *k-decomposition* that will turn out to be useful for characterizing a matrix in k-arrowhead form or bordered k-block diagonal form. How well the block structure can be exploited obviously depends on the properties of the decomposed matrices. In Section 2.4 we define functions that measure the quality of a decomposition with respect to one of these properties. Eventually, in Section 2.5.1 of this chapter we give an overview about the literature on detecting block structures in matrices and compare the different approaches.

2.1 Definitions

At first, we introduce general notions about the structure of matrices. Secondly, we define two block forms for matrices: The k-arrowhead form and a specialization of it, the bordered k-block diagonal form for an integer $k \in \mathbb{N}$.

Throughout this thesis, we are talking about matrices and the structure of its nonzero entries. If not stated otherwise, $A \in \mathbb{R}^{m\times n}$ is a matrix with entries a_{ij} for a row $i \in \{1,\dots m\}$ and a column $j \in \{1,\dots n\}$. For the rows and the columns of a matrix we want to use the following terms: If a row i has a nonzero entry in the distinct columns j_1 and j_2, we say that the columns j_1 and j_2 are *coupled* or *linked* by i. We also say that j_1 and j_2 have a *common row* i, if j_1 and j_2 are coupled by i. Similarly, if a column j has a nonzero entry in the distinct rows i_1 and i_2, we say that the rows i_1 and i_2 are *coupled* or *linked* by j. If i_1 and i_2 are linked by j, we call j a *common column* of i_1 and i_2.

We denote $[i] := \{1,\dots,i\}$ as the set of the first i natural numbers for $i \in \mathbb{N}$. Moreover, we will use certain submatrices of a matrix whose rows and columns are permuted. Consider a subset of rows $\{r_1, r_2, \dots, r_{m'}\} \subseteq [m]$ and a subset of columns $\{c_1, c_2, \dots, c_{n'}\} \subseteq \{1,\dots,n\}$ of a matrix $A \in \mathbb{R}^{m\times n}$. We denote the *permuted submatrix* of A by $A[r_1, r_2, \dots, r_{m'}; c_1, c_2, \dots, c_{n'}] \in \mathbb{R}^{m'\times n'}$, it includes rows and columns of A such that its i-th row is the r_i-th row of A and its j-th column is the c_j-th column of A with $m', n' \in \mathbb{N}$, $m' \leq m$, $n' \leq n$. For the sake of clarity, we give a small example:

Example 2.1.1
Consider the matrix $A \in \mathbb{R}^{4\times 4}$ such that

$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 2 & 0 & 1 & 0 \\ 2 & 1 & 0 & 2 \\ 4 & 2 & 5 & 42 \\ 0 & 3 & 0 & 1 \end{pmatrix} \end{array}.$$

The permuted submatrix $A[3,1;4,1,2] \in \mathbb{R}^{2\times 3}$ is then given by:

$$A[3,1;4,1,2] = \begin{array}{c c} & \begin{array}{ccc} 4 & 1 & 2 \end{array} \\ \begin{array}{c} 3 \\ 1 \end{array} & \begin{pmatrix} 42 & 4 & 2 \\ 0 & 2 & 0 \end{pmatrix} \end{array}.$$

Furthermore, let P be a set and $k \in \mathbb{N}$ be an integer. A *weak partition* of P is a set $\{P_1, \dots, P_k\}$ of subsets of P with $\bigcup_{i=1}^{k} P_i = P$ and $P_i \cap P_j = \emptyset$ for $i, j \in [k], i \neq j$. The set P_i is called *part* of P for every $i \in [k]$. If $\{P_1, \dots, P_k\}$ is a weak partition of P and $P_i \neq \emptyset$ for all $i \in [k]$, then we call $\{P_1, \dots, P_k\}$ a *partition* of P.

Moreover, we will make use of *tuples*. A *tuple* is a set whose elements are ordered. We will note tuples in parentheses "$(\dots)$" instead of curly brackets "$\{\dots\}$", and denote the *empty tuple* by $()$ and $\emptyset$. Every tuple is still a set and thus the common notation for sets extends to tuples naturally.

It is time to define the first block form:

Definition 2.1.2 (k-arrowhead form)
Let $A \in \mathbb{R}^{m\times n}$, $k \in \mathbb{N}_0$. We say A is in *k-arrowhead form* or *k-doubly-bordered block diagonal form*, if

$$A = \begin{pmatrix} B_1 & & & & C_1 \\ & B_2 & & & C_2 \\ & & \ddots & & \vdots \\ & & & B_k & C_k \\ R_1 & R_2 & \cdots & R_k & D \end{pmatrix},$$

with $B_i \in \mathbb{R}^{m_i \times n_i}$, $R_i \in \mathbb{R}^{r\times n_i}$, $C_i \in \mathbb{R}^{m_i\times c}$, $D \in \mathbb{R}^{r\times c}$ with $r, c \in \mathbb{N}_0$ and $m_i, n_i \in \mathbb{N}$ for $i = 1, ..., k$ and all other entries equal zero.

Remark 1:
It is necessary to restrict m_i and n_i for $i \in [k]$ to be strictly positive. Otherwise every matrix would be in k-arrowhead form for every $k \in \mathbb{N}_0$. For example, we could set $B_1 := A$ and leave the remaining matrices $B_2, \dots, B_k$ empty.

Observation 2.1.3
Every matrix is in 0*-arrowhead form and* 1*-arrowhead form.*

This can easily be seen by setting $D = A$ or setting $B_1 = A$, respectively.

Observation 2.1.4
Let $k \in \mathbb{N}_0$ be a nonnegative integer and $A \in \mathbb{R}^{m \times n}$ be a matrix in k-arrowhead form. Then A is also in r-arrowhead form with $r \in \mathbb{N}_0$ and $r \leq k$.

This is clear since we can merge two submatrices B_i and B_{i+1} to one single submatrix for $i \in [k-1]$.

Every submatrix B_i is called a *block*. The number of blocks is denoted by k. We call the submatrix

$$\begin{pmatrix} R_1 & R_2 & \cdots & R_k & D \end{pmatrix},$$

the *row border*. Every row of it is called *border row* or *coupling constraint* . Now we extend the general notation of the nonzero structure of a matrix to blocks. If a row l has a nonzero entry in block i and block j, we say that the distinct blocks i and j are *coupled* or *linked* by l. We notice that for a matrix in k-arrowhead form only a border row can couple two distinct blocks.

Similarly, we call the submatrix

$$\begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_k \\ D \end{pmatrix},$$

the *column border*. Its columns are called *border columns*. If a column q has a nonzero entry in block i and block j, we say that the blocks i and j are *coupled* or *linked* by q. As for rows we notice that in a matrix in k-arrowhead form a column that couples two distinct blocks is a border column.

Note that r and c may equal zero. On the one hand, if $c = 0 \neq r$, the border columns are missing. On the other hand, if $r = 0 \neq c$ the border rows are missing. In both cases the matrix is called *singly-bordered block diagonal form*. If $r = c = 0$ the matrix is called to be in *block diagonal form*. We are especially interested in the *singly-bordered block diagonal form* with empty column border :

Definition 2.1.5 (bordered k-block diagonal form)
Let $A \in \mathbb{R}^{m \times n}$, $k \in \mathbb{N}$. If A is in k-arrowhead form with empty column border, we say that A is in *bordered k-block diagonal form*.

A matrix A in bordered k-block diagonal form looks like this:

$$A = \begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_k \\ R_1 & R_2 & \cdots & R_k \end{pmatrix},$$

with $B_i \in \mathbb{R}^{m_i \times n_i}$, $R_i \in \mathbb{R}^{r \times n_i}$ for $i = 1, ..., k$ and all other entries equal zero. Note that if A is in bordered k-block diagonal form, then A^T is in singly-bordered block diagonal form with empty row border.

Before we present some applications that can exploit matrices in k-arrowhead form and bordered k-block diagonal form, we want to introduce some basic graph theoretical definitions.

2.1.1 Basic Definitions From Graph Theory

Throughout this thesis, we will make use of *graphs* and their generalizations: *hypergraphs.* So we will give a brief summary about the most important notions.

A *hypergraph* $\mathcal{H} = (\mathcal{N}, \mathcal{E})$ consists of a finite set of *nodes* $\mathcal{N}$ and a finite set of *hyperedges* $\mathcal{E}$. Nodes are also called *vertices.* Every hyperedge e connects a subset of nodes $\mathcal{N}_e \subseteq \mathcal{N}$, with $|\mathcal{N}_e| \geq 2$. For $s \in \mathcal{N}_e$, we also write $s \in e$ and say that s and e are *incident.* The *size* of e is $|\mathcal{N}_e|$. We say that the distinct nodes s and t are *adjacent* if there is a hyperedge e that is incident to both nodes. For a node s we call adj(s) the set of adjacent nodes of s. The *degree* of a node s is $|adj(s)|$. A hypergraph whose hyperedges have size exactly two is called *undirected graph.* The hyperedges of an undirected graph are simply called *edges.* For the sake of clearness, we will denote the edges of an undirected graph by E and the vertices by V.

We will also need *directed graphs.* A *directed graph* $G = (V, A)$ consists of a set of nodes V and a set of directed edges A. Directed edges are also called *arcs.* An arc is an ordered pair (i, j) of distinct nodes $i, j \in V$.

For an arbitrary hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{E})$ with $\mathcal{N} = \{v_1, ..., v_m\}$ and $\mathcal{E} = \{e_1, ..., e_n\}$, we define its *incidence matrix* $A^{\mathcal{H}} \in \mathbb{R}^{m \times n}$ with entries a_{ij} such that $a_{ij} = 1$ if $v_i \in e_j$ and $a_{ij} = 0$ otherwise.

Now we want to give some applications for motivation.

2.2 Applications

In this section we show that one can often exploit a coefficient matrix of a problem that is in k-arrowhead form or bordered k-block diagonal form to solve the problem. At first, we want to motivate the reader by presenting two well-known problems from linear algebra, where we can take advantage of block structures. For an introduction to linear algebra, we recommend [38]. Afterwards, we refer the reader to literature that covers techniques to exploit a coefficient matrix in bordered k-block diagonal form for solving linear and mixed integer programs. Finally, we sketch an approach that transforms a problem whose coefficient matrix is in arrowhead form into bordered k-block diagonal form.

2.2.1 Systems of Linear Equations

One of the most popular problems of linear algebra is solving a system of linear equations.

LINEAR EQUATIONS
Instance: A matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$
Solutions: $x \in \mathbb{R}^n$ with $Ax = b$

We call the matrix A the *coefficient matrix* and vector b the *right hand side* of the problem. In practice, the LINEAR EQUATIONS problem is solved by two kind of algorithms. There are direct methods, like *LU factorization* and iterative algorithms, like the *Gauss-Seidel* method. These algorithms can be adapted such that if one apply them to a coefficient matrix in k-arrowhead form one can take advantage of the special block structure. For a detailed view on how to adapt *LU* factorization and the Gauss-Seidel method to exploit the k-arrowhead form by parallel computations, we refer the reader to [31].

In the following, we want to have a look at *LU* factorization and how to exploit a coefficient matrix in k-arrowhead form by parallelizing computations in a three-phase approach. *LU* factorization can be managed by applying *elementary row operations* to a matrix A such that an *upper triangular* form is reached. We will call the set of *upper triangular* matrices UT. These elementary row operations are *row permutations* and *addition of a multiple* of a row to another row situated below the first row. *Row permutations* correlate to multiplication from the left with a matrix that has exactly one entry in every row and column which equals one. Such a matrix is called *permutation matrix.* Note, that permutation matrices are invertible and that the product of two permutation matrices is a permutation matrix. The second mentioned operation corresponds to multiplication from the left with a *lower triangular* matrix with all diagonal entries equal one. We call the set of all these matrices LT_1. We notice that the product of two matrices which are both in LT_1 is also in LT_1 and that the matrices in LT_1 are *invertible.* We can apply *elementary row operations* to factorize a matrix $A \in \mathbb{R}^{m \times n}$ like this:

$$LU = PA,$$

with $L \in \mathbb{R}^{m \times m}$ is in LT_1, $U \in \mathbb{R}^{m \times n}$ is in UT and $P \in \mathbb{R}^{m \times m}$ is a *permutation matrix.*

For a matrix in k-arrowhead form, one could use the following three-phase approach to solve LINEAR EQUATIONS:

1. Parallel factorization of each block.
2. Permutation of the unfactored rows and columns to their border.
3. Factorization of the unfactored rows.

In phase one, we factorize each block B_i and also apply the corresponding *elementary row operations* to the associated column submatrix C_i obtaining C_i'. In this phase we would benefit from around equally sized blocks because the time needed by phase one is determined by the makespan of the parallel computations.

Let $A \in \mathbb{R}^{m\times n}$ be in k-arrowhead form. After applying phase one to A, we get the partially factorized matrix

$$P_1 A = A_1 = \begin{pmatrix} L_1U_1 & & & & C_1' \\ & L_2U_2 & & & C_2' \\ & & \ddots & & \vdots \\ & & & L_kU_k & C_k' \\ R_1 & R_2 & \cdots & R_k & D \end{pmatrix},$$

such that the factorization of block $i \in [k]$ is $P_i' B_i = L_i U_i$ with $L_i \in \mathbb{R}^{m_i \times m_i}$ is in LT_1, $U_i \in \mathbb{R}^{m_i \times n_i}$ is in UT, $P_i' \in \mathbb{R}^{m_i \times m_i}$ is a permutation matrix, A_1 is the permutation of A after phase one and P_1 the *permutation matrix* that corresponds to all *row permutations* that were done in phase one. Each of these row permutations correspond to some permutation matrix P_i'.

Since the blocks are not necessarily quadratic and full-ranked, the matrix

$$U = \begin{pmatrix} U_1 & & & \\ & U_2 & & \\ & & \ddots & \\ & & & U_k \end{pmatrix}$$

is not an upper triangular matrix with all diagonal entries unequal zero, in general. In the second phase we permute all those rows and those columns that avoid this fact to the respective border. We get the following:

$$P_2 P_1 A P_2' = P_2 A_1 P_2' = A_2 = \begin{pmatrix} L^*U^* & C^* \\ R^* & D^* \end{pmatrix},$$

with P_2 and P_2' *permutation matrices*, U^* in UT with only nonzeros on its diagonal, L^* in LT_1, A_2 the permutation of A after phase two and the submatrices R^*, C^* and D^* which consist of the old border rows and columns, and the unfactored rows or columns of phase one.

Since U^* is in UT and all its diagonal elements are unequal zero, it is straightforward to factorize the remaining rows. Finally, we get the following factorization:

$$PAP_2' = LU,$$

with $L \in \mathbb{R}^{m\times m}$ is in LT_1, $U \in \mathbb{R}^{m\times n}$ is in UT and P, $P_2' \in \mathbb{R}^{n\times n}$ are *permutation matrices*.

We have seen that it is possible to parallelize the computations of the blocks in phase one. Therefore, it is favorable to do as much factorization as possible in phase one. We also want the borders to be as small as possible, in order to reduce the computation time in phase three. Because the computation time of phase one is determined by the makespan of the block factorizations, we prefer many equally sized blocks for a good work load distribution.

2.2.2 Least Squares

Now we want to have a short look at one of the fundamental problems of numerical linear algebra.

LEAST SQUARES ***Instance:*** A matrix $A \in \mathbb{R}^{m \times n}$ with $m > n$ and a vector $b \in \mathbb{R}^m$ ***Solution:*** $x \in \mathbb{R}^n$ ***Objective:*** Minimize $\|Ax - b\|_2$

The LEAST SQUARES problem is often solved with *QR factorization*. In this method, a matrix $A \in \mathbb{R}^{m \times n}$ is factored into an orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ and an upper triangular matrix $R \in \mathbb{R}^{n \times n}$ with all diagonal entries are not negative:

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

We can get a solution by solving $Rx = b'$, where $b' \in \mathbb{R}^n$ consists of the first n components of b. For a detailed view on LEAST SQUARES we recommend [11]. For a matrix $A \in \mathbb{R}^{m \times n}$ where A^T is in bordered k-block diagonal form we can use a three-phase-approach, that is similar to the one we have utilized for *LU* factorization:

1. Parallel factorization of each block.
2. Permutation of unfactored rows to the row border.
3. Factorization of the unfactored rows.

Given a matrix A with A^T in bordered k-block diagonal form :

$$A = \begin{pmatrix} B_1 & & & & C_1 \\ & B_2 & & & C_2 \\ & & \ddots & & \vdots \\ & & & B_k & C_k \end{pmatrix}.$$

In the first phase we simultaneously factorize B_i and the corresponding *border columns* in C_i:

$$\begin{pmatrix} B_i & C_i \end{pmatrix} = Q_i \begin{pmatrix} R_i & S_i \\ 0 & C'_i \end{pmatrix}, \text{ for } i = 1, \dots, k$$

Q_i is an orthogonal matrix and R_i is an upper triangular matrix with nonnegative diagonal elements. After phase two, we only have to factorize

$$C' = \begin{pmatrix} C'_1 & C'_2 & \dots & C'_k \end{pmatrix}^T.$$

Similar to the *LU* decomposition, we take most advantage from the parallelization, if the matrix in bordered k-block diagonal form has two properties:

- Many equally sized blocks, for a balanced computational work distribution in phase one.
- A small border to keep C' small which grants a fast factorization in phase three.

2.2.3 Linear Programming

Consider the well-known problem of solving a linear program (LP):

LINEAR PROGRAMMING
Instance: A matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$, and a vector $c \in \mathbb{R}^n$
Solution: $x \in \mathbb{R}^n_{\geq 0}$ with $Ax \geq b$
Objective: Minimize $c^T x$

One can exploit coefficient matrices of linear programs in bordered k-block diagonal form by *Dantzig-Wolfe decomposition*. It would go beyond the scope of this thesis to study Dantzig-Wolfe decomposition in detail. For a deep discussion of Dantzig-Wolfe decomposition applied to linear programming problems, we refer the reader to the book of Bertsimas and Tsitsiklis [9, Sec.6.4].

2.2.4 Mixed Integer Programming

A natural generalization of linear programming is MIXED INTEGER PROGRAMMING:

MIXED INTEGER PROGRAMMING
Instance: A matrix $A \in \mathbb{R}^{m \times n}$, a vector $b \in \mathbb{R}^m$, a vector $c \in \mathbb{R}^n$, and a subset $I \subseteq [n]$
Solution: $x \in \mathbb{R}^n_{\geq 0}$ with $Ax \geq b$ and $x_j \in \mathbb{Z}$ for all $j \in I$
Objective: Minimize $c^T x$

Instances of the MIXED INTEGER PROGRAMMING problem are called *mixed integer programs*. One can also exploit the coefficient matrices of mixed integer programs in bordered k-block diagonal form by Dantzig-Wolfe decomposition. This is scoped by the diploma thesis of Gerald Gamrath [18] and the current work of Bergner et al. [8].

2.2.5 Transformation k-Arrowhead to Bordered k-Block Diagonal Form

Consider an arbitrary mixed integer program J, i.e. a matrix $A \in \mathbb{R}^{m \times n}$, a vector $b \in \mathbb{R}^m$, a vector $c \in \mathbb{R}^n$, and a subset $I \subseteq [n]$. Moreover, suppose A is in k-arrowhead form for some $k \in \mathbb{N}$. We can construct another mixed integer program J' which consists of a matrix $A' \in \mathbb{R}^{m' \times n'}$, a vector $b' \in \mathbb{R}^{m'}$, a vector $c \in \mathbb{R}^{n'}$, and a subset $I' \subseteq [n']$ such that A' in bordered k-block diagonal form and moreover J and J' are equivalent (i.e. there is a bijection between the feasible solutions of both instances, that maintains the objective function value).

Let $C \subseteq [n]$ be the set of border columns with $c := |C|$. Let $\mathcal{B}_j$ be the set of blocks that has at least one nonzero entry in border column $j \in C$ and define $n_j := |\mathcal{B}_j|$. In the following we are going to sketch a procedure to obtain J'. The main idea is to add copies of each border column j whose corresponding variables should attain the same value. In fact, for a border column j we will add n_j copies, one for each block that has a nonzero entry in j. We initialize J' by setting $I' := I$, $b' := b$, $c' := c$ and $I' := I$. The procedure consists of three main steps:

At first, we successively add a new column to A' for every pair (j, t) with $j \in C$ and $t \in \mathcal{B}_j$. The column added for the pair (j, t) is denoted by $s_{(j,t)}$. It becomes the new last column of block t. The entries of $s_{(j,t)}$ in the rows of block t are the same entries that j has in the rows of block t. All other entries of $s_{(j,t)}$ are zero, except if $s_{(j,t)}$ is the first column that is added for j. If $s_{(j,b)}$ is the first column that is added for j, then it also has the same entries in the border rows that j has and the cost coefficient of $s_{(j,b)}$ is c_j. Moreover, $s_{(j,t)} \in I'$ if and only if $j \in I$.

Secondly, we delete all border columns $j \in C$ from J'.

Finally, we successively add rows to the border of A' that ensure that for every $j \in C$ the variables that belongs to one of the columns $s_{(j,t)}$ for some $t \in \mathcal{B}_j$ has the same value. For $j \in C$ we add $2 \cdot (|\mathcal{B}_j| - 1)$ many rows. Let $j \in C$ be fixed and consider $t_{j1}, \ldots, t_{jn_j}$ the elements of $\mathcal{B}_j$. For $p \in [n_j - 1]$ we add two rows to the border rows of A'. The first one has in column $s_{(j,t_{jp})}$ an entry of -1 and in column $s_{(j,t_{j(p+1)})}$ an entry of 1. All other entries are 0. The corresponding entry in b' is 0. The second one has in column $s_{(j,t_{jp})}$ an entry of 1 and in column $s_{(j,t_{j(p+1)})}$ an entry of -1. All other entries are 0 again. The corresponding entry in b' is also 0. Thus, the variables corresponding to the columns $s_{(j,t_{j(p+1)})}$ and $s_{(j,t_{jp})}$ attain the same value for all $p \in [n_j - 1]$. Hence, the variables of the columns $s_{(j,t_{j1})}, \ldots, s_{(j,t_{jn_j})}$ attain the same value.

It is easily seen that setting a variable x_j of J for $j \in C$ to the value of one of its copies in J' yields a bijection that maintains the objective function value.

2.3 Problem Formulations

In this section our goal is to introduce the mathematical problems MinBF and MinAF. In order to do so, we think about a good characterization of a matrix in k-arrowhead form and bordered k-block diagonal form. Afterwards, we ask how to measure the quality of a decomposition. We will start with a small example:

Example 2.3.1
The nonzero entries, marked with 'X', of the following 9×16 *matrix are in a "MESS":*

$$\begin{array}{rc}
\begin{array}{c} 1\\2\\3\\4\\5\\6\\7\\8\\9 \end{array} &
\begin{pmatrix}
X&&&&&&X&&&&X&X&X&&& \\
X&X&&&&X&X&&&&X&&&&& \\
X&X&X&&X&X&X&&&&X&X&X&&& \\
X&&X&X&X&&X&&&&&&X&&& \\
X&&&X&&&X&X&X&X&X&X&X&X&X&X \\
X&&&&&&X&X&&&&&&X&& \\
X&&&&&&X&X&X&X&&&&X&X&X \\
&&&&&&&X&&&&&&&&X \\
&&&&&&&X&X&X&&&&X&X&X
\end{pmatrix} \\
& \begin{array}{cccccccccccccccc} 1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16 \end{array}
\end{array}$$

By permuting the rows and columns, one can obtain a matrix in 2-arrowhead form. Column 1 and 7 have moved to the column border and row 5 has moved to the row border. Furhermore, the remainung columns are permuted blockwise:

$$\begin{array}{rc}
\begin{array}{c} 1\\2\\3\\4\\6\\7\\8\\9\\5 \end{array} &
\begin{pmatrix}
&&&&&X&X&X&&&&&&&X&X \\
X&&&&X&X&&&&&&&&&X&X \\
X&X&&X&X&X&X&X&&&&&&&X&X \\
&X&X&X&&&&X&&&&&&&X&X \\
&&&&&&&&X&&&X&&&X&X \\
&&&&&&&&X&X&X&X&X&X&X&X \\
&&&&&&&&X&&&&&X&& \\
&&&&&&&&X&X&X&X&X&X&& \\
&&X&&&X&X&X&X&X&X&X&X&X&X&X
\end{pmatrix} \\
& \begin{array}{cccccccccccccccc} 2&3&4&5&6&11&12&13&8&9&10&14&15&16&1&7 \end{array}
\end{array}$$

By definition 2.1.2 we can identify a k-arrowhead form only by looking at the matrix and identifying the blocks. Instead of verifying that a matrix is in k-arrowhead form or bordered k-block diagonal form by inspection, we should define a structure that specifies which rows and columns are included in the submatrices $B_1, \dots, B_k$.

2.3.1 Characterization of a Decomposition

In this subsection we want to develop some concepts to characterize a decomposition of a matrix. This subsection is rather technical and should give an idea how one could implement a data structure for matrices in block forms. From an algebraic point of view one would consider permutations of the rows and columns:

$$P_1 A P_2 = A',$$

with $P_1 \in \mathbb{R}^{m \times m}$ and $P_2 \in \mathbb{R}^{n \times n}$ are permutation matrices, A and A' are $m \times n$ matrices such that A' is in k-arrowhead form.

Another possibility to express permutations are bijective functions $\sigma : [m] \to [m]$ and $\tau \colon [n] \to [n]$. We could search for some σ and τ as defined above such that the i-th row of A is the $\sigma(i)$-th row of A' and the j-th column of A is the $\tau(j)$-th column of A'

However, since the order of the rows and columns inside the blocks does not matter, it is actually sufficient to give a partition for the rows and columns into the blocks and the respective border. We will naturally obtain a permutation of the rows and columns from that partition.

Definition 2.3.2 (*k*-decomposition of a matrix)
Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $k \in \mathbb{N}$. We call a pair $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ a *k-decomposition* of A if the tuple $\mathcal{R} = (\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the set of rows $[m]$ and the tuple $\mathcal{C} = (\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B)$ is a weak partition of the set of columns $[n]$ of A.

We call the sets $\mathcal{R}_1, ..., \mathcal{R}_k$ *row blocks* and the sets $\mathcal{C}_1, ..., \mathcal{C}_k$ *column blocks*. The sets $\mathcal{R}_B$ and $\mathcal{C}_B$ are called *row border part* and *column border part*, respectively.

Remark 2:
As the names already suggest, the elements of $\mathcal{R}_i$ and $\mathcal{C}_i$ will be the rows and columns, respectively, that belongs to block i for some $i \in \{1, \dots, k\}$. Furthermore, the elements of $\mathcal{R}_B$ and $\mathcal{C}_B$ will be the rows and columns that belongs to the respective border.

Now we explain how to obtain permutations of the rows and columns from these partitions. We assume w.l.o.g. that for $t \in [k]$ the elements of each of the sets $\mathcal{R}_t$, $\mathcal{R}_B$, $\mathcal{C}_t$ and $\mathcal{C}_B$ are sorted in ascending order by the index of the rows or columns. Otherwise, we sort them accordingly. For this purpose, we will denote them as tuples.

Therefore, by consecutive numbering of all rows in the tuple of tuples $(\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B)$ we obtain a uniquely determined permutation of the rows $\sigma^R\colon \{1, \dots, m\} \to \{1, \dots, m\}$ with $\sigma^R(i_1) < \sigma^R(i_2)$ for $i_1 \in \mathcal{R}_{b_1}$ and $i_2 \in \mathcal{R}_{b_2} \cup \mathcal{R}_B$ with $b_1 < b_2$. In other words, the rows are ordered blockwise by their values of σ^R .

Similarly, we get a unique permutation of the columns $\sigma^C\colon \{1, \dots n\} \to \{1, \dots n\}$ with $\sigma^C(j_1) < \sigma^C(j_2)$ for $j_1 \in \mathcal{C}_{b_1}$ and $j_2 \in \mathcal{C}_{b_2} \cup \mathcal{C}_B$ with $b_1 < b_2$ by consecutive numbering of all columns in the tuple of tuples $(\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B)$.

We call σ^R the *row permutation* induced by $\mathcal{D}$ and σ^C the *column permutation* induced by $\mathcal{D}$. We denote by $\mathcal{R}[i]$ and $\mathcal{C}[j]$ the i-th row of $\mathcal{R}$ and the j-th column of $\mathcal{C}$ for $i \in [m]$ and $j \in [n]$, respectively. Note that $\mathcal{R}[\sigma^R(i)] = i$ and $\mathcal{C}[\sigma^C(j)] = j$.

Observation 2.3.3
Note that for $\mathcal{R}_i = (r_{i1}, r_{i2}, \dots, r_{i|\mathcal{R}_i|})$ the numbers $\sigma^R(r_{i1}), \sigma^R(r_{i2}), \dots, \sigma^R(r_{i|\mathcal{R}_i|})$ are consecutive (i.e. $\sigma^R(r_{i(\ell+1)}) = \sigma^R(r_{i\ell}) + 1$ for $\ell \in [|\mathcal{R}_i| - 1]$). Analogously, the numbers $\sigma^C(c_{i1}), \sigma^C(c_{i2}), \dots, \sigma^C(c_{i|\mathcal{C}_i|})$ are consecutive for the tuple $\mathcal{C}_i = (c_{i1}, c_{i2}, \dots, c_{i|\mathcal{C}_i|})$.

This follows immediately from $\mathcal{R}[\sigma^R(i)] = i$ and $\mathcal{C}[\sigma^C(j)] = j$.

Since σ^R and σ^C are uniquely determined by a k-decomposition $\mathcal{D}$, we obtain a uniquely determined matrix by applying these permutations to the rows and columns of A, respectively. We denote this matrix by $\mathcal{D}(A)$ the *$\mathcal{D}$-decomposed matrix* or just *decomposed matrix* when no confusion can arise.

Note that the i-th row and the j-th column of $\mathcal{D}(A)$ are the $\mathcal{R}[i]$-th row and the $\mathcal{C}[j]$-th column of A, respectively. Moreover, the i-th row of A is the $\sigma^R(i)$-th row of $\mathcal{D}(A)$ and the j-th column of A is the $\sigma^C(j)$-th column of $\mathcal{D}(A)$. Thus, the following holds:

Observation 2.3.4
$\mathcal{D}(A) = A[\mathcal{R}[1], \dots, \mathcal{R}[m]; \mathcal{C}[1], \dots, \mathcal{C}[n]]$.

Now it is time to introduce some criteria for a k-decomposition $\mathcal{D}$ of A that will turn out to be sufficient to tell whether $\mathcal{D}(A)$ is in k-arrowhead form.

Definition 2.3.5 (Block condition)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix with entries a_{ij} for $i \in [m]$ and $j \in [n]$. Furthermore, let $k \in \mathbb{N}$ be a natural number and $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ be a k-decomposition of A. We say that $\mathcal{D}$ fulfills the *block condition* for A if for all $b, b' \in [k]$ with $b \neq b'$ holds: If $i \in \mathcal{R}_b$ for some $i \in [m]$ and $j \in \mathcal{C}_{b'}$ for some $j \in [n]$, then $a_{ij} = 0$.

Definition 2.3.6 (Load condition)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix. Let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}}, k \in \mathbb{N}$ integers such that $\ell^{\mathcal{R}} \leq u^{\mathcal{R}} \leq m$ and $\ell^{\mathcal{C}} \leq u^{\mathcal{C}} \leq n$. Let $\mathcal{D} = ((\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B), (\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B))$ be a k-decomposition of A. We say that $\mathcal{D}$ fulfills the *load condition* $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ if the following inequalities hold for all $t \in [k]$:

$$\ell^{\mathcal{R}} \leq |\mathcal{R}_t|, \tag{2.1}$$
$$u^{\mathcal{R}} \geq |\mathcal{R}_t|, \tag{2.2}$$
$$\ell^{\mathcal{C}} \leq |\mathcal{C}_t|, \text{ and} \tag{2.3}$$
$$u^{\mathcal{C}} \geq |\mathcal{C}_t|. \tag{2.4}$$

We call the equation 2.1 the *lower row load condition*, 2.2 the *upper row load condition*, 2.3 the *lower column load condition* and 2.4 the *upper column load condition*.

If $\mathcal{D}$ fulfills the block condition and the load condition $(1, m, 1, n)$, then $\mathcal{D}(A)$ is in k-arrowhead form:

Theorem 2.3.7 (Characterization of the k-arrowhead form)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. The following two statements hold:

1. *Let $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ be a k-decomposition of A. If $\mathcal{D}$ fulfills the block condition and the load condition $(1, m, 1, n)$, then $\mathcal{D}(A)$ is in k-arrowhead form.*
2. *If there are some permutation matrices $P_1 \in \mathbb{R}^{m\times m}$, $P_2 \in \mathbb{R}^{n\times n}$ and a matrix $A' \in \mathbb{R}^{m\times n}$ in k-arrowhead form with $P_1 A P_2 = A'$, then there is a k-decomposition $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ of A that fulfills the block condition, the load condition$(1, m, 1, n)$ and $\mathcal{D}(A)$ equals A', apart from the order of the rows and columns inside their blocks and their border.*

Proof: Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$ an integer. At first, we show statement *1.* Let $\mathcal{D} = ((\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B), (\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B))$ be a k-decomposition of A that fulfills the block condition and the load condition $(1, m, 1, n)$. We consider the matrix $\mathcal{D}(A)$ and declare which entries of $\mathcal{D}(A)$ are part of which submatrix B_i, C_i, R_i and D, for $i \in [k]$. After checking the dimensions of the submatrices, we verify that all other entries of $\mathcal{D}(A)$ are 0.

At first we define $r := |\mathcal{R}_B|$, $c := |\mathcal{C}_B|$, $m_i := |\mathcal{R}_i|$ and $n_i := |\mathcal{C}_i|$ for $i \in [k]$. We have $\mathcal{R}_B = (r_1^B, r_2^B, \dots$ and $r_r^B)$, $\mathcal{C}_B = (c_1^B, c_2^B, \dots, c_c^B)$. Moreover, for $i \in [k]$ we have $\mathcal{R}_i = (r_{i1}, r_{i2}, \dots, r_{im_i})$ and $\mathcal{C}_i = (c_{i1}, c_{i2}, \dots, c_{in_i})$. By Observation 2.3.4 we thus obtain

$$\begin{aligned} \mathcal{D}(A) = A[&r_{11}, \dots, r_{1m_1}, \dots, r_{k1}, \dots, r_{km_k}, r_1^B, \dots, r_r^B; \\ &c_{11}, \dots, c_{1n_1}, \dots, c_{k1}, \dots, c_{kn_k}, c_1^B, \dots, c_c^B]. \end{aligned} \tag{2.5}$$

Therefore, the rows and columns of each below defined matrix are consecutive rows and columns in $\mathcal{D}(A)$. Thus, they are submatrices of $\mathcal{D}(A)$.

$$\begin{aligned}
B_i &= A[r_{i1},\dots,r_{im_i};c_{i1},\dots,c_{in_i}] \in \mathbb{R}^{m_i\times n_i}, && i=1,\dots,k,\\
R_i &= A[r_1^B,\dots,r_r^B;c_{i1},\dots,c_{in_i}] \in \mathbb{R}^{r\times n_i}, && i=1,\dots,k,\\
C_i &= A[r_{i1},\dots,r_{im_i};c_1^B,\dots,c_c^B] \in \mathbb{R}^{m_i\times c}, && i=1,\dots,k,\\
D &= A[r_1^B,\dots,r_r^B;c_1^B,\dots,c_c^B] \in \mathbb{R}^{r\times c}.
\end{aligned}$$

If two of these submatrices have a common row, then their columns are pairwise different, hence the above defined submatrices have no common entries. With equation 2.5 we get the following:

$$\mathcal{D}(A) = \begin{array}{c} \\ r_{11},\dots,r_{1m_1} \\ \vdots \\ r_{k1},\dots,r_{km_k} \\ r_1^B,\dots,r_r^B \end{array} \begin{array}{c} \begin{array}{cccc} c_{11}\;\cdots\;c_{1n_1} & \cdots & c_{k1}\;\cdots\;c_{kn_k} & c_1^B\;\cdots\;c_c^B \end{array} \\ \left(\begin{array}{ccc|c} B_1 & & * & C_1 \\ & \ddots & & \vdots \\ * & & B_k & C_k \\ \hline R_1 & \cdots & R_k & D \end{array}\right) \end{array},$$

with $B_i \in \mathbb{R}^{m_i\times n_i}$, $R_i \in \mathbb{R}^{r\times n_i}$, $C_i \in \mathbb{R}^{m_i\times c}$, $D \in \mathbb{R}^{r\times c}$ with $r,c,m_i,n_i \in \mathbb{N}_0$ for all $i\in[k]$. Since the load condition $(1,m,1,n)$ is fulfilled we have $m_i\in[m]$ and $n_i\in[n]$ for all $i\in[k]$.

It remains to show that all other entries (marked with a star in the illustration above) of $\mathcal{D}(A)$ are zero. Let us consider such an entry $a^{\mathcal{D}}_{i^*j^*}$ in the i^*th-row and the j^*-th row of $\mathcal{D}(A)$ and let $i'=\sigma^R(i^*)\in[m]$ and $j'=\sigma^C(j^*)\in[n]$ be the corresponding indices of this entry in A. Since $a^{\mathcal{D}}_{i^*j^*}$ is not part of C_i, R_i and D for $i\in[k]$, we obtain $i'\notin\mathcal{R}_B$ and $j'\notin\mathcal{C}_B$. Hence, $i'\in\mathcal{R}_{b_1}$ and $j'\in\mathcal{C}_{b_2}$ for some $b_1,b_2\in[k]$. Because $a^{\mathcal{D}}_{i^*j^*}$ is not part of B_i, it holds that $b_1\neq b_2$. Thus, the block condition 2.3.5 implies that $0=a_{i'j'}=a^{\mathcal{D}}_{i^*j^*}$.

Now we prove point *2.* We construct a k-decomposition $\mathcal{D}$ by identifying the blocks of A' and verify that $\mathcal{D}(A)$ equals A' except for the order of rows and columns inside their blocks and border. Let P_1 and P_2 be permutation matrices and assume that $A'\in\mathbb{R}^{m\times n}$ is a matrix in k-arrowhead form such that $P_1AP_2=A'$. The entries of A' will be denoted with a'_{ij} for $i \in [m]$ and $j \in [n]$. It holds

$$A' = \begin{pmatrix} B_1 & & & & C_1 \\ & B_2 & & & C_2 \\ & & \ddots & & \vdots \\ & & & B_k & C_k \\ R_1 & R_2 & \cdots & R_k & D \end{pmatrix},$$

with $B_i \in \mathbb{R}^{m_i \times n_i}$, $R_i \in \mathbb{R}^{r \times n_i}$, $C_i \in \mathbb{R}^{m_i \times c}$, $D \in \mathbb{R}^{r \times c}$ with $r, c \in \mathbb{N}_0$ and $m_i, n_i \in \mathbb{N}$ for $i \in [k]$ by definition and all other entries equal zero. Consider the k-decomposition $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B, \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ such that for every row $i \in [m]$ of A and its corresponding index $i^* \in [m]$ in A' holds:

- If the i^*-th row of A' has an entry in the submatrix B_ℓ for some $\ell \in [k]$, then $i \in \mathcal{R}_\ell$, and
- if the i^*-th row of A' has no entry in the submatrix B_ℓ for all $\ell \in [k]$, then $i \in \mathcal{R}_B$.

Analogously, for all columns $j = 1, \ldots, n$ and its corresponding index $j^* \in [n]$ in A' we have:

- If the j^*-th column of A' has an entry in B_ℓ for some $\ell \in [k]$, then $j \in \mathcal{C}_\ell$, and
- if the j^*-th column of A' has no entry in B_ℓ for all $\ell \in \{1, \ldots, k\}$, then $j \in \mathcal{C}_B$.

Since A' is in k-arrowhead form, every row $i^* \in [m]$ of A' has entries in the submatrix B_ℓ for at most one $\ell \in [k]$ and hence the corresponding row $i \in [m]$ of A is in exactly one of the sets $\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B$; hence, $(\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A. Analogously, every column $j \in [n]$ is in exactly one of the sets $\mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B$; therefore, $(\mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ is a weak partition of the columns of A.

Furthermore, for every block $\ell \in \{1, \ldots, k\}$ holds $|\mathcal{R}_\ell| = m_\ell \geq 1$ and $|\mathcal{C}_\ell| = n_\ell \geq 1$. Therefore, the load condition$(1, m, 1, n)$ is fulfilled.

Let $b, b' \in [k], b \neq b'$ be two distinct blocks. Consider $i \in \mathcal{R}_b$ and $j \in \mathcal{C}_{b'}$; moreover, consider i^* the index in A' of the i-th row in A and j^* the index in A' of the j-th column of A. Because the i^*-th row of A' has an entry in B_b and the j^*-th column of A' has an entry in submatrix $B_{b'}$, with $b \neq b'$, we have $a'_{i^* j^*} = 0$. Hence, $a_{ij} = a'_{i^* j^*} = 0$ and therefore the block condition is fulfilled. With help of the construction used in the proof of point *1*, we obtain that $\mathcal{D}(A)$ equals A' apart from the order of the rows and columns inside the blocks and border.

$\square$

Remark 3:
The second point guarantees that every permutation yielding a matrix in k-arrowhead form, can be expressed (up to the order inside the blocks) as a k-decomposition that fulfills the block condition and the load condition$(1, m, 1, n)$.

These results expand naturally to the bordered k-block diagonal form since it is a special case of the k-arrowhead form:

Corollary 2.3.8 (Characterization of the bordered k-block diagonal form)
Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $k \in \mathbb{N}$ an integer. The following two statements hold:

1. *Let $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ be a k-decomposition of A. If $\mathcal{D}$ fulfills the block condition, the load condition $(1, m, 1, n)$ and $\mathcal{C}_B = \emptyset$ then $\mathcal{D}(A)$ is in bordered k-block diagonal form.*

2. *If there are some permutation matrices P_1, P_2 and a matrix $A' \in \mathbb{R}^{m \times n}$ in bordered k-block diagonal form with $P_1 A P_2 = A'$, then there is a k-decomposition $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ of A that fulfills the block condition, the load condition$(1, m, 1, n)$ and $\mathcal{D}(A)$ equals A' apart from permutations of the rows and columns inside their blocks and their border.*

Proof: We obtain statement *1* directly from the proof of Theorem 2.3.7, the number of border columns in the first proof is $c = |\mathcal{C}_B| = 0$, so $\mathcal{D}(A)$ is in particular in bordered k-block diagonal form. Statement *2* also follows directly from Theorem 2.3.7.

□

Remark 4:
Note that for $i \in [k]$ the rows and columns of B_i are the rows in $\mathcal{R}_i$ and the columns in $\mathcal{C}_i$, respectively. In particular, it is $B_i \in \mathbb{R}^{|\mathcal{R}_i| \times |\mathcal{C}_i|}$.

Consider a k-decomposition that fulfills the block condition, but some sets of the row partition $\mathcal{R}$ or the column partition $\mathcal{C}$ are empty. The corresponding blocks are "half-empty" and would not have any entry. It is possible to delete these block and assign the corresponig rows and columns to other blocks, without violating the block condition. In this way, it one would obtain a k'-decomposition with $k' < k$ that fulfills the block condition and all sets of its partitions are nonempty. Hence, Theorem 2.3.7 can be applied to obtain a matrix in k'-arrowhead form. A rigorous formulation of this fact is Lemma 8.2.1. It can be found in the Appendix. It shows that it can be convenient to allow empty blocks, e.g. if it is not known in how many blocks a matrix can be decomposed. The trivial solutions introduced in Remark 1 can also be excluded by choosing $u^{\mathcal{R}}$ and $u^{\mathcal{C}}$ small enough.

2.3.2 Problem formulation

At first, we introduce the problem of finding a Minimum Bordered Block Diagonal Form:

MINIMUM BORDERED BLOCK DIAGONAL FORM (MINBF)

Instance: Matrix $A \in \mathbb{R}^{m \times n}$, number of blocks $k \in \mathbb{N}$, $\ell^R, \ell^C \in \mathbb{N}_0$ lower block load bounds and $u^R, u^C \in \mathbb{N}$ upper block load bounds

Solution: A k-decomposition $\mathcal{D} = ((\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B), (\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B))$ that fulfills

1. the block condition,
2. the load condition $(\ell^R, u^R, \ell^C, u^C)$ and
3. $\mathcal{C}_B = \emptyset$.

Objective: Minimize $|\mathcal{R}_B|$

In the following we will call it MINBF.

We also want to look for decompositions to arrowhead form by solving the problem MINIMUM ARROWHEAD FORM:

MINIMUM ARROWHEAD FORM (MINAF)

Instance: Matrix $A \in \mathbb{R}^{m \times n}$, number of blocks $k \in \mathbb{N}$, $\ell^R, \ell^C \in \mathbb{N}_0$ lower block load bounds and $u^R, u^C \in \mathbb{N}$ upper block load bounds

Solution: A k-decomposition $\mathcal{D} = ((\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B), (\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B))$ that fulfills

1. the block condition and
2. the load condition $(\ell^R, u^R, \ell^C, u^C)$.

Objective: Minimize $|\mathcal{R}_B| + |\mathcal{C}_B|$

For a matrix $A \in \mathbb{R}^{m \times n}$, $k \in \mathbb{N}$, $\ell^R \in \mathbb{N}_0$, $u^R \in \mathbb{N}$, $\ell^C \in \mathbb{N}_0$, and $u^C \in \mathbb{N}$, we denote these problems by MINAF (A , k , ℓ^R , u^R , ℓ^C, u^C) or MINBF(A , k , ℓ^R , u^R , ℓ^C , u^C), respectively. If no confusion can arise, we will just write MINAF and MINBF. If the block load bounds are trivial (i.e. $\ell^R = \ell^C = 0$, $u^R = m$ and $u^C = n$) we will omit them and write just MINAF(A, k) and MINBF(A, k). If the column load bounds are trivial, we will just write MINAF(A, k, ℓ^R, u^R) and MINBF(A, k, ℓ^R, u^R).

Remark 5:
Notice that we could use the block load conditions, to express balance conditions on the size of the blocks (in terms of the number of rows or columns). If we want the blocks to be balanced in terms of rows for some real numbers $\alpha_1 \leq 1$ and $\alpha_2 \geq 1$:

$$\alpha_1 \frac{m}{k} \leq |\mathcal{R}_i| \leq \alpha_2 \frac{m}{k}, \qquad i \in [k],$$

we just set $\ell^R := \left\lceil \alpha_1 \frac{m}{k} \right\rceil$ and $u^R := \left\lfloor \alpha_2 \frac{m}{k} \right\rfloor$.

Similarly, if we want the number of columns in each block to be balanced for the real numbers $\beta_1 \leq 1$ and $\beta_2 \geq 1$:

$$\beta_1 \frac{n}{k} \leq |\mathcal{C}_i| \leq \beta_2 \frac{n}{k}, \qquad i \in [k],$$

we just set $\ell^C := \left\lceil \beta_1 \frac{n}{k} \right\rceil$ and $u^C := \left\lfloor \beta_2 \frac{n}{k} \right\rfloor$.

In the next section, we present some quality measures for decompositions.

2.4 Quality of a Decomposition

In the above defined problems the objective is to minimize the total number of border rows and columns. Thus, it is about minimizing the "size" of the border. Choosing this objective has two advantages:

- It is benefical for Applications 2.2.1 and 2.2.2.
- The objective function value is easy to calculate.

One could variate the "size" of the border by counting the number of all entries (even the zero entries), instead of counting the number of all rows and columns in the border. This number represents the "area" of the border. There are completely different criteria possible. We also could measure the balance of the block sizes. It is of interest to find a possibility to compare different decompositions concerning these criteria.

In order to do so, we will present a measure function μ for each of the above mentioned criteria. All of them have in common that they map an $m \times n$ matrix A and a corresponding k-decomposition $\mathcal{D}$ to a real number $\mu(A, \mathcal{D}) \in [0, 1]$. The higher $\mu(A, \mathcal{D})$ is, the better is $\mathcal{D}$ for A concerning the criteria of μ. Observe that every convex combination of these measures would be also a function that maps to the interval $[0, 1]$. Thus, it is possible to obtain mixed measures which compares decompositions concerning weighted criteria.

At first, we set up some helpful notations: Let $A \in \mathbb{R}^{m\times n}$ be a matrix, $k \in \mathbb{N}$ a natural number. Moreover, let $\mathcal{D} = ((\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B), (\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B))$ be a k-decomposition of A. Furthermore, let $\mathcal{D}^*_{m\times n}$ be the set of all k-decompositions of an $m \times n$ matrix. We define $m_i := |\mathcal{R}_i|$ and $n_i := |\mathcal{C}_i|$ for $i \in [k]$ and we declare $m_B := |\mathcal{R}_B|$, $n_B := |\mathcal{C}_B|$, $m^* := \max_{i\in\{1,...,k\}} m_i$ and $n^* := \max_{i\in\{1,...,k\}} n_i$.

Now we define three measure functions:

Definition 2.4.1 (border number measure)
The *border number measure* μ_{boN} is given by

$$\mu_{\text{boN}} \colon \mathbb{R}^{m\times n} \times \mathcal{D}^*_{m\times n} \to [0, 1],\ (A, \mathcal{D}) \mapsto \frac{(m + n) - (m_B + n_B)}{m + n}. \tag{2.6}$$

The border number measure is the ratio between the total number of nonborder rows and columns, and the total number of rows and columns. If there are neither border rows nor border columns, the value of μ_{boN} is one. On the other hand, if all rows were border rows and all columns were border columns, then μ_{boN} would be zero. There is a direct connection between the objective function value of the problems MinBf and MinAf for a k-decomposition and the value of μ_{boN}. Consider for a matrix $A \in \mathbb{R}^{m\times n}$ and $k \in \mathbb{N}$, two k-decompositions $\mathcal{D}_1$ and $\mathcal{D}_2$ that are feasible solutions for the problem MinAf or MinBf (for some fixed load condition) with objective function value val_1 and val_2, respetively. We then obtain

$$\mu_{\text{boN}}(A, \mathcal{D}_1) - \mu_{\text{boN}}(A, \mathcal{D}_2) = \frac{m + n - val_1}{m + n} - \frac{m + n - val_2}{m + n} = \frac{val_2 - val_1}{m + n}. \tag{2.7}$$

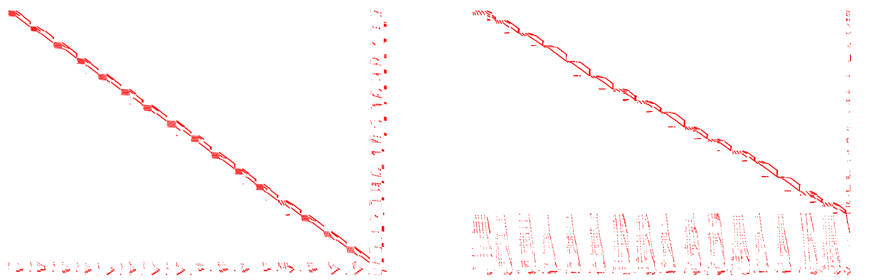
Hence, the bigger the value $\mu_{\mathrm{boN}}(A, \mathcal{D})$ is for a k-decomposition $\mathcal{D}$, the smaller the objective function value of $\mathcal{D}$ is. But notice, that if all rows were border rows, the value of μ_{boN} will not be zero unless there is at least one nonborder column. Therefore, for a k-decomposition that yields a matrix in bordered k-block diagonal form, this measure never attains zero.

This disadvantage can be revoked by using the *border area measure* μ_{boA} instead:

Definition 2.4.2
We define the *border area measure* μ_{boA} as the function:

$$\mu_{\mathrm{boA}} \colon \mathbb{R}^{m\times n} \times \mathcal{D}^*_{m\times n} \to [0,1],\ (A,P) \mapsto \frac{(m-m_B)(n-n_B)}{mn}. \tag{2.8}$$

This function measures the ratio between the number of entries that are not part of any border and the total number of entries. In other words it measures the ratio the “area” of the decomposed part of the matrix (the nonborder part) and the “area” of the complete matrix. It is easy to see that the value of μ_{boA} is zero if every row is a border row or all columns are border columns. The value of μ_{boA} is one if and only if there are no border rows and border columns. For the sake of visualization, we present the coefficient matrix of the mixed integer program 'a1c1s1.mps' from the MIPLIB 2003 [3] and MIPLIB 2010 [30]. In Figure 2.1a it is decomposed such that $\mu_{boA} = 0.91$ and Figure 2.1b shows a decomposition of it with $\mu_{boA} = 0.76$.



(a) 16-arrowhead form with $\mu_{boA} = 0.91$ (b) 16-arrowhead form with $\mu_{boA} = 0.76$

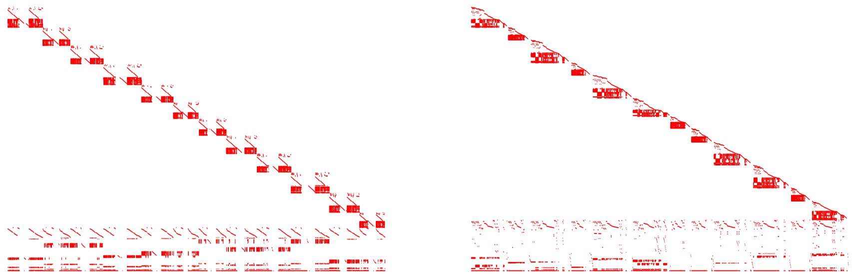
Figure 2.1: Coefficient matrix of a1c1s1.mps

The next measure should indicate how much the dimensions of the blocks differ:

Definition 2.4.3
The *block balance measure* μ_{blB} is given by

$$\mu_{\mathrm{blB}} \colon \mathbb{R}^{m\times n} \times \mathcal{D}^*_{m\times n} \to [0,1],\ (A,P) \mapsto \frac{1}{k^2}\sum_{i=1}^{k}\frac{m_i}{m^*}\sum_{j=1}^{k}\frac{n_j}{n^*}. \tag{2.9}$$

This measure indicates how much the number of rows and the number of columns of each block, differs from the highest number of rows and columns, respectively, that is attained by some block. We observe that if and only if all blocks are equal in the number of rows and columns, μ_{blB} becomes one. On the other hand, the value of μ_{blB} diminshes to zero, if the number of rows and columns become more and more variable. We display the coefficient matrix of 'arki001.mps' from the MIPLIB 2003 [3] decomposed in two different ways. Figure 2.2a shows a decomposition with $\mu_{blB} = 0.77$ and Figure 2.2b displays a decomposition with $\mu_{blB} = 0.56$. One can see that the sizes of the blocks of the latter one differ more than the block sizes of the first decomposition.



(a) decomposition with $\mu_{blB} = 0.77$ (b) decomposition with $\mu_{blB} = 0.56$

Figure 2.2: Coefficient matrix of arki001.mps in bordered 12-block diagonal form

Outlook

In the beginning of this section we have seen that it is possible to obtain mixed criteria measures by considering a convex combination of other measures. Ferris and Horn [17] suggest the following convex combination of the border area measure and the block balance measure for comparing decompositions: $\mu^* := 0.9\mu_{boA} + 0.1\mu_{blB}$. In order to compare our results with theirs, we will follow them and give some of our results with respect to μ^*.

In the next section we want to give an overview about previous literature dealing with detecting of block structures in matrices.

2.5 Literature review

In the following, we give a brief summary about the literature on matrix decomposing. There is rather narrow literature on decomposing unsymmetric, rectangular matrices, those we are interested in. We are aware of only one paper dealing with an exact method of decomposing a matrix to bordered block diagonal form. This is the work by Borndörfer, Ferreira and Martin [12]. However, we do not have notice of any exact algorithm for decomposing a matrix into arrowhead form. On the other hand, there are several papers for heuristic approaches. At first, we are going to give a short summary about the work of Borndörfer et al., followed by a brief overview about the literature on the heuristics.

2.5.1 Literature on Exact Decomposing Methods

The paper of Borndörfer et al. introduces the *matrix decomposition problem*(MDP) for a matrix $A \in \mathbb{R}^{m \times n}$, a number of blocks $\beta \in \mathbb{N}$, and a block capacity κ. The MDP is about assigning as many rows of A as possible to β blocks such that the following three conditions hold:

1. Each row is assigned to at most one block.
2. There are at most κ rows assigned to each block.
3. There do not exist two rows in different blocks that have a nonzero entry in the same column.

It can easily be seen that MDP(A, β, κ) is essentially the problem MinBf$(A, \beta, 0, \kappa, 0, n)$. In Section 5.1 we briefly present the integer program IP_B that solves the MDP. In practice this problem formulation has both advantages and disadvantages. On the one hand, it is possible to do some preprocessing operations on the matrix (e.g. deleting columns that are contained in other columns). But, on the other hand, one could obtain up to γ empty blocks if $(\beta - \gamma)\kappa \geq m$ for $\gamma \in \mathbb{N}$, because all rows could be assigned to the remaining $\beta - \gamma$ blocks. In addition to this lack of control of empty blocks, there is no possibility to balance the block sizes in terms of columns.

2.5.2 Literature on Heuristic Decomposing Methods

Now we give an overview of the literature on heuristics for decomposing matrices to arrowhead form and bordered k-block diagonal form.

The most recent work, is a paper [42] by Aykanat, Çatalyürek and Ucar from 2010. They presented three heuristic models for decomposing a matrix to arrowhead form. Although these models are customized to matrix-vector-multiplication, we will use one of them, namely the *fine-grain model* [42, 3.1]. However, we will denote it by the *hypercolrow model* since it fits better in our notation scheme. Moreover, the basic concepts of our *hypercol model* and *hyperrow model* are sketched in [42, 2.3] and are described in [41] and [40]. Furthermore, Ferris and Horn [17] suggest a bipartite graph model [17] to solve MinAf. This model is similar to the *bipartite model* we use in our *bipartite decomposing algorithm*. Moreover, they suggest the block balance measure 2.9, the border area measure 2.8 and the concept of dummy nodes 4.2.1. In Chapter 6 we will compare our results.

3 Complexity

Throughout this chapter, $A \in \mathbb{R}^{m \times n}$ will denote a matrix and $k \in \mathbb{N}$ an integer. First of all, we give some basic concepts of complexity theory. After this summary, we will present a polynomial algorithm that obtains for fixed $q \in \mathbb{N}_0$ a feasible solution for $\textsc{MinAf}(A, k, 1, m)$ or $\textsc{MinBf}(A, k, 1, m)$ with objective function value q if such a solution exists. We will the study the complexity of MinBf and MinAf. In order to do so, we will distinguish between two special cases. At first, we are going to look at $\textsc{MinAf}(A, m, 0, 1)$ and $\textsc{MinBf}(A, m, 0, 1)$ and reduce the $\mathcal{NP}$-hard problem Independent Set to both problems. The basic idea to reduce Independent Set to MinBf goes back to Borndörfer et al. [12]. Secondly, we present an additive inapproximability result for $k = 2$ and matrices with at most three nonzero entries in every row and at most two nonzero entries in every column.

3.1 Basic Definitions From Complexity Theory

We assume that the reader is aware of the fundamentals of complexity theory. For the sake of completeness, we give a short summary. For a thorough treatment, we refer to the book "Computers and intractability" by Garey and Johnson [19].

An algorithm that terminates after a number of steps bounded by a polynomial in the size of the input is called a *polynomial time algorithm*. Here, a step consists of performing one basic instruction. A problem is said to be *solvable in polynomial time* or *tractable* if it can be solved by a polynomial time algorithm.

A *decision problem* is a problem whose instances are each either a 'yes'- or a 'no'-instance. The class $\mathcal{P}$ consists of all decision problems that are tractable. $\mathcal{P}$ is a subclass of $\mathcal{NP}$, where the latter contains all decision problems that can be verified in polynomial time. This means that given a polynomial certificate of a solution, one can check if the certificate is correct in time polynomial in the size of the input. Problems that are in $\mathcal{NP}$ and are at least as hard as any problem in $\mathcal{NP}$ are called to be $\mathcal{NP}$-hard. By hardness we mean the concept of polynomial reducibility. A *reduction* is a procedure which transforms an arbitrary instance α of problem A into an instance β of problem B such that α is a 'yes'-instance if and only if β is a 'yes' instance. If this transformation takes polynomial time in the input size of α, we say that problem *A polynomial reduces to* problem B.

Many problems of combinatorial optimization have optimization tasks. We want to find a feasible solution that minimizes or maximizes a certain objective. We can expand the concept we just mentioned to optimization problems naturally. For every optimization problem we can define a corresponding decision problem by giving an upper bound on the objective value of a minimization task or a lower bound on the objective value of a maximization problem, respectively. An instance of such a decision problem is a 'yes'-instance if and only if there is a feasible solution whose objective value respects the bound condition.

3.2 A Polynomial Algorithm for Fixed Objective Value

In this section, we present a polynomial time algorithm that obtains for a fixed $q \in \mathbb{N}_0$ a feasible solution for $\textsc{MinAf}(A, k, 1, m)$ with objective function value q if such a solution exists. Afterwards, we show how to adapt this algorithm to $\textsc{MinBf}(A, k, 1, m)$. For our approach, it will be useful to find the *connected components* of a hypergraph. For the sake of completeness, we start with some basic ideas about connected components and the well-known problem of finding them:

Definition 3.2.1 (Connected component)
Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph. A nonempty subset of the nodes $\mathcal{S} \subseteq \mathcal{V}$ is called *component* of $\mathcal{H}$ if for all nodes v, w with $v \in \mathcal{S}$ and $w \in \mathcal{V} \setminus \mathcal{S}$ hold that v and w are not adjacent (i.e. there is no hyperedge e with $v \in e$ and $w \in e$). $\mathcal{S}$ is called *connected component* if $\mathcal{S}$ is a component with the following property: For all nodes $v_1, v_2 \in \mathcal{S}$ there is a *path*[1] in $\mathcal{S}$ that connects v_1 and v_2.

Remark 6:
The set of connected components of a hypergraph is unique and two distinct connected components of a hypergraph are disjoint. Moreover, the set of connected components is a partition of the set of vertices of a hypergraph.

The following remark follows directly from Definition 3.2.1.

Remark 7:
Let $\mathcal{S}_1, \ldots, \mathcal{S}_K$ be the connected components of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. Then for every hyperegde $e \in \mathcal{E}$ holds: If there is a node $v \in \mathcal{S}_b$ with $v \in e$ for some $b \in [k]$, then for all nodes $v' \in \mathcal{V}$ with $v' \in e$, it is also true that $v' \in \mathcal{S}_b$.

> CONNECTED COMPONENTS
> ***Instance:*** $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ a hypergraph.
> ***Solutions:*** The set $\mathcal{K} = \{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$ of all connected components of $\mathcal{H}$ with K is the number of the connected components of $\mathcal{H}$.

Remark 8:
Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = m$ and a node $v \in \mathcal{V}$, one can find the connected component that includes v by applying a breadth-first search algorithm starting at v. Hence, the problem of finding all connected components of a hypergraph can be solved in polynomial time by calling a breadth-first search algorithm not more than m times.

[1] By a *path* we mean a finite sequence of pairwise distinct nodes $v_1, v_2, \ldots, v_\ell$ with v_i and v_{i+1} are adjacent for $i \in [\ell - 1]$.

One can exploit this fact by detecting the connected components in the so-called hypercolumn graph of a matrix which encodes the structure of its nonzero entries:

Definition 3.2.2 (Hypercolumn Graph)
Let $A \in \mathbb{R}^{m \times n}$ be a matrix. The *hypercolumn graph* of A is defined as the hypergraph $\mathcal{H}_A^C = (\mathcal{V}, \mathcal{E})$ with

- $\mathcal{V} = \{v_i : i \in [m]\}$ and
- $\mathcal{E} = \{e_j : j \in [n]\}$, with $v_i \in e_j$ if and only if $a_{ij} \neq 0$.

We will denote by v_i the node that belongs to the i-th row of A and e_j as the hyperedge that belongs to the j-th column of A.

Creating the hypercolumn graph of a matrix A is the first step in Algorithm 1 that is presented below. More precisely, the function `CreateHyperColumnGraph()` creates and returns the hypercolumn graph $\mathcal{H}_A^C$ of A. Next, we iterate over all pairs of subsets of nodes and hyperedges of $\mathcal{H}_A^C$ that have together a total number of q elements. For each of these pairs we delete the corresponding nodes and hyperedges from $\mathcal{H}_A^C$ and find the connected components of the remaining graph by calling the method `FindConnectedComponents()`. This method can be implemented such that it has a running time polynomial in the size of the input as was noted in Remark 8. If the number of the connected components K is at least k, we can obtain a k-decomposition from them. This is accomplished in the method `BuildDecomposition()` that is displayed in Algorithm 2. The methods `CreateHyperColumnGraph()` and `FindConnectedComponents()` are clear from Definition 3.2.2 and Remark 8.

Algorithm 1: FixedCostsArrowhead

input : A matrix $A \in \mathbb{R}^{m \times n}$, an integer $k \in \mathbb{N}$ with $k \geq 2$ and $q \in \mathbb{N}_0$.
output: A k-decomposition of A that is feasible for $\textsc{MinAf}(A, k, 1, m)$ with objective function value equals q or a statement that there is no such k-decomposition.

$\mathcal{H}_A^C = (\mathcal{V}, \mathcal{E}) \leftarrow$ `CreateHyperColumnGraph(`A`)`;
foreach *pair of subsets* $(\bar{V}, \bar{E})$ *with* $\bar{V} \subseteq \mathcal{V}$, $\bar{E} \subseteq \mathcal{E}$ *and* $|\bar{V}| + |\bar{E}| = q$ **do**
  $\mathcal{H}' \leftarrow (\mathcal{V} \setminus \bar{V}, \mathcal{E} \setminus \bar{E})$;
  $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_K) \leftarrow$ `FindConnectedComponents(`$\mathcal{H}'$`)`;
  if $K \geq k$ **then**
    return $\mathcal{D} \leftarrow$ `BuildDecomposition(`$\mathcal{H}_A^C$,$\mathcal{S}$,k,$\bar{V}$, $\bar{E}$, m, n`)`;
  end
end
output that $\textsc{MinAf}(A, k, 1, m)$ *has no feasible solution with objective function value q;*

In the following, we will see that Algorithm 1 returns a feasible k-decomposition for $\textsc{MinAf}(A, k, 1, m)$ with objective value q if and only if there is such a feasible solution.

Algorithm 2: BuildDecomposition

input : A hypercolumn graph $\mathcal{H}_A^C = (\mathcal{V}, \mathcal{E})$ of a matrix $A \in \mathbb{R}^{m \times n}$, a set of connected components $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_K)$ of a subgraph of $\mathcal{H}_A^C$, $k \in \mathbb{N}$ with $k \geq 2$, $\bar{V} \subseteq \mathcal{V}$ a subset of vertices of $\mathcal{H}_A^C$, $\bar{E} \subseteq \mathcal{E}$ a subset of hyperedges of $\mathcal{H}_A^C$, $m \in \mathbb{N}, n \in \mathbb{N}$.

output: A k-decomposition of A that is feasible for $\textsc{MinAf}(A, k, 1, m)$.

$\mathcal{R}_B \leftarrow \{i \in [m] | v_i \in \bar{V}\}$;

$\mathcal{C}_B \leftarrow \{j \in [n] | e_j \in \bar{E}\}$;

for $b \in [k] \setminus \{1\}$ **do**

  $\mathcal{R}_b \leftarrow \{i \in [m] | v_i \in \mathcal{S}_b\}$;

  $\mathcal{C}_b \leftarrow \{j \in [n] | \exists v \in e_j : v \in \mathcal{S}_b\}$;

end

$\mathcal{R}_1 \leftarrow [m] \setminus \left(\bigcup_{b=2}^k \mathcal{R}_b \cup \mathcal{R}_B\right)$;

$\mathcal{C}_1 \leftarrow [n] \setminus \left(\bigcup_{b=2}^k \mathcal{C}_b \cup \mathcal{C}_B\right)$;

return $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$;

Lemma 3.2.3 (Correctness of Algorithm 1)
Let $A \in \mathbb{R}^{m \times n}$, $k \in \mathbb{N}$ and $q \in \mathbb{N}_0$. The following two statements are equivalent:

1. *There is a k-decomposition that is feasible for* $\textsc{MinAf}(A, k, 1, m)$ *with objective function value q.*

2. *Algorithm 1 returns a k-decomposition that is feasible for* $\textsc{MinAf}(A, k, 1, m)$ *with objecive function value q.*

Proof: Let $A \in \mathbb{R}^{m \times n}$, $k \in \mathbb{N}$ and $q \in \mathbb{N}_0$. Furthermore, let $\mathcal{H}_A^C = (\mathcal{V}, \mathcal{E})$ be the hypercolumn graph of A. At first, we proof the implication "$1 \Rightarrow 2$".

Let $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ be a k-decomposition that is a feasible solution for $\textsc{MinAf}(A, k, m, 1)$ with objective value q. We set $\bar{V} := \{v_i \in \mathcal{V} | i \in \mathcal{R}_B\}$ and $\bar{E} := \{e_j \in \mathcal{E} | j \in \mathcal{C}_B\}$ and notice that the pair $(\bar{V}, \bar{E})$ is chosen by the algorithm in line 2 since $|\bar{V}| + |\bar{E}| = q$ holds. We define $V' := \mathcal{V} \setminus \bar{V}$ and $E' := \mathcal{E} \setminus \bar{E}$.

Consider $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_K)$ the connected components of $\mathcal{H}' := (V', E')$ that is found in line 4. Our next goal is to show that $K \geq k$. In order to prove this, we set $\mathcal{V}^b := \{v_i \in \mathcal{V} : i \in \mathcal{R}_b\}$ for $b \in [k]$. The task is now to prove that $\{\mathcal{V}^1, \ldots, \mathcal{V}^k\}$ is a weak partition of the nodes of $\mathcal{H}'$ and that $\mathcal{V}^b$ is a component of $\mathcal{H}'$ for all $b \in [k]$. The set $\{\mathcal{V}^1, \ldots, \mathcal{V}^k\}$ is a weak partition of the nodes of $\mathcal{H}'$ because of the definition of $\mathcal{V}^b$ for $b \in [k]$ and the fact that $(\mathcal{R}_1, \ldots, \mathcal{R}_k)$ is a weak partition of $[m] \setminus \mathcal{R}_B$.

Consider now $\mathcal{V}^{b_1}$ for an arbitrary $b_1 \in [k]$. Our next claim is that $\mathcal{V}^{b_1}$ is a component of $\mathcal{H}'$. $\mathcal{V}^{b_1}$ is not empty because, on the one hand, $\mathcal{R}_{b_1}$ is not empty (since $\mathcal{D}$ fulfills the load condition $(1, m, 0, n)$). On the other hand, for all $i \in \mathcal{R}_{b_1}$ we conclude that $i \notin \mathcal{R}_B$, hence that $v_i \notin \bar{V}$, and finally that $v_i \in V'$. Let $v_{i_1}, v_{i_2} \in V'$ be nodes such that $v_{i_1} \in \mathcal{V}^{b_1}$ and $v_{i_2} \notin \mathcal{V}^{b_1}$. Hence, we obtain $i_1 \in \mathcal{R}_{b_1}$. Since $v_{i_2} \in V'$ and $v_{i_2} \notin \mathcal{V}^{b_1}$, we have $i_2 \notin \mathcal{R}_B$ and $i_2 \notin \mathcal{R}_{b_1}$, and hence $i2 \in \mathcal{R}_{b_2}$ with $b_2 \in [k] \setminus \{b_1\}$. To obtain a contradiction, suppose that v_{i_1} and v_{i_2} are adjacent, i.e. there is an edge $e_j \in E'$ with $v_{i_1} \in e_j$ and $v_{i_2} \in e_j$. Since $e_j \in E'$ we obtain $j \notin \mathcal{C}_B$ and therefore $j \in \mathcal{C}_b$ for $b \in [k]$. Due to the fact taht $v_{i_1} \in e$ and $v_{i_2} \in e$, we obtain that $a_{i_1 j} \neq 0$ and $a_{i_2 j} \neq 0$. Since $\mathcal{D}$ fulfills the block condition, we have $b = b_1$ and $b = b_2$, which contradicts $b_1 \neq b_2$. Hence, $\mathcal{V}^{b_1}$ is a component of $\mathcal{H}'$.

Therefore, the sets $\mathcal{V}^1, \ldots, \mathcal{V}^k$ are pairwise disjoint components of $\mathcal{H}'$. Hence, the number of connected components are at least k. Therefore, we have $K \geq k$ and hence the method `BuildDecomposition()` in line 6 is called. In this method the following sets are created:

- $\mathcal{R}^*_B = \{i \in [m] | v_i \in \bar{V}\}$,
- $\mathcal{C}^*_B = \{j \in [n] | e_j \in \bar{E}\}$,
- $\mathcal{R}^*_b = \{i \in [m] | v_i \in \mathcal{S}_b\}$ for $b \in [k] \setminus \{1\}$,
- $\mathcal{C}^*_b = \{j \in [n] | \exists v \in e_j : v \in \mathcal{S}_b\}$ for $b \in [k] \setminus \{1\}$,
- $\mathcal{R}^*_1 = [m] \setminus \left(\bigcup_{b=2}^k \mathcal{R}_b \cup \mathcal{R}_B \right)$ and
- $\mathcal{C}^*_1 = [n] \setminus \left(\bigcup_{b=2}^k \mathcal{C}_b \cup \mathcal{C}_B \right)$.

We notice that $\mathcal{R}^*_1 = \{i \in [m] | v_i \in \mathcal{S}_1\}$ since $(\mathcal{S}_1, \ldots, \mathcal{S}_K)$ is a partition of V'. Moreover, the set $\{\mathcal{R}^*_1, \ldots, \mathcal{R}^*_k, \mathcal{R}^*_B\}$ is a weak partition of $[m]$, and the set $\{(\mathcal{C}^*_1, \ldots, \mathcal{C}^*_k, \mathcal{C}^*_B\}$ is a weak partition of $[n]$ because of Remark 7. Hence, $\mathcal{D}^* := (\mathcal{R}^*_1, \ldots, \mathcal{R}^*_k, \mathcal{R}^*_B; \mathcal{C}^*_1, \ldots, \mathcal{C}^*_k, \mathcal{C}^*_B)$ is a k-decomposition of A. Since $|\mathcal{S}_b| \geq 1$ for all $b \in [k]$, $\mathcal{D}^*$ fulfills the load condition $(1, m, 0, n)$.

It remains to prove that $\mathcal{D}^*$ fulfills the block condition. Consider $i \in \mathcal{R}^*_{b_1}$ and $j \in \mathcal{C}^*_{b_2}$ with $a_{ij} \neq 0$ for some $b_1, b_2 \in [k]$. Since $a_{ij} \neq 0$, we have $v_i \in e_j$. We consider 2 cases:

1. If $b_1 \neq 1$, then we have $v_i \in S_{b_1}$. By definition of $\mathcal{C}^*_{b_1}$, it follows that $j \in \mathcal{C}^*_{b_1}$. Hence, we obtain $b_1 = b_2$ and $\mathcal{D}^*$ fulfills the block condition.
2. In the case $b_1 = 1$, we assume for contradiction that $j \notin \mathcal{C}^*_1$. Thus, we have $j \in \mathcal{C}^*_{b_2}$ for some $b_2 \in [k] \setminus \{1\}$. By definition of $\mathcal{C}^*_{b_2}$, there is a node $v \in e_j$ with $v \in \mathcal{S}_{b_2}$. From Remark 7, we conclude that $v_i \in \mathcal{S}_{b_2}$, and therefore $i \in \mathcal{R}^*_{b_2}$ with $b_2 \neq 1$, which is a contradiction. Hence, $j \in \mathcal{C}^*_1$ and thus $\mathcal{D}^*$ fulfills the block condition.

From the above it follows that $\mathcal{D}^*$ which is returned by Algorithm 1 in line 6, is feasible for $\textsc{MinAf}(A, k, 1, m)$ with objective function value

$$|\mathcal{R}^*_B| + |\mathcal{C}^*_B| = |\bar{V}| + |\bar{E}| = q.$$

The implication "$2 \Rightarrow 1$" is obviously true since Algorithm 1 already returns a feasible solution for $\textsc{MinAf}(A, k, 1, m)$ with objective function value q.

□

In the following we are going to prove that Algorithm 1 runs in polynomial time in the input size.

Theorem 3.2.4
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$. For a fixed value $q \in \mathbb{N}_0$, there is a polynomial algorithm that obtains a feasible solution for $\textsc{MinAf}(A, k, 1, m)$ *with objective function value equals q if there is such a solution.*

Proof: We show that for fixed $q \in \mathbb{N}_0$ Algorithm 1 runs in polynomial time in the input size. The method `CreateHyperColumnGraph()` can be implemented such that it runs in $\mathcal{O}(mn)$ time. It is known that the method inside the loop `FindConnectedComponents()` can be implemented in polynomial time in the input size as sketched in Remark 8. In the method `BuildDecomposition()`, the sets $\mathcal{R}_1^*, \ldots, \mathcal{R}_k^*, \mathcal{R}_B^*, \mathcal{C}_1^*, \ldots, \mathcal{C}_k^*$, and $\mathcal{C}_B^*$ are created. This can be implemented such that the method runs in polynomial time in the input. Finally, we have to count the maximal number of loop iterations in line 2. In the worst case, the loop is iterated for every pair $(\bar{V}, \bar{E})$ with $\bar{V} \subseteq \mathcal{V}$, $\bar{E} \subseteq \mathcal{E}$ and $|\bar{V}| + |\bar{E}| = q$. Thus, we have to count the number of all subsets of size q of a basic set that has cardinality $m + n$, which is $\binom{m+n}{q}$. We obtain

$$\begin{aligned}\binom{m+n}{q} &= \frac{(m+n)!}{q!(m+n-q)!} = \frac{m+n}{1}\frac{m+n-1}{2}\cdots\frac{m+n-(q-1)}{q}\\ &= \prod_{i=1}^{q}\frac{m+n+1-i}{i} \leq (m+n)^q.\end{aligned}$$

Hence, for fixed q the number of loop iterations is polynomially in the size of the input. Therefore, Algorithm 1 runs in polynomial time in the input size and due to Lemma 3.2.3 obtains a feasible solution for $\textsc{MinAf}(A, k, 1, m)$ with objective function value equal to q if there is one.

□

For the sake of completeness, we give an variation of Algorithm 1 that finds a solution of the problem $\textsc{MinBf}(A, k, m, 1)$ in polynomial time of the input size for a fixed objective function value q. This variation is presented in Algorithm 3.

Lemma 3.2.5 (Correctness of Algorithm 3)
Let $A \in \mathbb{R}^{m\times n}$, $k \in \mathbb{N}$ and $q \in \mathbb{N}_0$. The following two statements are equivalent:

1. *There is a k-decomposition that is feasible for* $\textsc{MinBf}(A, k, 1, m)$ *with objective function value q.*
2. *Algorithm 3 returns a k-decomposition that is feasible for* $\textsc{MinBf}(A, k, 1, m)$ *with objecive function value q.*

Proof: Essentially, we can use the same proof as for Lemma 3.2.3, except that we treat the $\textsc{MinAf}$ problem as a $\textsc{MinBf}$ problem. Therefore, the sets $\mathcal{C}_B$ and $\bar{E}$ are empty. Hence, $\mathcal{C}_B^*$ is empty and $\mathcal{D}^*$ is a feasible solution for $\textsc{MinBf}(A, k, m, 1)$ with objective function value q.

□

Algorithm 3: FixedCostsBorderedBlock

input : A matrix $A \in \mathbb{R}^{m \times n}$, an integer $k \in \mathbb{N}$ with $k \geq 2$ and $q \in \mathbb{N}_0$.
output: A k-decomposition of A that is feasible for MinBf$(A, k, 1, m)$ with objective function value equals q or a statement that there is no such k-decomposition.

$\mathcal{H}_A^C = (\mathcal{V}, \mathcal{E}) \leftarrow$ `CreateHyperColumnGraph(`A`)`;
foreach *subset* $\bar{V} \subseteq \mathcal{V}$ *with* $|\bar{V}| = q$ **do**
    $\mathcal{H}' \leftarrow (\mathcal{V} \setminus \bar{V}, \mathcal{E})$;
    $\mathcal{K} = (\mathcal{S}_1, \dots, \mathcal{S}_K) \leftarrow$ `FindConnectedComponents(`$\mathcal{H}'$`)`;
    if $K \geq k$ **then**
        return $\mathcal{D} \leftarrow$ `BuildDecomposition(`$\mathcal{H}_A^C$,$\mathcal{K}$,k,$\bar{V}$, $\bar{E}$, m, n`)`;
    end
end
output that MinBf$(A, k, 1, m)$ *has no feasible solution with objective function value q;*

Corollary 3.2.6
Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $k \in \mathbb{N}$. For a fixed value $q \in \mathbb{N}$, there is a polynomial algorithm that decides whether or not there is a feasible solution for MinBf$(A, k, 1, m)$ *with objective function value equals q.*

Proof: Since Algorithm 3 is essentially Algorithm 1 with a reduced number of loop iterations, the statement follows from the proof of Theorem 1. The number of loop iterations is the number of all subsets of nodes with cardinality equal to q. Therefore, the number of loop iterations is:

$$\begin{aligned} \binom{m}{q} &= \frac{(m)!}{q!(m-q)!} = \frac{m}{1}\frac{m-1}{2} \cdots \frac{m-(q-1)}{q} \\ &= \prod_{i=1}^{q} \frac{m+1-i}{i} \leq m^q. \end{aligned}$$

Hence, for fixed q the number of loop iterations is polynomial in the size of the input.

$\square$

In practice, however, this algorithm is only convenient for very small q.

3.3 Complexity for MinAf$(A, m, 0, 1)$ and MinBf$(A, m, 0, 1)$

At first, we will show that for incidence matrices of undirected graphs and minimum row block load $\ell^R = 0$, we can transform every solution of MinAf to a solution of MinBf, in polynomial time in the input size, without increasing the objective value. Here and subsequently, the objective function value of an instance S of MinBf and MinAf is denoted by $val(S)$ Then, we introduce the $\mathcal{NP}$-hard Independent Set problem and show that it reduces to MinBf$(A, m, 0, 1)$ and MinAf$(A, m, 0, 1)$.

Lemma 3.3.1
Let $A \in \mathbb{R}^{m\times n}$ be the incidence matrix of an undirected graph $G = (N, E)$. Let S be a feasible solution of MinAf$(A, k, 0, u)$ *for $k, u \in \mathbb{N}$. Then there is a feasible solution S' for the problem* MinBf$(A, k, 0, u)$ *with* $\mathrm{val}(S') \leq \mathrm{val}(S)$ *that can be obtained in polynomial time in size of the input.*

Proof: Consider a graph $G = (N, E)$ with $N = \{v_1, \dots, v_m\}$ and $E = \{e_1, \dots e_m\}$. Moreover, consider $k, u \in \mathbb{N}$. Let $A \in \mathbb{R}^{m\times n}$ be the incidence matrix of G with entries $a_{ij} = 1$ if $v_i \in e_j$, and otherwise is $a_{ij} = 0$. We consider a solution of MinAf$(A, k, 0, u)$ that is given by a k-decomposition $\mathcal{D} = ((\mathcal{R}^1, \dots, \mathcal{R}^k, \mathcal{R}^B), (\mathcal{C}^1, \dots, \mathcal{C}^k, \mathcal{C}^B))$, that fulfills the block condition and the load condition $(0, u, 0, n)$. In the following, we want to apply some modifications to $\mathcal{D}$ and obtain $\mathcal{D}' = ((\mathcal{R}'_1, ..., \mathcal{R}'_k, \mathcal{R}'_B), (\mathcal{C}'_1, \dots, \mathcal{C}'_k, \mathcal{C}'_B))$ which basically equals $\mathcal{D}$, except for the modifications described below. It will fulfill the block condition and the load condition $(0, u, 0, n)$.

At first, we notice that every column of A has only two entries; hence, in particular, every $j \in \mathcal{C}_B$ has exactly two nonzero entries. We denote them by i_1^j and i_2^j, and distinguish between three cases:

1. If $i_1^j, i_2^j \in \mathcal{R}_B$, then we can reassign j to an arbitrary column block $\mathcal{C}'_t$ for all $t \in [k]$, without violating the block condition. This modification decreases the objective function value by one.

2. If either $i_1^j \in \mathcal{R}_B$ or $i_2^j \in \mathcal{R}_B$, then we assume w.l.o.g. that $i_1^j \in \mathcal{R}_B$ and $i_2^j \in \mathcal{R}_{t'}$ for some $t' \in [k]$. Then we reassign j to $\mathcal{C}'_{t'}$ and the block condition is still fulfilled. Again, the objective function value decreases by one.

3. If i_1^j, $i_2^j \notin \mathcal{R}_B$, then we have $i_1^j \in \mathcal{R}_{b_1}$ and $i_2^j \in \mathcal{R}_{b_2}$ for some $b_1, b_2 \in [k]$. We consider two cases: If, on the one hand, $b_1 = b_2$, then we reassign j to $\mathcal{C}'_{b_1}$. This decreases the objective value by one and the block condition is still fulfilled. If, on the other hand, $b_1 \neq b_2$, then we also reassign j to $\mathcal{C}'_{b_1}$ but the block condition would be violated since $a_{i_2^j j} = 1 \neq 0$ with $i_2^j \in \mathcal{R}'_{b_2}$ and $j \in \mathcal{C}'_{b_1}$. Therefore, we reassign i_2^j to $\mathcal{R}'_B$ and the block condition is fulfilled again. This modification increases the objective function value by one, and thus altogether the objective value is maintained.

After applying this to all columns $j \in \mathcal{C}_B$, we get $\mathcal{D}'$, that fulfills the block condition and the load condition $(0, u, 0, n)$ because no row is added to a row block part. Furthermore, we have $\mathcal{C}'_B = \emptyset$. Hence, $\mathcal{D}'$ is a feasible solution for MinBf$(A, k, 0, u)$. Notice, that each of the three modifications can be implemented to run in polynomial time in the size of the input. Furthermore, the maximal number of the above described modifications is n. Thus, $\mathcal{D}'$ can be obtained in polynomial time in the input size.

$\square$

In the following, we introduce the decision version of the Independent Set problem and show that it reduces to MinBf and MinBf.

Definition 3.3.2 (Independent set)
Let $G = (N, E)$ be an undirected graph. A subset of nodes $S \subseteq N$ is called *independent set* if every edge $e \in E$ has at most one endpoint in S.

We define the *size* of S to be $|S|$.

Independent set
Instance: $G = (N, E)$ an undirected graph and an integer $K \in \mathbb{N}$.
Solutions: $S \subseteq N$, an independent set with size at least K.

Remark 9:
Independent Set is $\mathcal{NP}$-hard. This can be seen by reduction from 3-Satisfiability. We omit the proof and refer the reader to the classical work of Garey and Johnson [19].

Theorem 3.3.3
The problems MinAf *and* MinBf *are $\mathcal{NP}$-hard even when restricted to matrices with at most 2 nonzero entries per column and with row load bounds $\ell^R = 0$ and $u^R = 1$.*

Proof: Consider a graph $G = (N, E)$ with m nodes and n edges, and let $q \in \mathbb{N}$ be an integer. Let $A \in \mathbb{R}^{m \times n}$ be the incidence matrix of G. Hence, A has exactly two nonzero entries per column. We will reduce Independent Set to the decision problems for MinBf$(A, m, 0, 1)$ and MinAf$(A, m, 0, 1)$. In order to do so, it is sufficient to show that (G, q) is a 'yes'-instance of Independent Set if and only if there is a feasible solution S for MinBf$(A, m, 0, 1)$ with $\mathit{val}(S) \leq m - q$. The reason for that is the following: On the one hand, every feasible solution of MinBf$(A, m, 0, 1)$ is also a feasible solution of MinAf$(A, m, 0, 1)$ with the same objective value, and on the other hand, by Lemma 3.3.1 every feasible solution of MinAf$(A, m, 0, 1)$ can be transformed to a solution of MinBf$(A, m, 0, 1)$ in polynomial time in the size of the input. Therefore, MinBf$(A, m, 0, 1)$ reduces to prMinAf$(A, m, 0, 1)$ if A is an incidence matrix of an undirected graph, and thus Independent Set would also reduces to MinAf.

Consider such a solution S for MinBf$(A, m, 0, 1)$ with $\mathit{val}(S) \leq m - q$. To be more precise, this is a k -decomposition $\mathcal{D} = ((\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B), (\mathcal{C}_1, \dots, \mathcal{C}_k, \emptyset)$ of A with $|\mathcal{R}_B| \leq m - q$ that fulfills the block condition and the load condition $(0, 1, 0, n)$. Since $|\mathcal{R}_B| = \mathit{val}(S) \leq m - q$, there are at least q rows that are not in the row border part $\mathcal{R}_B$. Each of these nonborder rows is in one unique row block because the upper row load limit equals one. Hence, for all pairs of distinct nonborder rows (i_1, i_2) with $i_1, i_2 \in [m] \setminus \mathcal{R}_B, i_1 \neq i_2$, there is no column such that both rows have a nonzero entry in this column; otherwise, the block condition would be violated. Therefore, there is no pair of corresponding vertices (v_{i_1}, v_{i_2}) that is incident to the same edge. Thus, the vertices that belong to the nonborder rows are an independent set with size greater or equal to q. Hence, (G, q) is a 'yes'-instance.

On the other hand, let (G, q) be a 'yes'-instance of Independent Set. Then there is an independent set I with $|I| \geq q$. We will build a $|I|$-decomposition $\mathcal{D}$ for A that is a feasible solution for MinBf$(A, m, 0, 1)$ with value at most $m - q$. At first, we put each row that corresponds to a node from I to one unique row block. The remaining rows are assigned to $\mathcal{R}_B$. We note that $|\mathcal{R}_B| \leq m - q$ and that $(\mathcal{R}_1, \ldots, \mathcal{R}_{|I|}, \mathcal{R}_B)$ is a weak partition of the rows of A. Since I is an independent set, every edge of G is incident to at most one node in I. Hence, for every column j, there is at most one row i with $a_{ij} \neq 0$ and $i \notin \mathcal{R}_B$, but $i \in \mathcal{R}_{b_i}$ for some $b_i \in [|I|]$. For $j \in [n]$, we distinguish between two cases: Either there is such a row i for j (with $a_{ij} \neq 0$ and $i \in \mathcal{R}_{b_i}$ for some $b_i \in [|I|]$) or there is not such a row. If there is such a row we assign j to $\mathcal{C}_{b_i}$. If there is not such a row, then both rows which has an nonzero entry with j are in $\mathcal{R}_B$. Therefore, we can assign j to $\mathcal{C}_b$ for any $b \in [|I|]$ without violating the block condition. Since at most one row is assigned to a block part, the upper row and upper column load bounds are not exceeded. Hence, the $|I|$-decomposition $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_{|I|}, \mathcal{R}_B; \mathcal{C}_1 \ldots, \mathcal{C}_{|I|}, \emptyset)$ is a feasible solution for MinBf$(A, m, 0, 1)$ with objective value less or equal than $m - q$.

□

For the problem MinBf with $k = 2$, we can give a stronger result and obtain an additive non-approximability factor even for matrices with at most three nonzero entries in every row and at most two nonzero entries in every column.

3.4 Complexity for MinBf with $k = 2$

Definition 3.4.1 (α-vertex separator)
Let $G = (V, E)$ be an undirected graph with $|V| = m$ and let $\alpha \in \mathbb{R}$ be fixed with $\frac{1}{2} \leq \alpha < 1$. Moreover, let $P = (V_1, V_2, V_3)$ be a weak partition of the nodes of G. If $max(|V_1|, |V_2|) \leq \alpha m$, and no edge has one vertex in V_1 and the other in V_2, then we call P an *α-vertex separator*.

Minimum α-vertex separator ***Instance:*** $G = (V, E)$ an undirected graph and $\alpha \in \mathbb{R}$ with $\frac{1}{2} \leq \alpha < 1$ ***Solution:*** $P = (V_1, V_2, V_3)$ an α-vertex separator of G ***Objective:*** Minimize $|V_3|$

Theorem 3.4.2
Let $\varepsilon > 0$. Let $G = (V, E)$ be an undirected graph with m nodes that has a maximum node degree of three and let α be a fixed ratio with $\frac{1}{2} \leq \alpha < 1$. Let OPT *be the value of an optimal α-vertex separator for G and α. Unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial algorithm that finds an α-vertex separator for G and α with objective value smaller than* $\mathrm{OPT} + m^{\frac{1}{2}-\varepsilon}$.

Proof: This result was obtained by Bui and Jones. We omit the proof and refer the reader to [13].

□

We can use this result to get a non-approximability result for matrices with at most three nonzero entries in every row and at most two nonzero entries in every column.

Theorem 3.4.3
Let $\varepsilon > 0$. Let $A \in \mathbb{R}^{m \times n}$ be a matrix with at most three nonzero entries in every row and at most two nonzero entries in every column, $\ell \in \mathbb{N}_0$ and $u \in \mathbb{N}$. Let OPT *be the objective value of an optimal solution for* MinBf$(A, 2, \ell, u)$*. Unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial algorithm that finds a solution for* MinBf$(A, 2, \ell, u)$ *with objective value smaller than* $\mathrm{OPT} + m^{\frac{1}{2}-\varepsilon}$

Proof: Let $\varepsilon > 0$. We proof this theorem by contradiction. Suppose there is an algorithm $\mathcal{A}$ that finds a solution for MinBf$(A', 2, \ell, u)$ with objective function value smaller than $\mathrm{OPT} + m^{\frac{1}{2}-\varepsilon}$ for every matrix $A' \in \mathbb{R}^{m \times n}$ with at most three nonzero entries in every row and at most two nonzero entries in every column and for every $\ell \in \mathbb{N}_0$ and for every $u \in \mathbb{N}$. We will show that one could use $\mathcal{A}$ to find a feasible solution for minimum α-vertex seperator in polynomial time in the input size and objective value smaller than stated in Theorem 3.4.2. This would be a contradiction to Theorem 3.4.2.

For this purpose, we will transform an instance of the minimum α-vertex separator problem, including an undirected graph G with maximum node degree three, to an instance of the MinBf problem including a matrix that has at most three nonzero entries in every row and at most two nonzero entries in every column. We then show that one can obtain from every feasible solution of this MinBf instance a feasible solution of the minimum α-vertex separator problem and, vice versa. We will see that these tranformations can be done in polynomial time in the input size. Moreover, it will turn out that both feasible solutions have the same objective function value. Hence, both instances have the same optimal objective function value OPT.

At first, consider an instance I of the minimum α-vertex separator problem that is given by an undirected graph $G = (V, E)$ with $|V| = m$ and $|E| = n$, and a real number $\alpha \in \mathbb{R}$ with $\frac{1}{2} \leq \alpha < 1$. Now, we consider the incidence matrix $A \in \mathbb{R}^{m \times n}$ of G where every node of G corresponds to an row of A and every edge of G correpsonds to an column of A. We notice that every row of A has at most three nonzero entries and every column of A has at most two nonzero entries since G is an undirected graph with maximum node degree of three. We now consider an instance J of MinBf$(A, 2, 0, \lceil \alpha m \rceil)$.

We show now that every feasible solution of I can be transformed into a feasible solution of J with the same objective function value, and vice versa. We start with the transformation from a solution of J to a solution of I.

Let $\mathcal{D} = (\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_B; \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_B)$ be a 2-decomposition that is a feasible solution for MinBf$(A, 2, 0, \lceil \alpha m \rceil)$. We construct an α-vertex separator $P = (V_1, V_2, V_3)$ from $\mathcal{D}$ in the following way: For every vertex $v_i \in V$ with corresponding row $i \in [m]$ of A, we distinguish three cases:

1. If $i \in \mathcal{R}_1$, then we put v_i to V_1.

2. If $i \in \mathcal{R}_2$, then we assign v_i to V_2.

3. If $i \in \mathcal{R}_B$, then we attach v_i to V_3.

Since the load condition$(0, \lceil \alpha m \rceil, 0, n)$ is fulfilled, we obtain

$$\begin{aligned} |V_1| &= |\mathcal{R}_1| \le \lceil \alpha m \rceil \quad \text{and} \\ |V_2| &= |\mathcal{R}_2| \le \lceil \alpha m \rceil. \end{aligned}$$

Due to the fact that the cardinality of a set is an integral number, we have

$$\max\{|V_1|, |V_2|\} \le \alpha m.$$

We assume by contradiction that there is an edge $e_j \in E$ with $v_{i_1} \in e_j$ and $v_{i_2} \in e_j$ for nodes $v_{i_1} \in V_1$ and $v_{i_2} \in V_2$. Column j of A, that corresponds to e_j, would have nonzero entries in two rows of A, namely the rows i_1 and i_2. It holds that $r_1 \in \mathcal{R}_1$ and $r_2 \in \mathcal{R}_2$ because $v_{i_1} \in V_1$ and $v_{i_2} \in V_2$. Since $\mathcal{C}_B$ is empty, we have either $j \in \mathcal{C}_1$ or $j \in \mathcal{C}_2$. This is a contradiction to the fact that $\mathcal{D}$ fulfills the block condition because $a_{i_1 j} \neq 0$ and $a_{i_2 j} \neq 0$. Therefore, $P = (V_1, V_2, V_3)$ is a feasible solution of I with objective function value $|V_3| = |\mathcal{R}_B|$.

On the other hand let $P = (V_1, V_2, V_3)$ be a feasible solution of I. We now construct a 2-decomposition $\mathcal{D} = (\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_B, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_B)$ from P that fulfills the block condition and the load condition $(0, \lceil \alpha m \rceil, 0, n)$, and therefore is a feasible solution for $\textsc{MinBf}(A, 2, 0, \lceil \alpha m \rceil)$. We construct a weak partition of the rows from P as above, but in reverse: For every row $i \in [m]$ and its corresponding node v_i we distinguish between three cases:

1. If $v \in V_1$, then we put r to $\mathcal{R}_1$.

2. If $v \in V_2$, then we attach r to $\mathcal{R}_2$.

3. If $v \in V_3$, then we assign r to $\mathcal{R}_B$.

In the following, we construct a weak partition of the columns. Every column $j \in [n]$ of A has nonzero entries in two rows. These are the rows i_1 and i_2 with corresponding nodes v_{i_1}, $v_{i_2} \in V$. Since P is an α-edge separator, it neither holds that $v_{i_1} \in V_1$ and $v_{i_2} \in V_2$ at the same time, nor that $v_{i_2} \in V_1$ and concurrently $v_{i_1} \in V_2$. Hence, it neither holds that $i_1 \in \mathcal{R}_1$ and concurrently $i_2 \in \mathcal{R}_2$, nor $i_2 \in \mathcal{R}_1$ and $i_1 \in \mathcal{R}_2$. Therefore, we need only consider the following three cases:

1. If $i_1, i_2 \in \mathcal{R}_q$ with $q \in \{1, 2\}$, then we put j to $\mathcal{C}_q$.

2. If $i_1, i_2 \in \mathcal{R}_B$, then it actually does not matter to which column block j is assigned. We just attach j to $\mathcal{C}_1$.

3. If $i_1 \in \mathcal{R}_B$,$i_2 \in \mathcal{R}_q$ or $r_2 \in \mathcal{R}_B$,$r_1 \in \mathcal{R}_q$ for some $q \in \{1, 2\}$, then we assign j to $\mathcal{C}_q$.

Since $(\mathcal{C}_1, \mathcal{C}_2)$ is a weak partition of the columns of A, $\mathcal{D} = (\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_B, \mathcal{C}_1, \mathcal{C}_2, \emptyset)$ is a 2-decomposition of A.

For every column $j \in \mathcal{C}_q$ with $q \in \{1, 2\}$, there are two rows $i_1, i_2 \in [m]$ with $a_{ij} \neq 0$. From the above construction, it follows that $i_1, i_2 \notin \mathcal{R}_t$ with $t \in \{1, 2\} \setminus \{q\}$. Therefore, $\mathcal{D}$ fulfills the block condition.

Due to the fact that P is an α-vertex seperator, we get

$$\max\{|V_1|, |V_2|\} \leq \alpha m.$$

Hence, we obtain

$$\begin{aligned} |\mathcal{R}_1| = |V_1| \leq \alpha m \leq \lceil \alpha m \rceil \quad &\text{and} \\ |\mathcal{R}_2| = |V_2| \leq \alpha m \leq \lceil \alpha m \rceil. & \end{aligned}$$

Therefore, the load condition$(0, \lceil \alpha m \rceil, 0, n)$ is fulfilled and thus mp is a feasible solution for J.

Note that after both transformations

$$|V_3| = |\mathcal{R}_B|$$

holds. Thus, the objective function values are equal. Also note that both transformations are polynomial in the size of the input.

If we applied $\mathcal{A}$ to the MinBf instance J which we have got from the transformation, we would obtain a feasible solution of MinBf with objective function value smaller than $\text{OPT} + m^{\frac{1}{2}-\varepsilon}$. From this solution one can obtain a feasible solution of I with objective function value smaller than $\text{OPT} + m^{\frac{1}{2}-\varepsilon}$ which is a contradiction to Theorem 3.4.2. Hence, unless $\mathcal{P} = \mathcal{NP}$, there could not be such an algorithm $\mathcal{A}$.

□

Remark 10:
By Lemma 3.3.1, Theorem 3.4.2 can be extended to the problem MinAf.

Remark 11:
Since Theorem 3.3.3 covers the special case $u^{\mathcal{R}} = 1$, it is not redundant.

We have shown that it is $\mathcal{NP}$-hard to get a solution with objective value within an additive factor of the optimal objective value even for only two blocks, and martrices with at most three nonzero entries per row and at most two nonzero entries per column. Unfortunately, it is not clear if it is still difficult to find solution that are within a constant multiplicative factor of the optimal solution.

4 Heuristic Decomposing Methods

In the last chapter, we have seen the similarities in difficulty of detecting decompositions and graph partitioning. Now, we want to exploit these similarities to find block structures. We present four heuristic methods that solve graph partitionining problems on special graphs and hypergraphs representing the structure of the nonzero entries of the matrix.

At first, we introduce these graph partitioning problems, and after a brief summary of the literature on graph partitioning, we present a way to solve them. Afterwards, the heuristic methods for matrix decomposing will be introduced. All four algorithms basically follow one generic procedure. We sketch the similarities of the heuristics in Section 4.2. We then describe all four heuristic approaches in detail. Additionally, we indicate a relevant counterexample for each approach.

Throughout this chapter, $A \in \mathbb{R}^{m \times n}$ denotes a matrix and $k \in \mathbb{N}$ an integer. We assume without loss of generality that A has neither empty rows nor empty columns, these are rows or columns, respectively, with all entries equal zero. Otherwise, we delete these rows and columns, and try to find a k-decomposition for the remaining matrix. If we want these empty rows and columns to be in the decomposed matrix, then we assign them to some block row parts and column block parts, respectively, such that the load condition of the problem is still fulfilled. The rows and columns that could not be assigned because the load condition would be violated, are assigned to the respective border.

4.1 Solving Hypergraph Partitioning Problems

Hypergraph partitioning is an interesting problem with many applications, for example VLSI design [28] and data mining [35]. In this section, we introduce two graph partitioning problems on hypergraphs that will be used to obtain solutions for MinBf and MinAf. Both problems deal with finding a partition $P = (P_1, \ldots, P_k)$ of the vertices of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ for some integer $k \in \mathbb{N}$. The set of hyperedges that span multiple partitions is denoted by $Cut(P) := \{e \in \mathcal{E} | \exists u, v \text{ with } u \in e, v \in e \text{ and } u \in P_i, v \in P_j \text{ for some } i, j \in [k] \text{ with } i \neq j\}$. We call $Cut(P)$ the *edge cut* of P. Moreover, we call a partition $P = (P_1, \ldots, P_k)$ of $\mathcal{V}$ a *k-way α-hyperedge separator* if $|P_i| \leq \alpha \frac{|\mathcal{V}|}{k}$ for $i \in [k]$ with $\alpha \in \mathbb{R}$ and $\alpha \geq 1$. Given a weight function on the edges $w : \mathcal{E} \to \mathbb{R}_+$, the *weight* of an edge cut is the sum of the weights of its edges. The *weight* of a k-way α-hyperedge separator is the weight of its edge cut. Our first problem is about finding a minimum weighted k-way α-hyperedge separator:

Minimum weighted k-way α-hyperedge separator (HES)
Instance: $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ a hypergraph with a weight function on the hyperedges $w : \mathcal{E} \to \mathbb{R}_+$, an integral number $k \in \mathbb{N}$ and a real number $\alpha \geq 1$.
Solution: $P = (P_1, ..., P_k)$ a k-way α-hyperedge separator .
Objective: Minimize $\sum_{e \in Cut(P)} w(e)$, the weight of the edge cut of P.

If we deleted the hyperedges in the edge cut from the hypergraph, the remaining hypergraph would be decomposed into k components of balanced size. Instead of deleting hyperedges, the next problem is about finding a subset of vertices with minimum weight, such that after deleting this subset, the hypergraph decomposes into k components of balanced size:

Definition 4.1.1 (k-way α-hypervertex separator)
Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph, $k \in \mathbb{N}$ and $\alpha \in \mathbb{R}$ with $\alpha \geq 1$. Let $P = (P_1, \ldots, P_k, S)$ a weak partition of $\mathcal{V}$ with $P_t \neq \emptyset$ for all $t \in [k]$. We call P a *k-way α-hypervertex separator* if the following two conditions are fulfilled:

- $|P_t| \leq \alpha \frac{|\mathcal{V}|}{k}$ holds for $t \in [k]$.
- For all vertices $u, v \in \mathcal{V}$ with $u \in P_i$ and $v \in P_j$ for $i \neq j$, there is no edge $e \in \mathcal{E}$ with $u \in e$ and $v \in e$.

By deleting the vertices of the set S, we would obtain a hypergraph that decomposes into k components of balanced size. The *weight* of a k-way α-hypervertex separator is the cardinality of S.

k-way α-hypervertex separator (HVS)
Instance: $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ a hypergraph, an integral number $k \in \mathbb{N}$ and a real number $\alpha \geq 1$.
Solution: $P = (P_1, ..., P_k, S)$ a k-way α-hypervertex separator
Objective: Minimize $|S|$, the weight of P.

We denote HES$(\mathcal{H}, w_{\mathcal{E}}, k, \alpha)$ as the problem of finding a k-way α-hyperedge separator for the hypergraph $\mathcal{H}$ with minimum weight according to the hyperedge weight function $w_{\mathcal{E}}$. Sometimes we will just write HES$(\mathcal{H}, k, \alpha)$, then it does not matter which edge weighting function is used. Similarly, we denote HVS$(\mathcal{H}, k, \alpha)$ as the problem of finding a k-way α-hypervertex separator for the hypergraph $\mathcal{H}$ with minimum weight.

Observation 4.1.2
The HVS *problem is a generalization of the* Minimum α-vertex separator *problem presented in Section 3.4.*

Literature on Graph Partitioning

There is a wealth of literature on both problems even restricted to graphs and $k = 2$. It would go far beyound the scope of this thesis to study both problems in detail. Nevertheless, we give a brief overview before we present how we will solve the problems HVS and HES. One of the first important results was found by Lipton and Tarjan [32]. They showed that every planar graph with n vertices has a vertex separator of size $\mathcal{O}(\sqrt{n})$ that can be found in polynomial time. Later similar results could be found for further classes of graphs, e.g. in 1984 by Gilbert and Hutchinson [20], and by Alon et al. [5]. Furthermore, Bui and Jones [13] proved that finding an edge separator and vertex separator with size within an additive factor from the size of an optimal separator is $\mathcal{NP}$-hard . More recently, Feige et al. [16] obtained an $\mathcal{O}(\sqrt{\log opt})$ pseudoapproximation algorithm for finding a vertex separator with *opt* is the size of an optimal vertex separator. They obtained this result by applying an approximation algorithm that finds an edge separator with an approximation ratio of $\mathcal{O}(\sqrt{\log opt})$ which was published in 2004 by Aroa, Rao and Vazirani [6].

Furthermore, Alber and Fernau [4] give a parameterized view on graph separators. Schloegel [37] et al. presents an overview of graph partitioning algorithms. However, these papers deals with graph partitioning for graphs, but our problems cope hypergraphs. Ihler, D. Wagner and F. Wagner [22] published in 1993 a paper dealing with modeling hypergraphs with graphs, and therefore could be used to extend some of the results to hypergraphs. Finally, we want to mention the chapter “Hypergraph Partitioning and Clustering” in the book “Handbook of Approximation Algorithms and Metaheuristics” [36].

As one can see, there are numerous ways of solving graph partitioning problems. So, as not to go beyound the scope of this work, we decided to go one way to solve the problems HES and HVS. We are going to make use of Metis [27] and hMetis [29] to solve HES on undirected graphs and hypergraphs, respectively. Metis and hMetis are covered in a more detailed way in Chapter 6. The problem HVS is coped indirectly by solving a HES problem:

4.1.1 Heuristic for solving HVS by HES

One can solve an HVS problem indirectly by solving a HES problem on the same graph, possibly with special weights on the edges. There are different edge weighting schemes possible. Our choice of the weighting scheme depends on the underlying matrix decomposing heuristic. We will describe this in a more detailed way in the section of the respective matrix decomposing heuristic. After choosing an edge weighting scheme, we solve the HES problem heuristically with Metis [27] (or hMetis [29]) and find a feasible solution P with the corresponding edge cut $cut(P)$. Then, we try to obtain a solution for HVS from a graph that we have created according to $cut(P)$. We call this undirected graph the *edge cut graph* of P. The edge cut graph contains all nodes that are incident to a hyperedge of the edgecut. Two nodes are connected by an edge if and only if there is an hyperedge in the edge cut that connects these nodes.

Definition 4.1.3 (Edge cut graph)
Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph, $k \in \mathbb{N}$ an integer, $P = (P_1, \dots, P_k)$ a partition of $\mathcal{V}$ and $Cut(P)$ the corresponding edgcut of P. We define the *edge cut graph* of P to be the undirected graph $G_P^{\mathcal{H}} = (V, E)$ with

- $V := \{v \in \mathcal{V} | \exists e \in Cut(P) \text{ with } v \in e\}$ and
- $E := \{(v_1, v_2) \in V \times V | v_1 \in P_{i_1}, v_2 \in P_{i_2}$ for some $i_1, i_2 \in [k]$ with $i_1 \neq i_2$ and there is an edge $e \in Cut(P)$ such that $v_1 \in e$ and $v_2 \in e\}$.

After creating the edge cut graph, we are going to solve the well-known Minimum weighted vertex cover problem on it. For the sake of completeness, we define:

Definition 4.1.4 (Vertex cover)
Let $G = (V, E)$ be an undirected graph. A subset of nodes $X \subseteq V$ is called *vertex cover* of G if every edge $e \in E$ has at least one endpoint in X.

Minimum weighted vertex cover
Instance: $G = (V, E)$ a undirected graph with a weight function on the vertices $w \colon V \to \mathbb{R}_+$.
Solution: $X \subseteq V$, such that X is a vertex cover of G.
Objective: Minimize $\sum_{v \in X} w(v)$, the weight of the vertex cover X.

We will present some more information about the minimum weighted vertex cover problem in the end of this section.

From the found vertex cover X, often a solution for HVS can be constructed. This can be done by deleting the vertices of X from their corresponding set in $P = (P_1, \dots, P_k)$, with P is the k-way α-hyperedge separator that is feasible for the above solved HES problem. We define $P' := \{P'_1, \dots, P'_k, X\}$ with $P'_i := P_i \setminus (X \cap P_i)$ for $i \in [k]$. Although

$$|P'_i| \leq |P_i| \leq \alpha \frac{|\mathcal{V}|}{k} \text{ holds for all } i \in [k],$$

it is not clear that P' is a k-way α-hypervertex separator because P'_i could be empty for some $i \in [k]$. We will present an example later on. It is unclear which edge weighting schemes for HES lead to feasible solutions for HVS. It is also ambiguous how to guarantee the quality of the feasible solutions. In practice, however, this heuristic is performing decently. Our decision about the edge weighting scheme is based on the underlying matrix decomposing heuristics, as we will describe later. For a closer look on the topic of finding vertex separators we recommend the work by Liu [33].

In order to formalize this approach, we present a scheme for the indirect solving of HVS by HES in Algorithm 4 that summarizes the above explanations.

Algorithm 4: IndirectHVS

input : $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ a hypergraph, , an integer $k \in \mathbb{N}$, $\alpha \in \mathbb{R}$ with $\alpha \geq 1$ and information which heuristic model is underlying.

output: $P' = (P'_1, ..., P'_k, X)$ a k-way α-hypervertex separator or a statement that no k-way α-hypervertex separator could be found.

create edge weight function $w_\mathcal{E} \colon \mathcal{E} \to \mathbb{R}_+$ *according to the underlying heuristic model*;
$P = (P_1 \ldots, P_k) \leftarrow$`solveHES(`$\mathcal{H}, w_\mathcal{E}, k, \alpha$`)`;
if P *is not a* k*-way* α*-hyperedge separator* **then**
    output statement that no k*-way* α*-hypervertex separator could be found and quit;*
end
create edge cut graph $G_P^\mathcal{H}$ *according to* P *and a weight function* $w_\mathcal{V} \colon \mathcal{V} \to \mathbb{R}_{>0}$ *with* $w_\mathcal{V}(v) = 1$ *for all* $v \in \mathcal{V}$;
$X \leftarrow$`SolveMinimumWeightedVertexCover(`$G_P^\mathcal{H}, w_\mathcal{V}$`)`;
create $P' = (P'_1 \ldots, P'_k, X)$ *from* P *and* X;
if P' *is a* k*-way* α*-hypervertex separator* **then**
    return P';
end
output statement that no k*-way* α*-hypervertex separator could be found and quit*;

In general, we will use three different edge weighting schemes. These are the *unary*, the *prop size*, and the *aprop degree* weighting scheme. For an arbitrary hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ we say that an edge weight function $w_\mathcal{E} \colon \mathcal{E} \to \mathbb{R}_+$ *follows*

$$\text{the } \textit{unary weighting scheme} \text{ if} \quad w_\mathcal{E}(e) = 1, \quad \text{for all } e \in \mathcal{E}, \quad \text{and it follows} \tag{4.1}$$
$$\text{the } \textit{prop size weighting scheme} \text{ if} \quad w_\mathcal{E}(e) = |e| = |\{v \in \mathcal{V} | v \in e\}| \text{ for all } e \in \mathcal{E}. \tag{4.2}$$

Moreover, for an arbitrary undirected graph $G = (V, E)$, we say that an edge weight function $w_E \colon E \to \mathbb{R}_+$ *follows* the *aprop degree weigthing scheme*

$$\text{if} \quad w_E(e) = \left\lceil \frac{|V|}{max(d(i), d(j))} \right\rceil \quad \text{for all } e = (i, j) \in E, \tag{4.3}$$

with $d(i)$ is the degree of node $i \in V$. Aykanat [7] et al. introduced the following weighting scheme that is similar to the latter one :

$$w_E(e) = \frac{1}{max(d(i), d(j))},$$

for all $e = (i, j) \in E$. Since the graph partitioning software we use, can only handle integral weights, we adapted the weighting scheme. To simplify notation we assume that the information about the edge weighting scheme is stored in the corresponding hypergraph itself. By default we solve a HES problem by using the unary weighting scheme. Optionally, we will use the other ones.

The Minimum Weighted Vertex Cover Problem

The MINIMUM WEIGHTED VERTEX COVER problem with $w(v) = 1$ for all $v \in V$ is one of the classical $\mathcal{NP}$-hard problems that were presented by Karp [24] in 1972. The literature on this problem is vast and would go far beyound the scope of this work. Though, we mention the most recent work. Dinur and Safra [15] showed in 2004 that the problem cannot be approximated within a factor of 1.36. Furthermore, Karakostas [23] presented an algorithm that has an approximation ratio of $2 - \Theta\left(\frac{1}{\sqrt{\log n}}\right)$ which was published in 2009.

Although we just need to deal with the unweighted problem, we will use a simple greedy heuristic that is designed to cope even the weighted problem. For the sake of completeness, it is presented in Algorithm 5 . Note, that $adj(v) := \{u \in V : (u, v) \in E\}$.

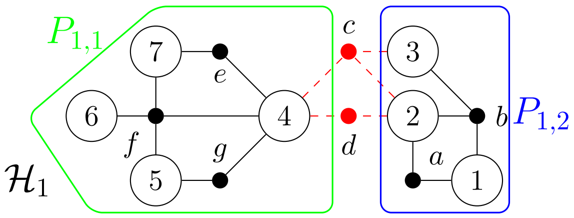
Algorithm 5: SolveMinimumWeightedVertexCover

input : $G = (V, E)$ an undirected graph, a weight function on the vertices $w\colon V \to \mathbb{R}_+$.
output: $X \subseteq V$ a vertex cover of G.

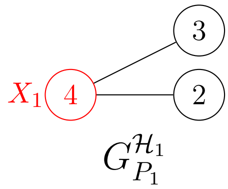
$X \leftarrow \emptyset$;
$H \leftarrow V$;
for $v \in V$ **do**
  $score(v) \leftarrow \sum_{u \in adj(v)} w(u)$;
end
while X *is no vertex cover of* G **do**
  choose $v \in H$ *with* $score(v) \geq score(u)$ *for all* $u \in H$;
  $X \leftarrow X \cup \{v\}$;
  $H \leftarrow H \setminus \{v\}$;
  for $u \in adj(v)$ **do**
    $score(u) \leftarrow score(u) - w(v)$;
  end
end
return X;

Finally, we present two small examples. While the first illustration shows a successful run, the second example illustrates a run that does not yield a feasible solution for HVS. We are going to solve the problems $\mathrm{HVS}(\mathcal{H}_i, 2, 1.5)$ for $i \in \{1, 2\}$. Both figures consist of three subfigures. For $i \in \{1, 2\}$, let w^i_{un} be an appropriate edge weight function for $\mathcal{H}_i$ that follows the unary weighting scheme. The first subfigure shows the given hypergraph $\mathcal{H}_i$ and the found solution P_i for $\mathrm{HES}(\mathcal{H}_i, w^i_{un}, 2, 1.5)$ visualized by the colored framings around the vertices of $\mathcal{H}_i$. The second one presents the corresponding edge cut graph $G^{\mathcal{H}_i}_{P_i}$ and a minimum vertex cover X_i. Finally, the third subfigure shows $\mathcal{H}_i$ with the deduced potential solution for $\mathrm{HVS}(\mathcal{H}_i, 2, 1.5)$.

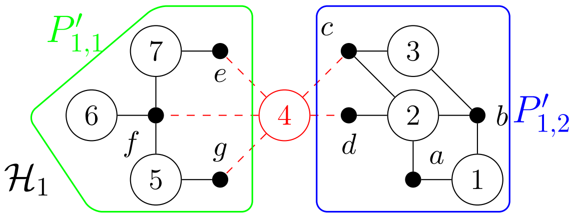
As figure 4.2c shows, Algorithm 4 can yield an infeasible solution, since $P'_{2,1} = \emptyset$.



(a) Hypergraph $\mathcal{H}_1$ with found solution $P_1 = (P_{1,1}, P_{1,2})$ for HES and edge cut $cut(P_1) = \{c, d\}$

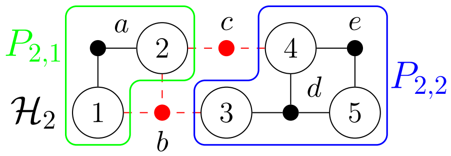


(b) Edge cut graph $G^{\mathcal{H}_1}_{P_1}$ with vertex cover $X_1 = \{4\}$

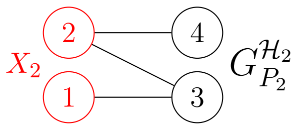


(c) Hypergraph $\mathcal{H}_1$ with obtained solution $P'_1 = (P'_{1,1}, P'_{1,2}, \{4\})$ for HVS

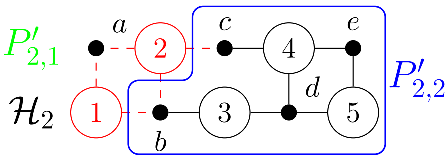
Figure 4.1: Successful run of IndirectHVS



(a) Hypergraph $\mathcal{H}_2$ with found solution $P_2 = (P_{2,1}, P_{2,2})$ for HES and edge cut $cut(P_1) = \{b, c\}$



(b) Edge cut graph $G^{\mathcal{H}_1}_{P_1}$ with vertex cover $X_2 = \{1, 2\}$



(c) Hypergraph $\mathcal{H}_2$ with obtained infeasible solution $P'_2 = (\emptyset, P'_{2,2}, \{1, 2\})$ for HVS

Figure 4.2: Failed run of IndirectHVS

4.2 Modeling Matrix Decomposing Problems as Graph Partitioning Problems

This short section is an introduction to the next two subsequent sections. Before we propose our heuristic approaches in detail, we sketch the commonalities of all four approaches. Essentially, all of them follow the generic procedure displayed in Figure 4.3. As one can see, every algorithm provides two kind of outputs: Either it returns a message that no solution could be found or it returns a feasible solution to the respective matrix decomposing problem.

Remark 12:
Since the algorithms are heuristics, propositions about the found solutions are rather weak. Nevertheless, we are going to prove statements of the following form: "If algorithm X does not quit by sending a message, then it returns a feasible solution for problem Y." In this context, we will call algorithm X "heuristically correct" and the corresponding propositions "Heuristic Correctness of Algorithm X". From a theoretical point of view, these kind of results are rather weak but it will turn out that in practice the heuristics often quit by returning a solution that is hence feasible. In fact, these propositions state that if a tuple of tuples of rows and columns $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ is returned (in this case no message is returned), then $\mathcal{D}$ is a k-decomposition that fulfills the block condition. (It is already verified in the algorithm that $\mathcal{D}$ fulfills the load condition.)

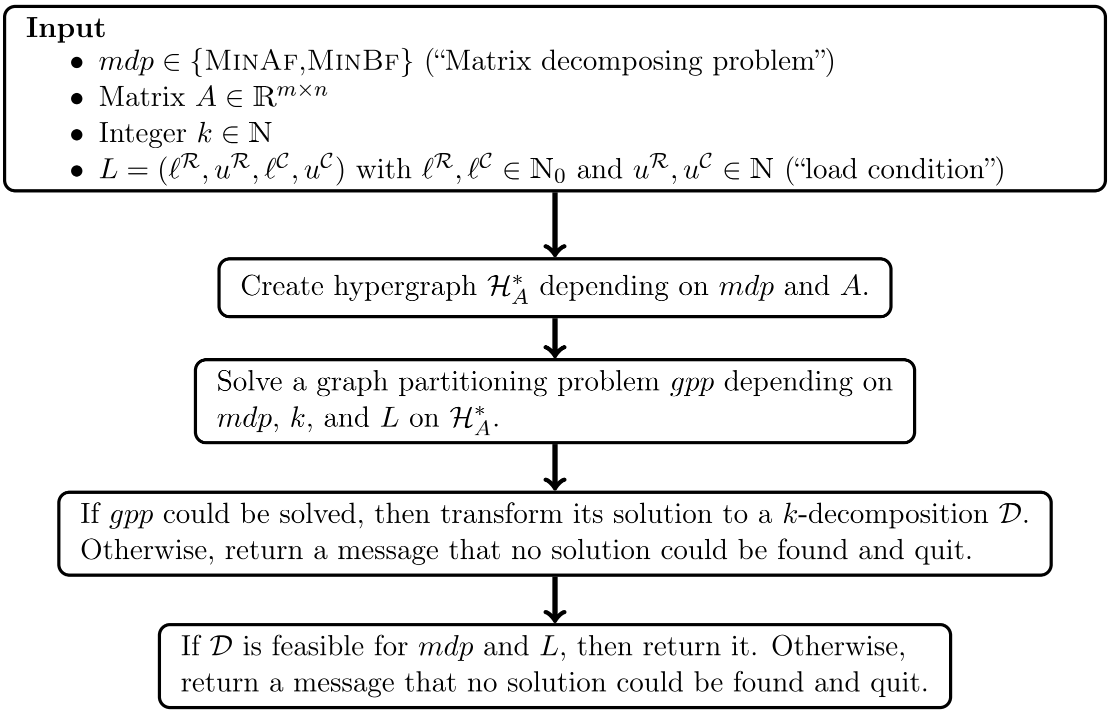


Figure 4.3: Sketch of the generic decomposing method

4.2.1 Outlook

Each of the next four subsections introduces one of the announced heuristic algorithms. These subsections have the same structure: To begin with, we define the graph that will be used. Secondly, the main idea of the algorithm is described. Afterwards, a small example of a successful run visualizes the algorithm. Then, the algorithm is described in detail. This includes a brief discussion about the choice of the parameters of the corresponding graph partitioning problem, namely the balance ratio α and possible weighting schemes. Eventually, the heuristic correctness of the algorithm is proven. Finally, a failed run is indicated.

Parameters

In the following, we introduce two parameters used in every matrix decomposing heuristic that will be presented in the remaining part of this chapter. To begin with, we will vary the method to solve the HES problem (HVS problems are solved indirectly by solving a HES problem). Secondly, we will optionally add dummy nodes. Dummy nodes are not adjacent to other nodes. Both parameters are applied to all heuristics and hence we will not describe them in detail here. However, they are described in Section 6.1.1 where all parameters are summarized.

4.3 Models for MinBf

In this section, we present two methods to solve the MinBf problem. Those are the *hyperrow decomposing algorithm* and the *hypercolumn decomposing algorithm*.

4.3.1 The Hyperrow Decomposing Algorithm

This subsection deals with the *hyperrow decomposing algorithm* which is similar to the row-net model of Aykanat [40]. At first, we define the *hyperrow graph* $\mathcal{H}_A^R$ of a matrix A. Then, we describe the main idea of the algorithm that uses the hyperrow graph. Thirdly, we show a small example to visualize it. After presenting the detailed algorithm, we are going to indicate an example run that fails.

Definition 4.3.1 (Hyperrow graph)
Let $A \in \mathbb{R}^{m \times n}$ be a matrix. We define the *hyperrow graph* of A to be the hypergraph $\mathcal{H}_A^R = (\mathcal{V}, \mathcal{E})$ with

- $\mathcal{V} = \{v_j : j \in [n]\}$ and
- $\mathcal{E} = \{e_i : i \in [m]\}$, with $v_j \in e_i$, if and only if $a_{ij} \neq 0$.

The hyperrow graph consists of vertices and hyperedges, representing the columns and rows, respectively, of A. Every hyperedge e connects exactly those vertices whose corresponding rows have a nonzero entry in the column belonging to e.

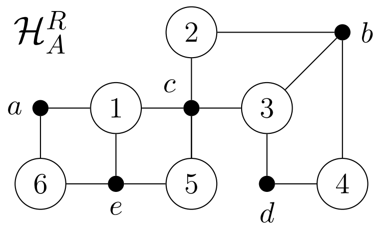
The idea of the hyperrow decomposing algorithm is the following: We are going to solve an HES problem on the hyperrow graph $\mathcal{H}_A^R$ of a matrix A with unit edge weights.

Thus, we obtain a partition $P = (P_1, \dots, P_k)$ of the vertices and a corresponding edge cut $Cut(P)$. The part P_t for $t \in [k]$ corresponds to block t. Every column that belongs to a vertex in P_t, will be assigned to the t-th column block part $\mathcal{C}_t$. The rows that are associated to hyperedges in the edge cut, become the border rows. Each remaining hyperedge spans vertices that belongs to the same part. We assign the corresponding row of this hyperedge to the row block part that corresponds this part. It will turn out that we thus obtain a k-decomposition.

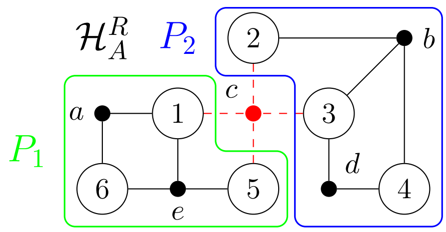
For a better illustration, we want to give a small example displayed in Figure 4.4. Consider matrix A presented in Subfigure 4.4a and its hyperrow graph $\mathcal{H}_A^R$ shown in Subfigure4.4b. From the (optimal) solution $P = (P_1, P_2)$ for $\text{HES}(\mathcal{H}_A^R, 2, 1.5)$, we can deduce a solution $\mathcal{D}$ for $\text{MinBf}(A, 2, 1, 5, 1, 6)$ for this example. The solution P is displayed in Subfigure 4.4c and the corresponding decomposed matrix $\mathcal{D}(A)$ is shown in Subfigure 4.4d.

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 1 & & & & & 1 \\ & 1 & 1 & 1 & & \\ 1 & 1 & 1 & & 1 & \\ & & 1 & 1 & & \\ 1 & & & & 1 & 1 \end{pmatrix} \end{array}$$

(a) Matrix $A \in \mathbb{R}^{5\times 6}$



(b) Hyperrow graph $\mathcal{H}_A^R$



(c) Hyperrow graph $\mathcal{H}_A^R$ with feasible HES solution $P = ((1,5,6),(2,3,4))$

$$\mathcal{D}(A) = \begin{array}{c} \\ a \\ e \\ b \\ d \\ c \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 5 & 6 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & & 1 & & & \\ 1 & 1 & 1 & & & \\ & & & 1 & 1 & 1 \\ & & & & 1 & 1 \\ 1 & 1 & & 1 & 1 & \end{pmatrix} \end{array}$$

(d) Decomposed matrix $\mathcal{D}(A)$ in bordered 2-block diagonal form with $\mathcal{D} = ((a,e),(b,d),(c);(1,5,6),(2,3,4),\emptyset)$

Figure 4.4: Successful run of the hyperrow decomposing algorithm

Consider the problem $\text{MinBf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ for $A \in \mathbb{R}^{m\times n}, k, u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ and $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$. When solving the problem $\text{HES}(\mathcal{H}_A^R, k, \alpha)$, the choice of α is an important point. If we chose α too big, then we could obtain a k-decomposition that does not fulfill the upper column block condition. On the other hand, if we chose α too small, then we might prune some solutions of HES that potentially could be transformed to feasible solutions of MinBf. We set $\alpha := \frac{u^{\mathcal{C}} k}{n}$ and notice that if $\alpha < 1$, then the instance $\text{MinBf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ is infeasible since $u^{\mathcal{C}} k < n$. Therefore, by solving

HES($\mathcal{H}_A^R, k, \frac{u^{\mathcal{C}}k}{n}$), we get a feasible solution $P = (P_1, \ldots, P_k)$ with:

$$|P_i| \leq \frac{u^{\mathcal{C}}k}{n}\frac{n}{k} = u^{\mathcal{C}},$$

for $i \in [k]$ and hence the upper column load condition is fulfilled.

The above described *hyperrow decomposing algorithm* is stated in detail in Algorithm 6. The implementation of method `CreateHyperrowGraph()` is clear from Definition 4.3.1. The method `SolveHES()` is solved by an HES solver that we will treat as 'black box' here. The method `TransformPartToDecomp()` is described in detail in Algorithm 7.

Algorithm 6: HyperrowDecomposingAlgorithm

input : $A \in \mathbb{R}^{m \times n}$ a matrix, $k \in \mathbb{N}$ an integer, $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$.
output: A k-decomposition $\mathcal{D}$ that fulfills the block condition and the load condition($\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}}$), a message that no solution exists or a message that no solution could be found.

$\mathcal{H}_A^R \leftarrow$ `CreateHyperrowGraph(`A`)` ;
$\alpha \leftarrow \frac{u^{\mathcal{C}}k}{n}$;
if $\alpha < 1$ **then**
  return message that no solution exists and quit;
end
$P = (P_1, \ldots, P_k) \leftarrow$ `SolveHES(`$\mathcal{H}_A^R$,k,α`)`;
if *P is not feasible for* HES*($\mathcal{H}_A^R$,k,α)* **then**
  return message that no feasible solution could be found and quit;
end
$\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B) \leftarrow$ `TransformPartToDecomp(`$\mathcal{H}_A^R$,k,P`)`;
if *$\mathcal{D}$ fulfills the load condition* $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ **then**
  return $\mathcal{D}$;
end
else
  return message that no feasible solution could be found;
end

Although Algorithm 6 is a heuristic, we can prove its heuristic correctness in terms of Remark 12:

Lemma 4.3.2 (Heuristic correctness of the hyperrow decomp. algorithm)
Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers.
If Algorithm 6 ends without sending a message for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, then it returns a k-decomposition that is feasible for MinBf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Proof: Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers. Suppose that Algorithm 6 ends with-

Algorithm 7: TransformPartToDecomp

input : $\mathcal{H}_A^R = (\mathcal{V}, \mathcal{E})$ the hyperrow graph of some matrix $A \in \mathbb{R}^{m \times n}$, $k \in \mathbb{N}$ an integer, and $P = (P_1, \dots, P_k)$ a solution of $\mathrm{HES}(\mathcal{H}_A^R, k, \alpha)$ for some $\alpha \geq 1$.

output: A k-decomposition $\mathcal{D}$ that fulfills the block condition.

```
R_B ← [m];
for b ∈ [k] do
    R_b ← {i ∈ [m] | v ∈ P_b for all v ∈ e_i};
    C_b ← {j ∈ [n] | v_j ∈ P_b};
    R_B ← R_B ∖ R_b;
end
return D = (R_1, ..., R_k, R_B; C_1, ..., C_k, ∅);
```

out sending a message for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Then, it ends by returning $\mathcal{D}$ in line 12.

We will show that if line 10 is reached, the tuple $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ is a k-decomposition that fulfills the block condition and $\mathcal{C}_B = \emptyset$. Thus, if line 12 is reached, the k-decomposition $\mathcal{D}$ additionally fulfills the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, and hence $\mathcal{D}$ is feasible for $\textsc{MinBf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Since the algorithm quits not by sending a message, Algorithm 6 reaches line 12 for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Consider the tuple $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ which has been returned by the method `TransformPartToDecomp()` in line 10. Let $\mathcal{H}_A^R$ be the hyperrow graph of A and $P = (P_1, \dots, P_k)$ the solution of $\mathrm{HES}(A, k, \frac{u^{\mathcal{C}} k}{n})$ that was returned in line 6. At first, we will show that $(\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A, and that $(\mathcal{C}_1, \dots, \mathcal{C}_k)$ is a weak partition of the columns of A. Finally, we show that $\mathcal{D}$ fulfills the block condition. In line 10 of Algorithm 6 we have:

- $\mathcal{R}_b = \{i \in [m] | v \in P_b \text{ for all } v \in e_i\}$ for all $b \in [k]$,
- $\mathcal{C}_b = \{j \in [n] | v_j \in P_b\}$ for all $b \in [k]$,
- $\mathcal{R}_B = [m] \setminus \bigcup_{b \in [k]} \mathcal{R}_b$, and
- $\mathcal{C}_B = \emptyset$.

Therefore, it is

$$\bigcup_{b \in [k]} \mathcal{R}_b \cup \mathcal{R}_B = \bigcup_{b \in [k]} \mathcal{R}_b \cup \left([m] \setminus \bigcup_{b \in [k]} \mathcal{R}_b \right) = [m],$$

and for all $b \in [k]$ holds $\mathcal{R}_b \cap \mathcal{R}_B = \emptyset$, obviously. Moreover, for $b, b' \in [k]$ with $b \neq b'$ it is $\mathcal{R}_b \cap \mathcal{R}_{b'} = \emptyset$, because of the following fact: If there was a row $i \in [m]$ with $i \in \mathcal{R}_b$ and

$i \in \mathcal{R}_{b'}$, then for all $v \in e_i$ ($e_i \neq \emptyset$ since A has no empty rows) would hold that $v \in P_b$ and $v \in P_{b'}$, contradicting the fact that P is a partition of the vertices of $\mathcal{H}_A^R$. Hence, $(\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A.

Let $j \in [n]$ be an arbitrary column. There is exactly one $b \in [k]$ with $v_j \in P_b$ because $(P_1, \ldots, P_k)$ is a partition of the vertices of $\mathcal{H}_A^R$. Hence, $j \in \mathcal{C}_b$ and $j \notin \mathcal{C}_{b'}$ for $b' \in [k]$ with $b' \neq b$. Therefore, $(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ is a partition of the columns of A. Thus, $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \emptyset)$ is a k-decomposition of A.

Now, we are going to show that $\mathcal{D}$ fulfills the block condition. Consider $i \in \mathcal{R}_b$ and $j \in \mathcal{C}_{b'}$ with $a_{ij} \neq 0$ for some $b, b' \in [k]$. Since $j \in \mathcal{C}_{b'}$ holds, we have $v_j \in P_{b'}$. Furthermore, it holds that $v_j \in e_i$ because $a_{ij} \neq 0$. On account of $i \in \mathcal{R}_b$, we obtain that $v \in P_b$ for all $v \in e_i$. Therefore, in particular, $v_j \in P_b$ and $v_j \in P_{b'}$. Since $(P_1, \ldots, P_k)$ is a partition of the vertices, we have $b = b'$. Hence, the block condition is fulfilled.

Because line 12 is reached, $\mathcal{D}$ also fulfills the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Therefore, $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \emptyset)$ is a feasible solution for MinBf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. □

We want to emphasize that even if the solution of the HES problem is balanced, the obtained k-decomposition could fail the load condition. An example is indicated in Figure 4.5. Consider the problem MinBf$(A, 2, 1, 5, 1, 5)$ for the matrix $A \in \mathbb{R}^{5\times 5}$ displayed in Subfigure 4.5a. The hyperrow graph $\mathcal{H}_A^R$ of A is shown in Subfigure 4.5b. In Subfigure 4.5c a feasible solution $P = (P_1, P_2)$ for HES$(\mathcal{H}_A^R, 2, 2)$ is displayed. The deduced 2-decomposition is $\mathcal{D} = ((a, b, c), (), (); (1, 2, 3)(4, 5), ())$ that does not fulfill the load condition $(1, 5, 1, 5)$. The corresponding decomposed matrix $\mathcal{D}(A)$ is shown in Subfigure 4.5d. Obviously, it is not in bordered 2-block diagonal form.

4.3.2 The Hypercolumn Decomposing Algorithm

This subsection introduces the *hypercolumn decomposing algorithm.* It is structured similarly to the preceding subsection. To begin with, we recapitulate Definition 3.2.2, the hypercolumn graph of a matrix. Secondly, we present the main idea of the algorithm is to solve a k-way α-hypervertex separator problem on the hypercolumn graph of the matrix. This approach is akin to the column-net model of Aykanat [40]. Afterwards, we give a successful example for the sake of visualization. Moreover, the detailed algorithm will be indicated. Subsequently, we give an example run of the hypercolumn decomposing algorithm that is not successful. Finally, we will introduce an alternative weighting scheme that can be used when solving the k-way α-hypervertex separator problem indirectly.

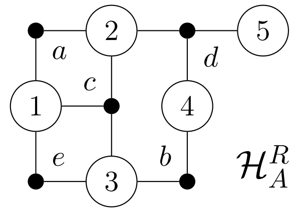
Let us recapitulate Definition 3.2.2: Given a matrix $A \in \mathbb{R}^{m\times n}$, the *hypercolumn graph* of A is defined to be the hypergraph $\mathcal{H}_A^C = (\mathcal{V}, \mathcal{E})$ with

- $\mathcal{V} = \{v_i : i \in [m]\}$ and
- $\mathcal{E} = \{e_j : j \in [n]\}$, such that $v_i \in e_j$, if and only if, $a_{ij} \neq 0$.

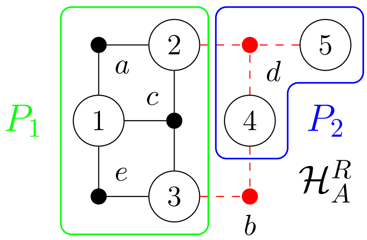
In this way, we denote v_i as the node that belongs to the i-th row of A and e_j as the hyperedge that belongs to the j-th column of A. We are going to solve a k-way α-hypervertex separator problem on $\mathcal{H}_A^C$, obtaining an α-hypervertex separator $P = (P_1, \ldots, P_k, S)$. Afterwards, this weak partition of the vertices is transformed

$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{pmatrix} 1 & 1 & & & \\ & & 1 & 1 & \\ 1 & 1 & 1 & & \\ & 1 & & 1 & 1 \\ 1 & & 1 & & \end{pmatrix} \end{array}$$

(a) Matrix $A \in \mathbb{R}^{5\times 5}$



(b) Hyperrow graph $\mathcal{H}_A^R$ of A



(c) $\mathcal{H}_A^R$ with $P = (P_1, P_2)$ the feasible solution for HES

$$\mathcal{D}(A) = \begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} a \\ c \\ e \\ b \\ d \end{array} & \begin{pmatrix} 1 & 1 & & & \\ 1 & 1 & 1 & & \\ 1 & & 1 & & \\ & 1 & 1 & 1 & \\ & 1 & & 1 & 1 \end{pmatrix} \end{array}$$

(d) Decomposed matrix $\mathcal{D}(A)$ for the deduced 2-decomposition $\mathcal{D} = ((a,b,c),(),();(1,2,3)(4,5),())$

Figure 4.5: Failed run of hyperrow decomposing algorithm: The deduced 2 - decomposition $\mathcal{D} = ((a,b,c),(),();(1,2,3)(4,5),())$ does not fulfill the load condition $(1,5,1,5)$.

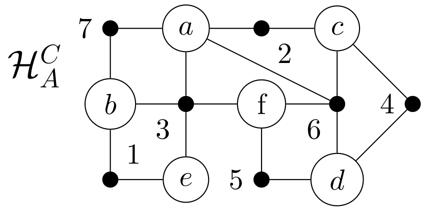
to a k-decomposition $\mathcal{D}$. This is done by assigning the rows that corresponds to vertices in part P_i to the row block part $\mathcal{R}_i$ and the rows corresponding to vertices in S to the row border $\mathcal{R}_B$. The columns are assigned in the following way: Column $j \in [n]$ is assigned to column block part $\mathcal{C}_t$ if there is a node $v \in e_j$ with $v \in P_t$. The remaining columns will be assigned to column block parts such that the upper column block load is fulfilled. If there are columns remaining after this step, the instance is infeasible. As we will see, $\mathcal{D}$ fulfills the block condition but the load condition may be violated.

Before we describe the algorithm in detail, we give a small example in Figure 4.6. We want to solve the problem MinBf$(A, 2, 1, 5, 1, 5)$ for the matrix $A \in \mathbb{R}^{6\times 7}$ shown in Subfigure 4.6a. The hypercolumn graph $\mathcal{H}_A^C$ of A is displayed in Subfigure 4.6b. The solution $P = (\{b,e\},\{c,d\},\{a,f\})$ of the HVS problem on $\mathcal{H}_A^C$ is shown in Subfigure 4.6c. One can deduce the 2-decomposition $\mathcal{D} = ((b,e),(c,d),(a,f);(1,3,7),(2,4,5,6),\emptyset)$ from P. The corresponding decomposed matrix $\mathcal{D}(A)$ is in bordered 2-block diagonal form, as one can see in Subfigure 4.6d.

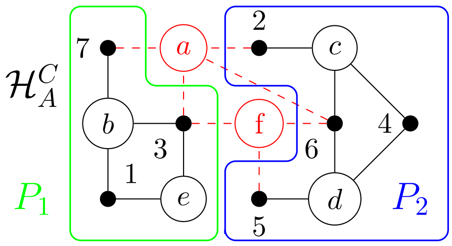
Before we describe the algorithm in detail, we have a look at the choice of α for the HVS problem from a theoretical point of view. We want to choose α big enough such that we do not lose solutions of HVS that potentially can be transformed to feasible solutions of MinBf. On the other hand, if we choose α to big, we maybe obtain a k-decomposition that does not fulfill the upper row load condition. We have to ensure that

$$A = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline a & & 1 & 1 & & & 1 & 1 \\ b & 1 & & 1 & & & & 1 \\ c & & 1 & & 1 & & 1 & \\ d & & & & 1 & 1 & 1 & \\ e & 1 & & 1 & & & & \\ f & & & 1 & & 1 & 1 & \end{array}$$

(a) Matrix $A \in \mathbb{R}^{6\times 7}$



(b) Hypercolumn graph $\mathcal{H}_A^C$ of A



(c) $\mathcal{H}_A^C$ with $P = (P_1, P_2, \{a, f\})$ the feasible solution of the corresponding HVS problem

$$\mathcal{D}(A) = \begin{array}{c|ccccccc} & 1 & 3 & 7 & 2 & 4 & 5 & 6 \\ \hline b & 1 & 1 & 1 & & & & \\ e & 1 & 1 & & & & & \\ c & & & & 1 & 1 & & 1 \\ d & & & & & 1 & 1 & 1 \\ a & & 1 & 1 & 1 & & & 1 \\ f & & 1 & & & & 1 & 1 \end{array}$$

(d) Decomposed matrix $\mathcal{D}(A)$ for $\mathcal{D} = ((b, e), (c, d), (a, f); (1, 3, 7), (2, 4, 5, 6), \emptyset)$

Figure 4.6: Successful run of the hypercolumn decomposing algorithm

$\alpha \geq 1$ and set $\alpha := \max(\frac{u^{\mathcal{R}}k}{m}, 1)$. Therefore, by solving HVS$(\mathcal{H}_A^R, k, \alpha)$ we get a feasible solution $P = (P_1, \ldots, P_k)$ with:

$$|P_i| \leq \frac{u^{\mathcal{R}}k}{m}\frac{m}{k} = u^{\mathcal{R}}, \text{ if } \frac{u^{\mathcal{R}}k}{m} \geq 1$$

for $i \in [k]$ and hence the upper row load condition is fulfilled if $\frac{u^{\mathcal{R}}k}{m} \geq 1$. On the other hand, if $\frac{u^{\mathcal{R}}k}{m} < 1$, then we cannot be sure that the obtained k-decomposition fulfills the upper row load constraint.

The following lemma states that Algorithm 8 is heuristically correct in terms of Remark 12.

Lemma 4.3.3 (Heuristic correctness of the hypercolumn decomp. algorithm)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers.
If Algorithm 8 ends without sending a message for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, then it returns a k-decomposition that is feasible for MinBf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Proof: Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers.

Algorithm 8: HypercolDecomposingAlgorithm

input : $A \in \mathbb{R}^{m \times n}$ a matrix, $k \in \mathbb{N}$ an integer, $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$, and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$.
output: A k-decomposition $\mathcal{D}$ that fulfills the block condition and the load condition($\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}}$), a statement that no solution exists or a statement that no solution could be found.

1 $\mathcal{H}_A^C \leftarrow$ `CreateHypercolumnGraph(`A`)`;
2 $\alpha \leftarrow \frac{u^{\mathcal{R}} k}{m}$;
3 **if** $\alpha < 1$ **then**
4   *return statement that no solution exists and quit*;
5 **end**
6 $P = (P_1, \ldots, P_k, S) \leftarrow$ `SolveHVS(`$\mathcal{H}_A^C$,k,α`)`;
7 **if** *P is not feasible for* HVS*($\mathcal{H}_A^C$,k,α)* **then**
8   *return statement that no feasible solution could be found and quit*;
9 **end**
10 $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B) \leftarrow$ `TransformPartToDecompHC(`$\mathcal{H}_A^C$,k,P`)`;
11 **if** *$\mathcal{D}$ fulfills the load condition ($\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}}$)* **then**
12   **return** $\mathcal{D}$;
13 **end**
14 **else**
15   *return statement that no feasible solution could be found*;
16 **end**

Algorithm 9: TransformPartToDecompHC

input : $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ a hypergraph, $k \in \mathbb{N}$ an integer, and $P = (P_1, \ldots, P_k, S)$ a solution of HVS problem on $\mathcal{H}$.
output: A k-decomposition $\mathcal{D}$ of A that fulfills the block condition.

1 $\mathcal{R}_B \leftarrow \{i \in [m] | v_i \in S\}$;
2 $\mathcal{C}_{\text{remain}} \leftarrow [n]$;
3 **for** $b \in [k]$ **do**
4     $\mathcal{R}_b \leftarrow \{i \in [m] | v_i \in P_b\}$;
5     $\mathcal{C}_b \leftarrow \{j \in [n] | \exists v \in e_j \text{ with } v \in P_b\}$;
6     $\mathcal{C}^{\text{remain}} \leftarrow \mathcal{C}^{\text{remain}} \setminus \mathcal{C}_b$;
7 **end**
8 *create a partition* $(\mathcal{C}_1^{remain}, \ldots, \mathcal{C}_k^{remain})$ *of the remaining columns in* $\mathcal{C}^{remain}$;
9 **for** $b \in [k]$ **do**
10     $\mathcal{C}_b \leftarrow \mathcal{C}_b \cup \mathcal{C}_b^{\text{remain}}\}$;
11 **end**
12 **return** $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \emptyset)$;

We will show that if line 10 is reached, the tuple $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$

is a k-decomposition with $\mathcal{C}_B = \emptyset$ that fulfills the block condition. Thus, if line 12 is reached, the k-decomposition $\mathcal{D}$ additionally fulfills the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, and therefore is feasible for MinBf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Let Algorithm 8 reach line 10 for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ and consider the tuple $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ which has been returned by the method `TransformPartToDecompHC()`. We notice that $\mathcal{C}_B = \emptyset$. Also, let $(\mathcal{C}_1^{\text{remain}}, \dots, \mathcal{C}_k^{\text{remain}})$ be the weak partition of the remaining columns $\mathcal{C}^{\text{remain}}$ that are not assigned to a column block part in line 7 of Method `TransformPartToDecompHC()`. Let $\mathcal{H}_A^C$ be the hypercolumn graph of A and $P = (P_1, \dots, P_k, S)$ the solution of HVS$(A, k, \frac{u^{\mathcal{R}} k}{m})$ that was returned in line 6. Analogously to the proof of Lemma 4.3.2, we will show that $(\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A, and that $(\mathcal{C}_1, \dots, \mathcal{C}_k)$ is a weak partition of the columns of A. Afterwards, we prove that $\mathcal{D}$ fulfills the block condition. In line 10 of Algorithm 8, it holds that:

- $\mathcal{R}_b = \{i \in [m] | v_i \in P_b\}$ for all $b \in [k]$,
- $\mathcal{C}_b = \{j \in [n] | \exists v \in e_j : v \in P_b\} \cup \mathcal{C}_b^{\text{remain}}$ for all $b \in [k]$, and
- $\mathcal{R}_B = \{i \in [m] | v_i \in S\}$.

For an arbitrary row $i \in [m]$ there is either exactly one $b \in [k]$ with $v_i \in P_b$ or $v_i \in S$, because $(P_1, \dots, P_k, S)$ is a weak partition of the vertices of $\mathcal{H}_A^C$. Therefore, there either is exactly one b with $i \in \mathcal{R}_b$ or $i \in \mathcal{R}_B$. Hence, $(\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A.

$$
\begin{aligned}
\bigcup_{b \in [k]} \mathcal{C}_b &= \left(\bigcup_{b \in [k]} \{j \in [n] | \exists v \in e_j : v \in P_b\} \right) \cup \left(\bigcup_{b \in [k]} \mathcal{C}_b^{\text{remain}} \right) \\
&= \left(\bigcup_{b \in [k]} \{j \in [n] | \exists v \in e_j : v \in P_b\} \right) \cup \mathcal{C}^{\text{remain}} \\
&= \left(\bigcup_{b \in [k]} \{j \in [n] | \exists v \in e_j : v \in P_b\} \right) \cup \left([n] \setminus \bigcup_{b \in [k]} \{j \in [n] | \exists v \in e_j : v \in P_b\} \right) \\
&= [n].
\end{aligned}
$$

It remains to show that $\mathcal{C}_{b1} \cap \mathcal{C}_{b_2} = \emptyset$ for $b_1, b_2 \in [k]$ with $b_1 \neq b_2$. Consider arbitrary $b_1, b_2 \in [k]$, with $\mathcal{C}_{b1} \cap \mathcal{C}_{b_2} \neq \emptyset$. Let $j^* \in \mathcal{C}_{b1} \cap \mathcal{C}_{b_2}$. There are two cases to consider:

1. The first case is that for all $v \in e_{j^*}$ it is true that $v \in S$. It then holds that $j^* \in \mathcal{C}^{\text{remain}}$. Hence, there is exactly one $b \in [k]$ with $j \in \mathcal{C}_b^{\text{remain}}$ because $(\mathcal{C}_1^{\text{remain}}, \dots, \mathcal{C}_k^{\text{remain}})$ is a weak partition of $\mathcal{C}^{\text{remain}}$. On the other hand, for all $b \in [k]$ it holds that $j^* \notin \{j \in [n] | \exists v \in e_j \text{ with } v \in P_b\}$. Thus, there is exactly one b with $j^* \in \mathcal{C}_b$. Therefore, we have $b_1 = b_2$.

2. In the second case there are $v_1, v_2 \in e_{j^*}$ with $v_1 \in P_{b_1}$ and $v_2 \in P_{b_2}$. Since $(P_1, \ldots, P_k, S)$ is a k-way α-hypervertex separator, it holds that $b_1 = b_2$.

Therefore, $(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ is a weak partition of the columns of A and hence , the tuple $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \emptyset)$ is a k-decomposition of A. It remains to show that $\mathcal{D}$ fulfills the block condition.

Consider $i \in \mathcal{R}_{b_1}$ and $j \in \mathcal{C}_{b_2}$ with $a_{ij} \neq 0$ for some $b_1, b_2 \in [k]$. Because $i \in \mathcal{R}_{b_1}$, it holds that $v_i \in P_b$ and as $a_{ij} \neq 0$, we have $v_i \in e_j$. Hence, by the definition of $\mathcal{C}_{b_1}$, we obtain that $j \in \mathcal{C}_{b_1}$. Due to the fact that $(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ is a weak partition of the columns of A, we obtain $b_1 = b_2$. Thus, the block condition is fulfilled.

□

In the following we give an example run that fails. It is illustrated in Figure 4.7. Let us consider the problem $\textsc{MinBf}(A, 2, 1, 6, 1, 6)$ with matrix $A \in \mathbb{R}^{6\times 9}$ displayed in Subfigure 4.7a. The hypercolumn graph $\mathcal{H}_A^C$ and a solution $P = (\{c, e\}, \{a, d, f\}, \{b\})$ of the corresponding HVS problem are displayed in Subfigure 4.7b. Although P is balanced, the deduced 2-decomposition $\mathcal{D} = ((c, e), (a, d, f), (b); (1, 8), (2, 3, 4, 5, 6, 7, 9), \emptyset)$ does not fulfill the load condition $(1, 6, 1, 6)$. The decomposed matrix $\mathcal{D}(A)$ is shown in Subfigure 4.7c. One can see that the second block includes seven columns. We observe that the 2-decomposition $\mathcal{D}_2 = ((c, e), (a, d), (b, f); (1, 4, 8), (2, 3, 5, 6, 7, 9), \emptyset)$ of A fulfills the load condition $(1, 6, 1, 6)$. The decomposed matrix $\mathcal{D}_2(A)$ is illustrated in Subfigure 4.7d.

Weighting Schemes

To apply the hypercolumn decomposing algorithm, we have to solve an HVS problem. This is done indirectly by solving an HES problem followed by searching a minimum vertex cover on the obtained edge cut graph. We notice that the objective function value of the obtained k-decomposition and the cardinality of the found vertex cover are equal. Hence, it might be preferable to get an edge cut graph with relatively few edges. In order to obtain such an edge cut graph, one can assign higher weights to those hyperedges that are likely to span more than two partitions when solving the HES problem. Therefore, it could be useful to use a weight function that follows the prop size weighting scheme (4.2). Furthermore, we will make use of weight functions that follow the unary weighting scheme (4.1).

4.4 Models for MinAf

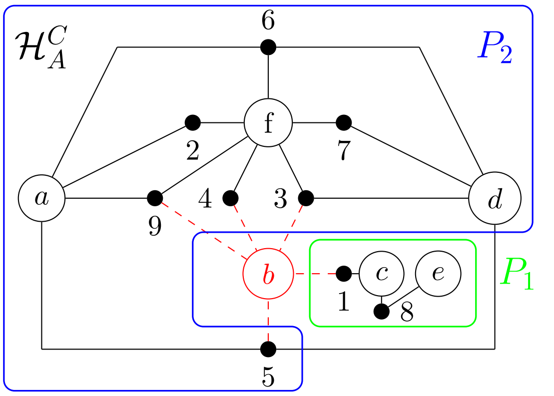
In this section, we introduce two heuristic algorithms for solving the problem $\textsc{MinAf}$, namely the *hypercolrow decomposing algorithm* and the *bipartite decomposing algorithm*.

4.4.1 Hypercolrow Decomposing Algorithm

This subsection deals with the *hypercolrow decomposing algorithm*. At first, we define the *hypercolrow graph* $\mathcal{H}_A^{CR}$ of a matrix A, and describe the main idea of the algorithm which is essentially to solve an HES problem on $\mathcal{H}_A^{CR}$. Secondly, we show a small example to

$$\begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline a & & 1 & & & 1 & 1 & & & 1 \\ b & 1 & & 1 & 1 & 1 & & & & 1 \\ c & 1 & & & & & & & 1 & \\ d & & & 1 & & 1 & 1 & 1 & & \\ e & & & & & & & & 1 & \\ f & & 1 & & & & 1 & 1 & & \end{array}$$

(a) Matrix $A \in \mathbb{R}^{6\times 9}$



(b) Hypercolumn graph $\mathcal{H}_A^C$ of A with solution $P = (P_1, P_2, \{b\})$ of the problem $\mathrm{HVS}(\mathcal{H}_A^C, 2, 2)$

$$\begin{array}{c|ccccccccc} & 1 & 8 & 2 & 3 & 4 & 5 & 6 & 7 & 9 \\ \hline c & 1 & 1 & & & & & & & \\ e & & 1 & & & & & & & \\ a & & & 1 & & & 1 & 1 & & 1 \\ d & & & & 1 & & 1 & 1 & 1 & \\ f & & & 1 & 1 & 1 & & 1 & 1 & 1 \\ b & 1 & & & 1 & 1 & 1 & & & 1 \end{array}$$

(c) Decomposed matrix $\mathcal{D}(A)$ such that $\mathcal{D}$ does not fulfill the load condition

$$\begin{array}{c|ccccccccc} & 1 & 4 & 8 & 2 & 3 & 5 & 6 & 7 & 9 \\ \hline c & 1 & & 1 & & & & & & \\ e & & & 1 & & & & & & \\ a & & & & 1 & & 1 & 1 & & 1 \\ d & & & & & 1 & 1 & 1 & 1 & \\ b & 1 & 1 & & & 1 & 1 & & & 1 \\ f & & 1 & & 1 & 1 & & 1 & 1 & 1 \end{array}$$

(d) $\mathcal{D}_2(A)$ such that $\mathcal{D}_2$ fulfills the load condition

Figure 4.7: Failed run of the hypercolumn decomposing algorithm; the deduced 2-decomposition $\mathcal{D} = ((c,e),(a,d,f),(b);(1,8),(2,3,4,5,6,7,9),\emptyset)$ does not fulfill the load condition $(1,6,1,6)$.

visualize a successful run of the hypercolrow decomposing algorithm. Thirdly, we present the formal algorithm. After indicating a failing example run, we introduce an alternative weighting scheme to solve the corresponding HES problem.

Definition 4.4.1 (Hypercolrow graph of a matrix)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix. We define the *hypercolrow graph* of A to be the hypergraph $\mathcal{H}_A^{CR} = (\mathcal{V}, \mathcal{E})$ with

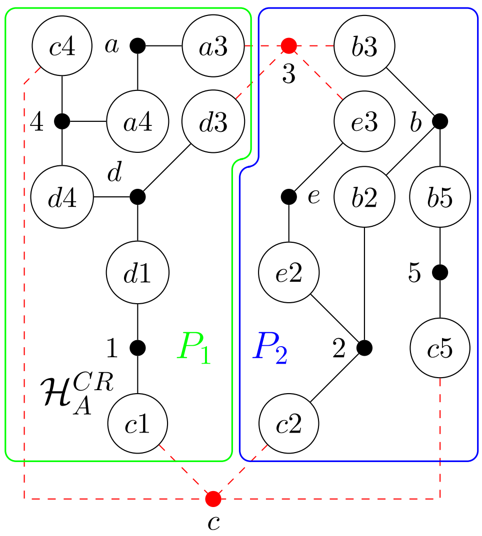
- $\mathcal{V} = \{v_{ij} | i \in [m], j \in m \text{ with } a_{ij} \neq 0\}$ and
- $\mathcal{E} = \mathcal{E}_R \cup \mathcal{E}_C$ with $\mathcal{E}_R := \{e_i^R | i \in [m]\}$ and $\mathcal{E}_C := \{e_j^C | j \in [n]\}$ such that $v_{ij} \in e_i^R$ and $v_{ij} \in e_j^C$ for all $i \in [m]$ and for all $j \in [n]$.

The hypercolrow graph $\mathcal{H}_A^{CR}$ of a matrix A has a vertex for every nonzero entry of A. Each hyperedge of $\mathcal{H}_A^{CR}$ either stands for a row or column. Every hyperedge representing a row r connects exactly those vertices whose corresponding nonzero entries belong to r. Analogously, a hyperedge related to a column c spans exactly those vertices whose corresponding nonzero entries are in c. We are going to solve an HES problem on $\mathcal{H}_A^{CR}$ and want to deduce a k-decomposition $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ from the obtained solution $P = (P_1 \dots, P_k)$ of the HES problem. For $t \in [k]$, part P_t corresponds to the row block part $\mathcal{R}_t$ and the column block part $\mathcal{C}_t$ of $\mathcal{D}$. To be more precise, rows and columns belonging to edges in the edgecut of P become border rows and columns, respectively. Each of the remaining edges spans vertices of just one part of the obtained partition. The corresponding rows and columns are assigned to the according row block part and column block part, respectively.

For the sake of visualization, let us consider a small example displayed in Figure 4.8. We want to solve the problem $\textsc{MinAf}(A, 2, 1, 4, 1, 4)$ for the matrix $A \in \mathbb{R}^{5\times 5}$ illustrated in Subfigure 4.8a. The corresponding hypercolrow graph $\mathcal{H}_A^{CR}$ is displayed in Subfigure 4.8b. In the same subfigure one can find a solution $P = (P_1, P_2)$ with the edge cut $Cut(P) = \{e, 3\}$ for the problem HES($\mathcal{H}_A^{CR}$,2,2). We obtain the 2-decomposition $\mathcal{D} = ((a, d), (b, c), (e); (1, 4), (2, 5), (3))$ from P. The decomposed matrix $\mathcal{D}(A)$ is in 2-arrowhead form and is illustrated in Subfigure 4.8c.

$$
\begin{array}{c}
\begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5
\end{array}\\
\begin{array}{c} a \\ b \\ c \\ d \\ e \end{array}
\begin{pmatrix}
 & & 1 & 1 & \\
 & 1 & 1 & & 1 \\
1 & 1 & & 1 & 1 \\
1 & & 1 & 1 & \\
 & 1 & 1 & &
\end{pmatrix}
\end{array}
$$

(a) Matrix $A \in \mathbb{R}^{5\times 5}$



(b) Hypercolrow graph $\mathcal{H}_A^{CR}$ with obtained solution $P = (P_1, P_2)$ for the HES problem

$$
\begin{array}{c}
\begin{array}{cccccc}
 & 1 & 4 & 2 & 5 & 3
\end{array}\\
\begin{array}{c} a \\ d \\ b \\ e \\ c \end{array}
\begin{pmatrix}
 & 1 & & & 1 \\
1 & 1 & & & 1 \\
 & & 1 & 1 & 1 \\
 & & 1 & & 1 \\
1 & 1 & 1 & 1 &
\end{pmatrix}
\end{array}
$$

(c) Decomposed matrix $\mathcal{D}(A)$ in 2-arrowhead form.

Figure 4.8: Succesful run of hypercolrow decomposing algorithm

The hypercolrow decomposing algorithm is displayed in Algorithm 10. One crucial point is the choice of α before solving the HES problem. We want to choose α in a way such that no solution of HES that would yield a feasible solution of $\textsc{MinAf}$ is pruned. A block containing the maximum number of rows $u^{\mathcal{R}}$ and the maximum number of columns $u^{\mathcal{C}}$ includes maximal $u^{\mathcal{R}} \cdot u^{\mathcal{C}}$ nonzero entries. But a part of a vertex partition that would yield such a block can contain more than $u^{\mathcal{R}} \cdot u^{\mathcal{C}}$ vertices because it might also include vertices that belong to nonzero entries that are part of the border. In fact, it may contain up to $u^{\mathcal{R}} \cdot r^* \cdot u^{\mathcal{C}} \cdot c^*$ vertices with r^* is the maximum number of nonzero entries in a row of A and c^* is the maximum number of nonzero entries in a column of A. Hence, we should set $\alpha := \max\left(\frac{u^{\mathcal{R}} r^* u^{\mathcal{C}} c^* k}{z}, 1\right)$ where z is the number of nonzero entries in A. If $\frac{u^{\mathcal{R}} r^* u^{\mathcal{C}} c^* k}{z} \geq 1$, we obtain $|P_t| \leq \frac{u^{\mathcal{R}} r^* u^{\mathcal{C}} c^* k}{z} \frac{z}{k} = u^{\mathcal{R}} r^* u^{\mathcal{C}} c^*$ for all $t \in [k]$, as we intend.

The above described *hypercolrow decomposing algorithm* is indicated in detail in Algorithm 10. Three methods are used in Algorithm 10. It is clear from Definition 4.3.1, how the first method is `CreateHypercolrowGraph()` could be implemented. The second method is `SolveHES()`. It is solved by an HES solver that is treated as 'black box'. The method `TransPartToDecompHCR()` is described in detail in Algorithm 11.

Algorithm 10: HypercolrowDecomposingAlgorithm

input : $A \in \mathbb{R}^{m \times n}$ a matrix with z nonzero entries, the maximum number of nonzero entries in a row of A is r^* and the maximum number of nonzero entries in a column of A is c^*, $k \in \mathbb{N}$ an integer, $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$.

output: A k-decomposition $\mathcal{D}$ that fulfills the block condition and the load condition$(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ or a statement that no solution could be found.

1 $\mathcal{H}_A^{CR} \leftarrow$ `CreateHypercolrowGraph(`A`)`;
2 $\alpha \leftarrow \max\left(\frac{u^{\mathcal{R}} r^* u^{\mathcal{C}} c^* k}{z}, 1\right)$;
3 $P = (P_1, \ldots, P_k) \leftarrow$ `SolveHES(`$\mathcal{H}_A^{CR}$`,`k`,`α`)`;
4 **if** P *is not feasible for* HES*(*$\mathcal{H}_A^{CR}$*,*k*,*α*)* **then**
5 | *return statement that no feasible solution could be found and quit*;
6 **end**
7 $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B) \leftarrow$ `TransPartToDecompHCR(`$\mathcal{H}_A^{CR}$`,`k`,`P`)`;
8 **if** $\mathcal{D}$ *fulfills the load condition* $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ **then**
9 | **return** $\mathcal{D}$;
10 **end**
11 **else**
12 | *return statement that no feasible solution could be found*;
13 **end**

The next lemma states that Algorithm 10 is heuristically correct in terms of Remark 12:

Lemma 4.4.2 (Heuristic correctness of the hypercolrow decomp. algorithm)
Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers.
If Algorithm 10 ends without sending a message for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, then it returns a k-decomposition that is feasible for MinAf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Proof: Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers.

We are going to prove the following statement: If line 7 of Algorithm 10 is reached, then the tuple $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ in line 7 is a k-decomposition that fulfills

Algorithm 11: TransformPartToDecompHCR

input : $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ a hypergraph, $k \in \mathbb{N}$ an integer, and $P = (P_1, \dots, P_k)$ a solution of HES problem on $\mathcal{H}$.

output: A k-decomposition $\mathcal{D}$ of A that fulfills the block condition.

$\mathcal{R}_B \leftarrow [m]$;
$\mathcal{C}_B \leftarrow [n]$;
for $b \in [k]$ **do**
  $\mathcal{R}_b \leftarrow \{i \in [m] | v \in P_b \text{ for all } v \in e_i^R\}$;
  $\mathcal{C}_b \leftarrow \{j \in [n] | v \in P_b \text{ for all } v \in e_j^C\}$;
  $\mathcal{R}_B \leftarrow \mathcal{R}_B \setminus \mathcal{R}_b$;
  $\mathcal{C}_B \leftarrow \mathcal{C}_B \setminus \mathcal{C}_b$;
end
return $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$;

the block condition. Thus, if line 9 is reached, the k-decomposition $\mathcal{D}$ additionally fulfills the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, and hence is feasible for MinAf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Let Algorithm 10 ends without sending a message for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, then it reaches line 7. Consider the tuple $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ which has been returned by the method `TransPartToDecompHCR()`. Let $\mathcal{H}_A^{CR}$ be the hypercolrow graph of A and let α have the same value as in line 2. Furthermore, let $P = (P_1, \dots, P_k)$ be the solution of HES(A, k, α) that was returned in line 3. Similarly to the proof of Lemma 4.3.3, we will show that $(\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B)$ is a partition of the rows of A and that $(\mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ is a partition of the columns of A. Afterwards, we prove that $\mathcal{D}$ fulfills the block condition. It holds in line 7 of Algorithm 10:

- $\mathcal{R}_b = \{i \in [m] | v \in P_b \text{ for all } v \in e_i^R\}$ for all $b \in [k]$,
- $\mathcal{C}_b = \{j \in [n] | v \in P_b \text{ for all } v \in e_j^C\}$ for all $b \in [k]$,
- $\mathcal{R}_B = [m] \setminus \bigcup_{k \in [b]} \mathcal{R}_b$ and
- $\mathcal{C}_B = [n] \setminus \bigcup_{k \in [b]} \mathcal{C}_b$.

Therefore, it is

$$\bigcup_{b \in [k]} \mathcal{R}_b \cup \mathcal{R}_B = \bigcup_{b \in [k]} \mathcal{R}_b \cup \left([m] \setminus \bigcup_{b \in [k]} \mathcal{R}_b \right) = [m]$$

and

$$\bigcup_{b \in [k]} \mathcal{C}_b \cup \mathcal{C}_B = \bigcup_{b \in [k]} \mathcal{C}_b \cup \left([n] \setminus \bigcup_{b \in [k]} \mathcal{C}_b \right) = [n].$$

Moreover, for all $b_1, b_2 \in [k]$ with $b_1 \neq b_2$, it is $\mathcal{R}_{b_1} \cap \mathcal{R}_{b_2} = \emptyset$ and $\mathcal{C}_{b_1} \cap \mathcal{C}_{b_2} = \emptyset$. In order to see that fact, suppose there are arbitrary $b_1, b_2 \in [k]$ with $\mathcal{R}_{b_1} \cap \mathcal{R}_{b_2} \neq \emptyset$ or $\mathcal{C}_{b_1} \cap \mathcal{C}_{b_2} \neq \emptyset$. If we have $\mathcal{R}_{b_1} \cap \mathcal{R}_{b_2} \neq \emptyset$, then there is an $i \in [m]$ with $i \in \mathcal{R}_{b_1}$ and $i \in \mathcal{R}_{b_2}$. Hence, by definition of $\mathcal{R}_{b_1}$ and $\mathcal{R}_{b_2}$, for all $v \in e_i^R$ is $v \in P_{b_1}$ and $v \in P_{b_2}$. Since $(P_1, \ldots, P_k)$ is a partition of the vertices of $\mathcal{H}_A^{CR}$, we obtain $b_1 = b_2$. Analogously, if $\mathcal{C}_{b_1} \cap \mathcal{C}_{b_2} \neq \emptyset$, then there is an $j \in [n]$ with $j \in \mathcal{C}_{b_1}$ and $j \in \mathcal{C}_{b_2}$. Thus, by definition of $\mathcal{C}_{b_1}$ and $\mathcal{C}_{b_2}$, for all $v \in e_j^C$ is $v \in P_{b_1}$ and $v \in P_{b_2}$. Since $(P_1, \ldots, P_k)$ is a partition of the vertices of $\mathcal{H}_A^{CR}$, we obtain $b_1 = b_2$. Therefore, $(\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A, and $(\mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ is a weak partition of the columns of A. Hence, $\mathcal{D}$ is a k-decomposition of A.

It remains to show that $\mathcal{D}$ fulfills the block condition. In order to see that, consider $i \in \mathcal{R}_{b_1}$ and $j \in \mathcal{C}_{b_2}$ with $a_{ij} \neq 0$ for some $b_1, b_2 \in [k]$. Hence, by definition, for all $v \in e_i^R$ is $v \in P_{b_1}$ and for all $v \in e_j^C$ we have $v \in P_{b_2}$. Since $a_{ij} \neq 0$, there exists a vertex v_{ij} of $\mathcal{H}_A^{CR}$ with $v_{ij} \in e_i^R$ and $v_{ij} \in e_j^C$. Therefore, it holds that $v_{ij} \in P_{b_1}$ and $v_{ij} \in P_{b_2}$. Because $(P_1, \ldots, P_k)$ is a partition of the vertices of $\mathcal{H}_A^{CR}$, we obtain $b_1 = b_2$; hence, $\mathcal{D}$ fulfills the block condition. Therefore, if line 9 is reached, $\mathcal{D}$ is feasible for $\textsc{MinAf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

□

In Figure 4.9 a failed run of the hypercolrow algorithm is presented. We want to solve the problem $\textsc{MinAf}(A, 2, 1, 6, 1, 6)$ for the matrix A presented in Subfigure 4.9a. The corresponding hypercolrow graph $\mathcal{H}_A^{CR}$ of A is displayed in Subfigure 4.9c. Furthermore, a solution $P = (P_1, P_2)$ for the corresponding HES prolem is displayed in Subfigure 4.9c that yield the 2-decomposition $\mathcal{D} = ((c, g), (a, b, d, f, h, i, j), (e); (4, 7), (1, 2, 3, 6, 8, 9), (5))$ of A that does not fulfill the load condition. Nevertheless, the permuted matrix $\mathcal{D}(A)$ is displayed in Subfigure 4.9b. One can see that the first block contains only two rows while the second block includes seven rows.

Weighting Schemes

As a variation, we want to solve the HES problem not only according to the unary weighting scheme (4.1), but also according to the prop size weighting scheme (4.2).

4.4.2 Bipartite Decomposing Algorithm

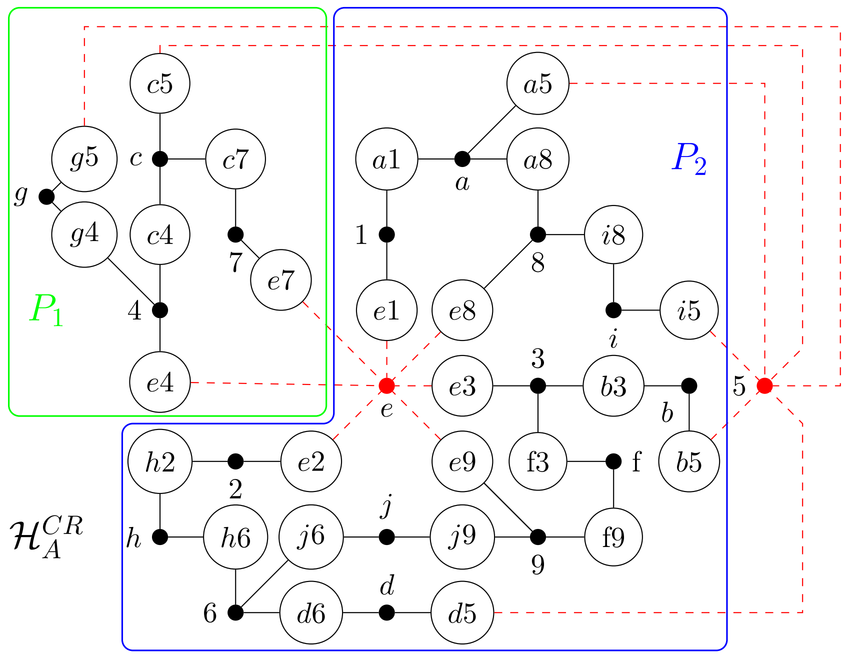
In this subsection, we introduce the *bipartite decomposing algorithm.* At first, we define the *bipartite graph* G_A^B of a matrix A and show the main idea of the algorithm. Afterwards, we show a small example to visualize a successful run of it . Thirdly, we give a formal description of the algorithm. Finally, as for the proceeding algorithms, we will indicate a failed run of algorithm.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \end{array}
\begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\
\begin{pmatrix}
1 & & & & 1 & & & 1 & \\
& & 1 & & 1 & & & & \\
& & & 1 & 1 & & 1 & & \\
& & & & 1 & 1 & & & \\
1 & 1 & 1 & & & & 1 & 1 & 1 \\
& & 1 & & & & & & 1 \\
& & & 1 & 1 & & & & \\
& 1 & & & & 1 & & & \\
& & & & 1 & & & 1 & \\
& & & & & 1 & & & 1
\end{pmatrix} \end{array}$$

(a) Matrix $A \in \mathbb{R}^{10\times 9}$ for $\textsc{MinAf}(A, 2, 1, 6, 1, 6)$

$$\begin{array}{c} \\ c \\ g \\ a \\ b \\ d \\ f \\ h \\ i \\ j \\ e \end{array}
\begin{array}{c} \begin{array}{ccccccccc} 4 & 7 & 1 & 2 & 3 & 6 & 8 & 9 & 5 \end{array} \\
\begin{pmatrix}
1 & 1 & & & & & & & 1 \\
1 & & & & & & & & 1 \\
& & 1 & & & & 1 & & 1 \\
& & & & 1 & & & & 1 \\
& & & & & 1 & & & 1 \\
& & & & 1 & & & 1 & 1 \\
& & & 1 & & 1 & & & \\
& & & & & & 1 & & 1 \\
& & & & & 1 & & 1 & \\
1 & 1 & 1 & 1 & 1 & & 1 & 1 &
\end{pmatrix} \end{array}$$

(b) Decomposed matrix $\mathcal{D}(A)$



(c) Hypercolrow graph $\mathcal{H}_A^{CR}$ with solution $P = (P_1, P_2)$ for HES that yields the 2-decomposition $\mathcal{D} = ((c, g), (a, b, d, f, h, i, j), (e); (4, 7), (1, 2, 3, 6, 8, 9), (5))$ that does not fulfill the load condition $(1, 6, 1, 6)$

Figure 4.9: Failed run of hypercolrow decomposing algorithm

Definition 4.4.3 (Bipartite graph of a matrix)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix. We define the *bipartite graph* of A as the undirected graph $G_A^B = (V, E)$, with

- $V := V_R \cup V_C$ with $V_R := \{v_i^R | i \in [m]\}$ and $V_C := \{v_j^C | j \in [n]\}$, and
- $E = \{(v_i^R, v_j^C) \in V_R \times V_C | i \in [m], j \in [n] \text{ with } a_{ij} \neq 0\}$.

Observation 4.4.4
The bipartite graph of a matrix $A \in \mathbb{R}$ is bipartite

The main idea of the algorithm is to solve an HVS problem on the bipartite graph of the matrix to decompose. The obtained weak partition of the vertices $P = (P_1, \ldots .P_k, S)$ is transformed into a k-decomposition that will turn out to fulfill the block condition. The transformation is easy: The rows and columns that correspond to vertices in a part P_b (of the obtained partition P) for some $b \in [k]$ are assigned to row block part $\mathcal{R}_b$ and column block part $\mathcal{C}_b$, respectively. The remaining rows and columns are assigned to the border row part $\mathcal{R}_B$ and border column part $\mathcal{C}_B$, respectively. If the load condition is fulfilled, the run of the heuristic was successful.

A successful run is illustrated in Figure 4.10. We want to solve MinAf$(A, 2, 1, 3, 1, 3)$ for the displayed matrix $A \in \mathbb{R}^{5\times 5}$. The bipartite graph G_A^B of A is shown in Subfigure 4.10a. The solution $P = (P_1, P_2, \{a, 3\})$ of the HVS$(G_A^B, 2, 2)$ problem is visualized in Figure 4.10c. We obtain the 2-decomposition $\mathcal{D} = ((c, e), (b, d), (a); (2, 4), (1, 5), (3))$ from P. $\mathcal{D}$ fulfills the load condition $(1, 3, 1, 3)$ and the block condition. The decomposed matrix $\mathcal{D}(A)$ is in 2-arrowhead form and is displayed in Subfigure 4.10d.

In order to solve the HVS problem on the bipartite graph of A, the choice of α is a crucial point from a theoretical point of view. Since the number of nodes in a part P_b and the total number of rows and columns in block b are the same, we want every vertex part to have at most $u^{\mathcal{R}} + u^{\mathcal{C}}$ vertices. We set $\alpha := \max(\frac{k(u^{\mathcal{R}}+u^{\mathcal{C}})}{m+n}, 1)$ and obtain

$$|P_t| \leq \alpha \frac{m+n}{k} = \frac{k(u^{\mathcal{R}} + u^{\mathcal{C}})}{m+n} \frac{m+n}{k} = u^{\mathcal{R}} + u^{\mathcal{C}}, \text{ if } \frac{k(u^{\mathcal{R}} + u^{\mathcal{C}})}{m+n} \geq 1$$

for all $t \in [k]$.

The formal procedure of the bipartite decomposing algorithm is displayed in detail in Algorithm 12. It makes use of three methods. Method `CreateBipartiteGraph()` is clear from Definition 4.4.3. The second method, namely `SolveHVS()`, can be implemented by Algorithm 4 described in Subsection 4.1.1. Finally, an implementation of method `TransPartToDecompBip()` is indicated in Algorithm 13.

The next lemma states that Algorithm 12 is heuristically correct, in terms of Remark 12.

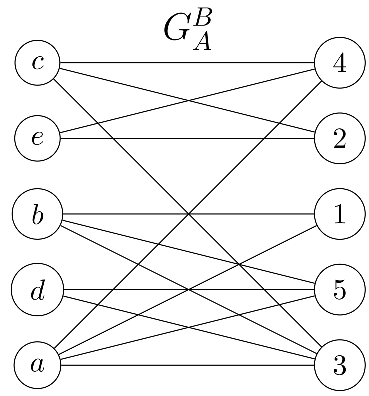
Lemma 4.4.5 (Heuristic correctness of the bipartite decomp. algorithm)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers.
If Algorithm 12 ends without sending a message for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, then it returns a k-decomposition that is feasible for MinAf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Proof: Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$ be an integer. Furthermore, let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be integers.

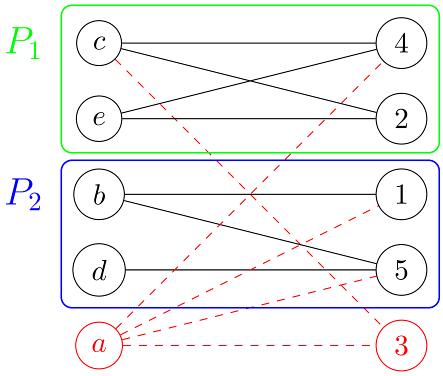
Let Algorithm 12 ends without sending a message for the input $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, then it reaches line 7. In the following, we prove the statement: If line 7 of Algorithm 12 is reached, then the tuple $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ is a k-decomposition that

$$
\begin{array}{c|c}
 & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\
\begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} &
\begin{pmatrix}
1 & & 1 & 1 & 1 \\
1 & & 1 & & 1 \\
 & 1 & 1 & 1 & \\
 & & 1 & & 1 \\
 & 1 & & 1 &
\end{pmatrix}
\end{array}
$$

(a) Matrix $A \in \mathbb{R}^{5\times5}$



(b) Bipartite graph G_A^B of A



(c) Bipartite graph of A with obtained solution $P = (P_1, P_2, \{a, 3\})$ for the HVS problem

$$
\begin{array}{c|c}
 & \begin{matrix} 2 & 4 & 1 & 5 & 3 \end{matrix} \\
\begin{matrix} c \\ e \\ b \\ d \\ a \end{matrix} &
\begin{pmatrix}
1 & 1 & & & 1 \\
1 & 1 & & & \\
 & & 1 & 1 & 1 \\
 & & & 1 & 1 \\
 & 1 & 1 & 1 & 1
\end{pmatrix}
\end{array}
$$

(d) Decomposed matrix $\mathcal{D}(A)$ in 2-arrowhead form

Figure 4.10: Succesful run of bipartite decomposing algorithm

fulfills the block condition. Thus, if line 9 is reached, the k-decomposition $\mathcal{D}$ also fulfills the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, and hence is feasible for $\textsc{MinAf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

Consider the tuple $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ which has been returned by the method `TransPartToDecompBip()`. Let G_A^B be the bipartite graph of A and set $\alpha := \max(\frac{k(u^{\mathcal{R}}+u^{\mathcal{C}})}{m+n}, 1)$. Moreover, let $P = (P_1, \ldots, P_k, S)$ be the solution of $\text{HVS}(A, k, \alpha)$ that was returned in line 3 by method `SolveHVS()`. Analogously to the proof of Lemma 4.4.2, we will show that $(\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A, and that $(\mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ is a weak partition of the columns of A. Eventually, we prove that $\mathcal{D}$ fulfills the block condition. In line 7 of Algorithm 12 we have:

- $\mathcal{R}_b = \{i \in [m] | v_i^R \in P_b\}$ for all $b \in [k]$,
- $\mathcal{C}_b = \{j \in [n] | v_j^C \in P_b\}$ for all $b \in [k]$,

Algorithm 12: BipartiteDecomposingAlgorithm

input : $A \in \mathbb{R}^{m\times n}$ a matrix, $k \in \mathbb{N}$ an integer, $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ and $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$.
output: A k-decomposition $\mathcal{D}$ for A that fulfills the block condition and the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ or a statement that no solution could be found.

```
G_A^B ← CreateBipartiteGraph(A);
α ← max(k(u^R + u^C)/(m+n), 1);
P = (P_1, ..., P_k, S) ← SolveHVS(G_A^B, k, α);
if P is not feasible for HVS(G_A^B, k, α) then
    return statement that no feasible solution could be found and quit;
end
D = (R_1, ..., R_k, R_B; C_1, ..., C_k, C_B) ← TransPartToDecompBip(G_A^B, k, P);
if D fulfills the load condition (ℓ^R, u^R, ℓ^C, u^C) then
    return D;
end
else
    return statement that no feasible solution could be found;
end
```

Algorithm 13: TransPartToDecompBip

input : $G = (V, E)$ an undirected graph, $k \in \mathbb{N}$ an integer, and $P = (P_1, \dots, P_k, S)$ a solution of a HVS problem on G .
output: A k-decomposition $\mathcal{D}$ of A that fulfills the block condition.

```
R_B ← {i ∈ [m] | v_i^R ∈ S};
C_B ← {j ∈ [n] | v_j^C ∈ S};
for b ∈ [k] do
    R_b ← {i ∈ [m] | v_i^R ∈ P_b};
    C_b ← {j ∈ [n] | v_j^C ∈ P_b};
end
return D = (R_1, ..., R_k, R_B; C_1, ..., C_k, C_B);
```

- $\mathcal{R}_B = \{i \in [m] | v_j^C \in S\}$ and
- $\mathcal{C}_B = \{j \in [n] | v_j^C \in S\}$.

Consider an arbitrary row $i \in [m]$ of A. Since $(P_1 \dots, P_k, S)$ is a partition of the rows, there is exactly one set $Q \in \{P_1, \dots, P_k, S\}$ with $v_i^R \in Q$. Due to the definition of $\mathcal{R}_B$ and $\mathcal{R}_b$ for $b \in [k]$, there is exactly one set $Q' \in \{\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B\}$ with $i \in Q'$. Hence, $(\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B)$ is a weak partition of the rows of A. Analogously, for an arbitrary column $j \in [n]$ of A, there is exactly one set $W \in \{P_1, \dots, P_k, S\}$ with $j \in W$. Owing to the definition of $\mathcal{C}_B$ and $\mathcal{C}_b$ for $b \in [k]$, there is exactly one set $W' \in \{\mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B\}$

with $j \in W'$. Therefore, $(\mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ is a weak partition of the columns of A. Hence, $\mathcal{D} = (\mathcal{R}_1, \dots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \dots, \mathcal{C}_k, \mathcal{C}_B)$ is a k-decomposition for A.

Now we show that $\mathcal{D}$ fulfills the block condition. Consider $i \in \mathcal{R}_{b_1}$ and $j \in \mathcal{C}_{b_2}$ with $a_{ij} \neq 0$ for some $b_1, b_2 \in [k]$. Then we have $v_i^R \in P_{b_1}$ and $v_j^C \in P_{b_2}$. Since $a_{ij} \neq 0$, there is an edge e of G_A^B with $v_i^R \in e$ and $v_j^C \in e$. Because $(P_1, \dots, P_k, S)$ is a k-way α-hypervertex separator and $v_i^R \in P_{b_1}$, we know that $v_j^C \in P_{b_1}$. Since $(P_1, \dots, P_k, S)$ is a weak partition of the vertices of G_A^B, we obtain $b_1 = b_2$. Hence, the block condition is fulfilled.

□

It is worth pointing out that even if the solution of the HVS problem is balanced, the obtained k-decomposition could fail the load condition. An example is shown in Figure 4.11. Consider the problem $\textsc{MinAf}(A, 2, 1, 6, 1, 6)$ for the matrix $A \in \mathbb{R}^{11 \times 11}$ displayed in Subfigure 4.11a. The bipartite graph G_A^B of A is visualized in Subfigure 4.11c. Furthermore, this subfigure illustrates the feasible solution $P = (P_1, P_2, \{d, 8\})$ of $\mathrm{HVS}(G_A^B, 2, 2)$. We can obtain the 2-decomposition $\mathcal{D}$ from P with

$$\mathcal{D} = ((a, b, c, f, g, i, j), (e, h, k), (d); (2, 5, 9), (1, 3, 4, 6, 7, 10, 11), (8))$$

that does not fulfill the load condition $(1, 6, 1, 6)$. The decomposed matrix $\mathcal{D}(A)$ is shown in Subfigure 4.11b.

Weighting Schemes

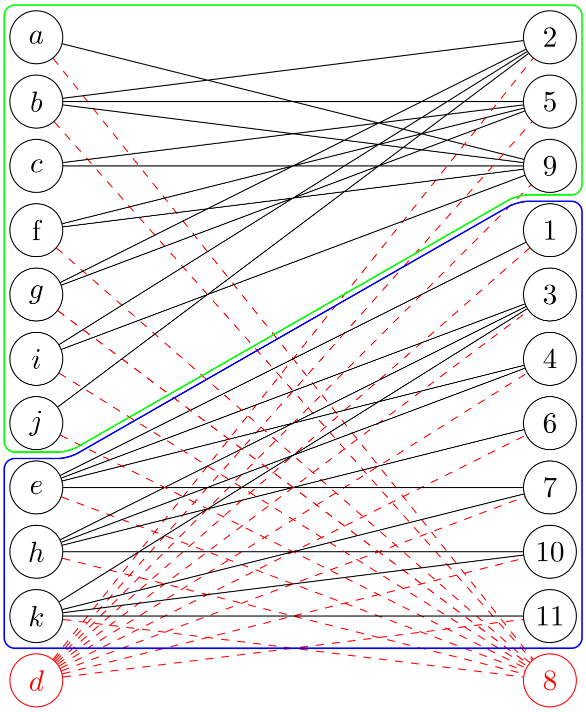
We solve the HVS indirectly as described in Section 4.1.1 by solving an HES problem. Thereby, we make use of the unary weighting scheme (4.1) and the aprop degree weighting scheme (4.3).

$$
\begin{array}{c} \\ a\\ b\\ c\\ d\\ e\\ f\\ g\\ h\\ i\\ j\\ k \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} 1&2&3&4&5&6&7&8&9&10&11 \end{array}\\
\left(\begin{array}{ccccccccccc}
 & & & & & & &1&1& & \\
 &1& & &1& & &1&1& & \\
 & & & &1& & & &1& & \\
1&1&1&1&1&1&1& &1&1&1\\
1& &1&1& & &1&1& & & \\
 & & & &1& & &1&1& & \\
 &1& & &1& & &1& & & \\
 & &1&1& &1& &1& &1& \\
 &1& & & & & &1&1& & \\
 &1& & & & & &1& & & \\
 & &1& & & &1&1& &1&1
\end{array}\right)
\end{array}
$$

(a) Matrix $A \in \mathbb{R}^{11\times 11}$

$$
\begin{array}{c} \\ a\\ b\\ c\\ f\\ g\\ i\\ j\\ e\\ h\\ k\\ d \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} 2&5&9&1&3&4&6&7&10&11&8 \end{array}\\
\left(\begin{array}{ccccccccccc}
 & &1& & & & & & & &1\\
1&1&1& & & & & & & &1\\
 &1&1& & & & & & & & \\
 &1&1& & & & & & & &1\\
1&1& & & & & & & & &1\\
1& &1& & & & & & & &1\\
1& & & & & & & & & &1\\
 & & &1&1&1& &1& & &1\\
 & & & &1&1&1& &1& &1\\
 & & & &1& & &1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&
\end{array}\right)
\end{array}
$$

(b) Decomposed matrix $\mathcal{D}(A)$ with $\mathcal{D}$ does not fulfill the load condition $(1, 6, 1, 6)$



(c) Bipartite graph G_A^B with solution $P = (P_1, P_2)$ for the HVS problem

Figure 4.11: Failed run of the bipartite decomposing algorithm; although the partition $P = (P_1, P_2)$ is perfectly balanced, the deduced decomposition fails the load condition

5 Exact Decomposing Methods

This chapter consists of three sections. At first, we introduce the model used by Borndörfer et al. [12] to solve MinBf for some load conditions by a so-called branch-and-cut algorithm. However, we may obtain solutions with empty blocks, if the upper row load capacity is not set properly. Secondly, we present an integer program to solve the problems MinAf and MinBf. Unfortunately, it will turn out to be rather weak. Moreover, we introduce cuts to improve its performance. Finally, we suggest a column generation approach to solve the problem MinAf.

5.1 Borndörfer's Approach to MinBf

Borndörfer et al. suggested a branch-and-cut algorithm[12] that copes the MDP introduced in Section 2.5.1. It is about assigning as many rows of A as possible to β blocks such that the following three conditions hold:

1. Each row is assigned to at most one block.
2. There are at most κ rows assigned to each block.
3. There do not exist two rows in different blocks that have a nonzero entry in the same column.

Obviously, it is similar to the problem $\textsc{MinBf}(A, k, 0, u^{\mathcal{R}}, 0, n)$ for a matrix $A \in \mathbb{R}^{m \times n}$, and the integers $k, u^{\mathcal{R}} \in \mathbb{N}$. Borndörfer suggests a branch-and-cut algorithm that is based on the integer program IP_B described below.

It contains a binary variable y_{ti} for every pair (t, i) where $t \in [k]$ is a block and $i \in [m]$ is a row. The variable y_{ti} has value one, if and only if, row i is assigned to block t.

$$
(IP_B) \qquad
\begin{array}{lll}
\text{Maximize} & \displaystyle\sum_{i=1}^{m}\sum_{t=1}^{k} y_{ti} & \\
\text{subject to} & \displaystyle\sum_{t=1}^{k} y_{ti} \le 1, & \text{for } i \in [m]; \qquad (B1) \\
& \displaystyle\sum_{i=1}^{m} y_{ti} \le u^{\mathcal{R}}, & \text{for } t \in [k]; \qquad (B2) \\
& y_{ti} + y_{t'j} \le 1, & \text{for } t, t' \in [k], t \neq t' \text{ and} \qquad (B3) \\
& & \text{for } i, j \in [m] \text{ such that} \\
& & \mathrm{a}_{i\ell} \neq 0 \neq a_{j\ell} \text{ for some } \ell \in [n]; \\
& y_{ti} \in \{0, 1\}, & \text{for } t \in [k], i \in [m]. \qquad (B4)
\end{array}
$$

It is a simple matter to obtain a β-decomposition of A from a solution of $\mathrm{MDP}(A, \beta, \kappa)$. However, it is not our purpose to study IP_B in detail. For a fuller treatment of IP_B, we refer the reader to [12].

5.2 Assignment Approach for MinAf

The following integer program is a straight forward assignment model denoted by IP_A. It is defined for the parameters $(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ where $A \in \mathbb{R}^{m\times n}$ is a matrix, $k, u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ are positive integers, and $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ are nonnegative integers. For simplicity, we write IP_A if the parameters are clear, otherwise we write $IP_A(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$.

At first, we present $IP_A(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Afterwards, we describe the variables of the integer program and how an assignment of them can be transformed to a solution $\mathcal{D}$ of MinAf and vice versa. Thirdly, we describe the constraints of IP_A. We will see easily that the assignment of the variables is feasible for IP_A if and only if the corresponding $\mathcal{D}$ is feasible for MinAf. Next, it is shown how the model can be adapted to solve MinBf. Afterwards, we will see that the LP-relaxiation of IP_A is rather weak by giving a fractional solution that has an objective function value of zero. Finally, we give some constraints that should speed up the solving process without strengthening the LP-relaxiation.

$$
IP_A \qquad
\begin{array}{lllr}
\text{Minimize} & \displaystyle\sum_{i=1}^{m} x_i^R \;+\; \sum_{j=1}^{n} x_j^C & & \\
\text{subject to} & \displaystyle\sum_{t=1}^{k} y_{ti}^R + x_i^R = 1, & \text{for } i \in [m]; & (A1R) \\
& \displaystyle\sum_{t=1}^{k} y_{tj}^C + x_j^C = 1, & \text{for } j \in [n]; & (A1C) \\
& \displaystyle\sum_{i=1}^{m} y_{ti}^R \geq \ell^{\mathcal{R}}, & \text{for } t \in [k]; & (A2R\ell) \\
& \displaystyle\sum_{j=1}^{n} y_{tj}^C \geq \ell^{\mathcal{C}}, & \text{for } t \in [k]; & (A2C\ell) \\
& \displaystyle\sum_{i=1}^{m} y_{ti}^R \leq u^{\mathcal{R}}, & \text{for } t \in [k]; & (A2Ru) \\
& \displaystyle\sum_{j=1}^{n} y_{tj}^C \leq u^{\mathcal{C}}, & \text{for } t \in [k]; & (A2Cu) \\
& y_{tj}^C - y_{ti}^R - x_i^R \leq 0, & \text{for } t \in [k],\ i \in [m], & (A3C) \\
& & j \in [n] \text{ with } a_{ij} \neq 0; & \\
& y_{ti}^R, y_{tj}^C, x_i^R, x_j^C \in \{0,1\}, & \text{for } t \in [k],\ i \in [m],\ j \in [n]. & (A4)
\end{array}
$$

We introduce a binary variable y^R_{ti} for every pair (t,i), where $t \in [k]$ is a block and $i \in [m]$ is a row. It attains a value of one if and only if row i is assigned to block t (i.e. $i \in \mathcal{R}_t$). Moreover, a binary variable x^R_i is defined for every row $i \in [m]$ that takes a value of one if and only if row i is assigned to the row border (i.e. $i \in \mathcal{R}_B$). Similarly, we introduce a binary variable y^C_{ti} for every pair (t,j), with $t \in [k]$ is a block and $j \in [n]$ is a column. y^C_{ti} has value one if and only if column j is assigned to block t (i.e. $j \in \mathcal{C}_t$). Furthermore, there is a binary variable x^C_j for every column $j \in [n]$ that takes a value of one if and only if j is assigned to the column border (i. e. $j \in \mathcal{C}_B$). For abbreviation, let $z(v)$ denote the value of variable v for the assignment z. This way, we defined how an assignment of the variables and a tuple $\mathcal{D} = (\mathcal{R}_1,\dots,\mathcal{R}_k,\mathcal{R}_B;\mathcal{C}_1,\dots,\mathcal{C}_k,\mathcal{C}_B)$ can be obtained from each other. For an assignment z of the variables of IP_A and a tuple $\mathcal{D} = (\mathcal{R}_1,\dots,\mathcal{R}_k,\mathcal{R}_B;\mathcal{C}_1,\dots,\mathcal{C}_k,\mathcal{C}_B)$, we write $z \simeq \mathcal{D}$, if and only if they can be obtained from each other.

The constraints $(A1R)$ and $(A1C)$ ensure that every row and every column is assigned either to the respective border or to exactly one block. Therefore, it is evident that $(\mathcal{R}_1,...,\mathcal{R}_k,\mathcal{R}_B)$ is a weak partition of the rows and $(\mathcal{C}_1,...,\mathcal{C}_k,\mathcal{C}_B)$ is a weak partition of the columns of A. Hence, $\mathcal{D}$ is a k-decomposition. The constraints $(A2R\ell)$, $(A2C\ell)$, $(A2Ru)$ and $(A2Cu)$ are respected if and only if $\mathcal{D}$ fulfills the load condition $(\ell^{\mathcal{R}},u^{\mathcal{R}},\ell^{\mathcal{C}},u^{\mathcal{C}})$. The constraints (A3C) ensure that a column c can be assigned to a block t only if every row that has a nonzero entry in c is assigned to block t or to the row border. The following theorem provides a rigorous formulation of the fact that one can use IP_A to solve the problem MinAf.

Theorem 5.2.1
Let $A \in \mathbb{R}^{m\times n}$ be a matrix, let $k,u^{\mathcal{R}},u^{\mathcal{C}} \in \mathbb{N}$ be positive integers, and let $\ell^{\mathcal{R}},\ell^{\mathcal{C}} \in \mathbb{N}_0$ be nonnegative integers. Let $\mathcal{D} = (\mathcal{R}_1,\dots,\mathcal{R}_k,\mathcal{R}_B;\mathcal{C}_1,\dots,\mathcal{C}_k,\mathcal{C}_B)$ be a k-decomposition of A and z be an assignment of the variables of $IP_A(A,k,\ell^{\mathcal{R}},u^{\mathcal{R}},\ell^{\mathcal{C}},u^{\mathcal{C}})$ such that $z \simeq \mathcal{D}$. Then the following holds: The assignment z is feasible for $IP_A(A,k,\ell^{\mathcal{R}},u^{\mathcal{R}},\ell^{\mathcal{C}},u^{\mathcal{C}})$ if and only if $\mathcal{D}$ is feasible for MinAf$(A,k,\ell^{\mathcal{R}},u^{\mathcal{R}},\ell^{\mathcal{C}},u^{\mathcal{C}})$. *Furthermore, if z and $\mathcal{D}$ are feasible for the corresponding problem, then the respective objective function values are equal.*

Proof: Let $A,k,\ell^{\mathcal{R}},\ell^{\mathcal{C}},u^{\mathcal{R}},u^{\mathcal{C}},z$ and $\mathcal{D}$ be as defined in Theorem 5.2.1. Throughout the proof, the variables of IP_A are assigned according to z.

Let us first prove that feasibility of z implies feasibility of $\mathcal{D}$. Suppose z is feasible for $IP_A(A,k,\ell^{\mathcal{R}},u^{\mathcal{R}},\ell^{\mathcal{C}},u^{\mathcal{C}})$. Consider an arbitrary row $i \in [m]$ and an arbitrary column $j \in [n]$. Since all variables of IP_A are binary and the constraints $(A1R)$ are respected, we know that exactly one of the variables $y^R_{1i},\dots,y^R_{ki},x^R_i$ reaches value one. Hence, the tuple $(\mathcal{R}_1,...,\mathcal{R}_k,\mathcal{R}_B)$ is a weak partition of the rows of A. Similarly, exactly one of the variables $y^C_{1j},\dots,y^C_{kj},x^C_j$ takes value one since the constraints $(A1C)$ are not violated. Therefore, the tuple $(\mathcal{C}_1,...,\mathcal{C}_k,\mathcal{C}_B)$ is a weak partition of the columns of A. Therefore, $\mathcal{D} = (\mathcal{R}_1,\dots,\mathcal{R}_k,\mathcal{R}_B;\mathcal{C}_1,\dots,\mathcal{C}_k,\mathcal{C}_B)$ is a k-decomposition of A. According to the constraints $(A2R\ell)$, $(A2C\ell)$, $(A2Ru)$ and $(A2Cu)$, $\mathcal{D}$ fulfills the load condition $(\ell^{\mathcal{R}},u^{\mathcal{R}},\ell^{\mathcal{C}},u^{\mathcal{C}})$. Consider a row $i \in [m]$ and a column $j \in [n]$ with $a_{ij} \neq 0$, and $i \in \mathcal{R}_t$ and $j \in \mathcal{C}_{t'}$ for some $t,t' \in [k]$. We have $y^R_{ti} = 1$ and $y^C_{t'j} = 1$ which implies $x^R_i = 0$ and $y^R_{t^*i} = 0$ for $t^* \in [k] \setminus \{t\}$. Since the constraint $(A3C)$ for (t',i,j) is fulfilled, we conclude that $y^C_{t'j} - y^R_{t'i} - x^R_i \leq 0$, hence that $1 - y^R_{t'i} \leq 0$, and finally that $y^R_{t'i} = 1$. This clearly forces $t = t'$. It follows that the block condition is fulfilled.

Now we show that feasibility of $\mathcal{D}$ yields feasibility of z. The integrality conditions ($A4$) are fulfilled which is clear from $z \simeq \mathcal{D}$. Consider a row $i \in [m]$ and a column $j \in [n]$. Since $\mathcal{D}$ is k-decomposition, the tuples $(\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B)$ and $(\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B)$ are weak partitions of the rows and the columns, respectively. Therefore, there is exactly one set $Q_1 \in \{\mathcal{R}_1, ..., \mathcal{R}_k, \mathcal{R}_B\}$ and exactly one set $Q_2 \in \{\mathcal{C}_1, ..., \mathcal{C}_k, \mathcal{C}_B\}$ with $i \in Q_1$ and $j \in Q_2$. Thus, exactly one of the variables $y^R_{1i}, \ldots, y^R_{ki}, x^R_i$ attains value 1. Similarly, exactly one of the variables $y^C_{1j}, \ldots, y^C_{kj}, x^C_j$ reaches value 1. It follows that the constraints ($A1R$) and ($A1C$) are fulfilled for all $i \in [m]$ and for all $j \in [n]$, respectively. Let $t \in [k]$ be an arbitrary block. As $\mathcal{D}$ fulfills the load condition, at least $\ell^{\mathcal{R}}$ and at most $u^{\mathcal{R}}$ of the variables $y^R_{t1}, \ldots, y^R_{tm}$ take a value of one. Therefore, the constraints ($A2R\ell$) and ($A2Ru$) are fulfilled. Similarly, at least $\ell^{\mathcal{C}}$ and at most $u^{\mathcal{C}}$ of the variables $y^C_{t1}, \ldots, y^C_{tn}$ reach a value of 1. Hence, the constraints ($A2C\ell$) and ($A2Cu$) are respected. Consider again an arbitrary block $t \in [k]$. Also let $i \in [m]$ be a row and $j \in [n]$ be a column with $a_{ij} \neq 0$. For contradiction, suppose that the constraint ($A3C$) for (t, i, j) is violated. It follows that $y^C_{tj} = 1$, $y^R_{ti} = 0$ and $x^R_i = 0$. Hence, $y^R_{t^*i} = 1$ for some $t^* \in [k] \setminus \{t\}$. We thus get $i \in \mathcal{R}_t$ and $j \in \mathcal{C}_{t^*}$ for $t \neq t^*$, which contradicts the fact that $\mathcal{D}$ fulfills the block condition. Therefore, the constraints ($A3C$) are fulfilled and, in consequence, z is a feasible assignment for the variables of IP_A.

It remains to show that the objective function values of z and $\mathcal{D}$ are equal. Let $\mathcal{D}$ be feasible for $\textsc{MinAf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Then, as seen above, z is feasible for $IP_A(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. The objective value of $\mathcal{D}$ is

$$|\mathcal{R}_B| + |\mathcal{C}_B| = \sum_{i=1}^{m} x^R_i \ + \ \sum_{j=1}^{n} x^C_j ,$$

which is the objective function value of the assignment z.

$\square$

Remark 13:
Let IP^*_A denote the integer program IP_A with variable x^C_j fixed to zero for all $j \in [n]$. Consider a feasible solution z of IP^*_A. Let $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ be a k-decomposition obtained from z. Then we have $\mathcal{C}_B = \emptyset$. On the other hand, for every assignment that is obtained from a feasible solution of $\textsc{MinBf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, we obtain $x^C_j = 0$ for all $j \in [n]$. Therefore, IP^*_A solves $\textsc{MinBf}$.

A common method to solve integer programs like IP_A is the LP-based branch-and-bound algorithm. For an introduction to this algorithm we refer the reader to [39]. Its performance highly depends on the so-called "strength" of the so-called *LP-relaxation*. One obtains the LP-relaxation of an integer program by omitting the integrality conditions on the variables. Explicitly, for the LP-relaxation LP_A of IP_A the constraints ($A4$) are substituted by the following constraints:

$$0 \le y^R_{ti}, y^C_{tj}, x^R_i, x^C_j \le 1, \qquad \text{for } t \in [k],\ i \in [m],\ j \in [n]; \qquad (A4^*)$$
$$y^R_{ti}, y^C_{tj}, x^R_i, x^C_j \in \mathbb{R}, \qquad \text{for } t \in [k],\ i \in [m],\ j \in [n]. \qquad (A4^{**})$$

Thus, the set of feasible solution is not reduced (i.e. every feasible solution of IP_A is a feasible solution of LP_A). Therefore, the objective function value OPT^* of an optimal solution of LP_A provides a lower bound on the optimal objective function value OPT of IP_A. In the scope of this thesis, we will use the following intuition: If the gap between OPT and OPT^* is relatively big, then we call the LP-relaxation weak, otherwise we call it strong. For a deeper discussion of the "strength" of an LP-relaxation we refer the reader to [39].

Unfortunately, the LP-relaxation of IP_A is weak. This can be seen by considering the following assignment of variables: We set $y^R_{ti} := \frac{1}{k}$ for all $t \in [k]$, $i \in [m]$, $y^C_{tj} := \frac{1}{k}$ for all $t \in [k]$, $j \in [m]$, and the remaining variables to zero. Since $k \in \mathbb{N}$, this assignment fulfills the constraints $(A4^*)$ and $(A4^{**})$. Moreover, for all $i \in [m]$ we have

$$\sum_{t=1}^{k} y^R_{ti} + x^R_i = k\frac{1}{k} + 0 = 1,$$

and for all $j \in [n]$ we get

$$\sum_{t=1}^{k} y^C_{tj} + x^C_j = k\frac{1}{k} + 0 = 1,$$

and hence the constraints $(A1R)$ and $(A1C)$ are fulfilled. Furthermore, we have

$$y^C_{tj} - y^R_{ti} - x^R_i = \frac{1}{k} - \frac{1}{k} \le 0$$

for all $i \in [m], j \in [n]$, and $t \in [k]$. Therefore, the constraints $(A3C)$ are fulfilled. For all $t \in [k]$ we obtain

$$\sum_{i=1}^{m} y^R_{ti} = \frac{m}{k}$$

and

$$\sum_{j=1}^{n} y^C_{tj} = \frac{n}{k}.$$

Therefore, if we had

$$(*) \qquad \ell^{\mathcal{R}} \le \frac{m}{k} \le u^{\mathcal{R}} \qquad \text{and} \qquad \ell^{\mathcal{C}} \le \frac{n}{k} \le u^{\mathcal{C}},$$

the constraints $(A2R\ell)$, $(A2C\ell)$, $(A2Ru)$ and $(A2Cu)$ would be fulfilled. Since $\frac{m}{k}$ is the average number of rows per block and $\frac{n}{k}$ is the average number of columns per block, provided that the respective border is empty, it seems that assumption $(*)$ is natural. However, the objective function value of the assignment is zero. Hence, the assignment is an optimal solution of LP_A, if assumption $(*)$ holds. Since the objective function value is nonnegative, the gap between the the optimal function values of LP_A and IP_A is maximal for a fixed instance of MinAf. Thus, the LP-relaxation of IP_A is weak.

There is another problem with the formulation. Consider a feasible solution z of IP_A. Let $t, t' \in [k], t \neq t'$ be two distinct blocks. We obtain another feasible solution $\bar{z}$ of IP_A by permuting the assignment of the variables y^R_{ti} and $y^R_{t'i}$ for all $i \in [m]$ and exchanging the assignment of the variables y^C_{tj} and $y^C_{t'j}$ for all $j \in [n]$. Obviously, the objective function values of z and $\bar{z}$ are equal. All rows and columns, assigned to block t, are assigned to block t', and vice versa. We call such a permutation a *block permutation* of an assignment. There are not only pairwise block permutations. In fact, every block permutation of a feasible assignment, except the identity, yields another feasible assignment with the same objective function value. If the variables of an integer program can be permuted without changing the structure of the problem, then we call this integer program *symmetric*. It would go beyond the scope of this thesis to discuss symmetry of integer programs in detail. For a thorough treatment of this topic we refer the reader to [34]. However, if IP_A has at least one feasible solution, then there are at least $k!$ optimal solutions. Of course, there is at least one of them whose blocks are sorted in non-ascending order by their number of rows or columns. In the following, we are going to introduce two types of constraints. The first type of constraints is called *row block order constraints*$(ARBO)$. They forbid all assignments whose blocks are not sorted in nonascending order by their number of rows:

$$\sum_{i=1}^{m} y^R_{ti} - \sum_{i=1}^{m} y^R_{(t+1)i} \geq 0 \qquad \text{for } t \in [k-1]. \qquad (ARBO)$$

Clearly, these constraints ensure that the number of rows assigned to block t are not less than the number of rows assigned to block $t+1$ for $t \in [k-1]$. Similarly, we define the *column block order constraints*$(ACBO)$ as follows:

$$\sum_{j=1}^{n} y^C_{tj} - \sum_{j=1}^{n} y^C_{(t+1)j} \geq 0 \qquad \text{for } t \in [k-1]. \qquad (ACBO)$$

It is easily seen that these constraints ensure that the number of columns assigned to block t are not less than the number of columns assigned to block $t+1$ for $t \in [k-1]$. Both types of constraints can forbid feasible solutions of IP_A, but if IP_A is feasible, then for each of both kinds of constraints there is at least one optimal solution that fulfills all constraints of this type.

Let IP_{AR} denote the integer program IP_A with the additional constraints $(ARBO)$ and denote by IP_{AC} the integer program IP_A with the additional constraints $(ARCO)$. From the above it follows that optimal solutions of IP_{AR} and IP_{AC} are also optimal solutions of IP_A.

It is worth pointing out that if we added constraints of both types to IP_A, then we might prune all optimal solutions.

We will examine in section 6.2 whether there is an improvement in performance by solving IP_{AR} or IP_{AC} instead of IP_A.

5.3 Column Generation Approach for MinAf

In the following section, we introduce an integer program that is based on IP_A, but hopefully provides a stronger LP-relaxation. At first, we present the main idea of the model. After giving some necessary definitions, we present the model and verify that it can be used to solve MINAF and MINBF. Furthermore, we show how to solve the LP-relaxation of the model with a so-called column generation approach. .

The weakness of the LP-relaxation of IP_A was caused by the fact that every row and column could be fractionally assigned to every block. We want to avoid assignments whose number of fractionally assigned rows or columns exceeds one of the block capacities. The main idea is to introduce a binary variable for every block and every possible assignment of rows and columns ("block pattern"). The advantage of using these kind of variables lies in the fact that assignments may force some other rows or columns to be part of the border. For instance, consider a row assignment that includes row i, but excludes column j with j has a nonzero entry in i. It is necessary that j is assigned to the border since assigning c to another block would violate the block condition.

We continue by giving some helpful definitions.

Definition 5.3.1 (Block pattern for a matrix)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix. We call a pair $p = (R, C)$ of sets $R \subseteq [m]$ and $C \subseteq [n]$ a *block pattern* for A.

Let $A \in \mathbb{R}^{m\times n}$ be a matrix and let $p = (R, C)$ be a block pattern for A. For $i \in R$ and $j \in C$, we say p *contains* i and j, respectively.

Observation 5.3.2
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $k \in \mathbb{N}$ be a positive integer. Furthermore, suppose that $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ is a k-decomposition of A. Then for $t \in [k]$, $p_t = (\mathcal{R}_t, \mathcal{C}_t)$ is a block pattern for A.

Definition 5.3.3 (Neighborhood of a block pattern)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $p = (R, C)$ a block pattern for A. We call the pair of sets $\bar{p} = (\bar{R}, \bar{C})$ the *neighborhood* of p with $\bar{R} = \{i \in [m] | i \notin R \wedge \exists j \in C : a_{ij} \neq 0\}$ the set of *neighbor rows* of p and $\bar{C} = \{j \in [n] | j \notin C \wedge \exists i \in R : a_{ij} \neq 0\}$ the set of *neighbor columns* of p.

This lemma provides an alternative formulation for the block condition and will be needed later.

Lemma 5.3.4 (Alternative block conditions)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and let $k \in \mathbb{N}$ be a positive integer. Moreover, suppose that $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ is a k-decomposition of A and define $p_t = (\mathcal{R}_t, \mathcal{C}_t)$ for $t \in [k]$. Furthermore, let $\bar{p}_t = (\bar{\mathcal{R}}_t, \bar{\mathcal{C}}_t)$ be the neighborhood of p_t. Then the following conditions are equivalent:

(i) $\mathcal{D}$ fulfills the block condition.

(ii) For all $t \in [k]$ holds $\bar{\mathcal{R}}_t \subseteq \mathcal{R}_B$.

(iii) For all $t \in [k]$ holds $\bar{\mathcal{C}}_t \subseteq \mathcal{C}_B$.

Proof: Let $A \in \mathbb{R}^{m\times n}$ be a matrix and let $k \in \mathbb{N}$ be a positive integer. Suppose that $\mathcal{D} = (\mathcal{R}_1,\dots,\mathcal{R}_k,\mathcal{R}_B;\mathcal{C}_1,\dots,\mathcal{C}_k,\mathcal{C}_B)$ is a k-decomposition of A and write $p_t = (\mathcal{R}_t,\mathcal{C}_t)$ for $t \in [k]$. Moreover, let $\bar{p}_t = (\bar{\mathcal{R}}_t,\bar{\mathcal{C}}_t)$ be the neighborhood of p_t.

We begin by deducing (ii) from (i). Suppose that $\mathcal{D}$ fulfills the block condition. Let $t \in [k]$ be a block and consider $i \in \bar{\mathcal{R}}_t$. This gives $i \notin \mathcal{R}_t$ and there is a column $j \in \mathcal{C}_t$ such that $a_{ij} \neq 0$. Since the block condition would be violated if $i \in \mathcal{R}_{t'}$ for some $t' \in [k] \setminus \{t\}$, it follows that $i \notin \mathcal{R}_{t'}$ for all $t' \in [k] \setminus \{t\}$. By the definition of a k-decomposition, $(\mathcal{R}_1,...,\mathcal{R}_k,\mathcal{R}_B)$ is a weak partition of the rows of A, and consequently $i \in \mathcal{R}_B$.

We now proceed by deducing (iii) from (ii). Suppose that for all $t \in [k]$ holds $\bar{\mathcal{R}}_t \subseteq \mathcal{R}_B$. Consider an arbitrary block $t \in [k]$ and a column $j \in \bar{\mathcal{C}}_t$. By the definition of neighborhood, we obtain that $j \notin \mathcal{C}_t$ and there is a row $i \in \mathcal{R}_t$ such that $a_{ij} \neq 0$. We claim that $j \in \mathcal{C}_B$. Suppose, contrary to our claim, that $j \in \mathcal{C}_{t'}$ for some $t' \in [k] \setminus \{t\}$. It follows that $i \in \bar{\mathcal{R}}_{t'}$ since $i \notin \mathcal{R}_{t'}$. By (ii) we obtain $i \in \mathcal{R}_B$. This contradicts the fact that $i \in \mathcal{R}_t$, because $(\mathcal{R}_1,...,\mathcal{R}_k,\mathcal{R}_B)$ is a weak partition of the rows of A. Hence, we have $j \in \mathcal{C}_B$.

Finally, we prove that (iii) implies (i). Suppose that for all $t \in [k]$ holds $\bar{\mathcal{C}}_t \subseteq \mathcal{C}_B$. Let $i \in \mathcal{R}_t$ be a row and $j \in \mathcal{C}_{t'}$ be a column for some $t,t' \in [k]$ such that $a_{ij} \neq 0$. To obtain a contradiction, suppose that $t \neq t'$. Since $(\mathcal{C}_1,...,\mathcal{C}_k,\mathcal{C}_B)$ is a weak partition of the columns, we have $j \notin \mathcal{C}_t$ and therefore get $j \in \bar{\mathcal{C}}_t$. From (iii) we obtain $j \in \mathcal{C}_B$, contrary to $j \in \mathcal{C}_{t'}$. This clearly forces $t = t'$. Consequently, $\mathcal{D}$ fulfills the block condition.

$\square$

Since we are only interested in block patterns that fulfill some load condition, we define:

Definition 5.3.5 (Feasible block pattern for a matrix)
Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $p = (R,C)$ a block pattern for A. Furthermore, let $u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be positive integers, and let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ be nonnegative integers. We call p a *feasible block pattern* for A under the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ if $\ell^{\mathcal{R}} \leq |R| \leq u^{\mathcal{R}}$ and $\ell^{\mathcal{C}} \leq |C| \leq u^{\mathcal{C}}$.

To shorten notation, we write *feasible block pattern* if it is clear which matrix and load condition are meant.

Even for small instances the number of feasible block patterns can be large. The number of all i-elementary subsets of an m-elementary set is $\binom{m}{i}$. Therefore, the number of all feasible row assignments and all column assignments is

$$\sum_{i=\ell^{\mathcal{R}}}^{u^{\mathcal{R}}} \binom{m}{i} \text{ and } \sum_{j=\ell^{\mathcal{C}}}^{u^{\mathcal{C}}} \binom{n}{j},$$

respectively. Since every feasible block pattern is a unique combination of a feasible row and feasible column assignment, the number of feasible block patterns is

$$\left(\sum_{i=\ell^{\mathcal{R}}}^{u^{\mathcal{R}}} \binom{m}{i}\right) \left(\sum_{j=\ell^{\mathcal{C}}}^{u^{\mathcal{C}}} \binom{n}{j}\right),$$

the product of the number of feasible row assignments and the number of feasible column assignments.

Given an arbitrary matrix $A \in \mathbb{R}^{m\times n}$ and the load condition $(1, m-1, 1, n-1)$, we can calculate the number of feasible block patterns. Due to the fact that for $k \in \mathbb{N}$ we have

$$\sum_{i=1}^{k-1} \binom{k}{i} = \sum_{i=0}^{k} \binom{k}{i} - \binom{k}{0} - \binom{k}{k} = (1+1)^k - 2 = 2^k - 2,$$

it follows that there are $2^m - 2$ feasible row assignments and $2^n - 2$ feasible column assignments. Hence, there are

$$(2^m - 2)(2^n - 2) = 2^{m+n} - 2^{m+1} - 2^{n+1} + 4$$

feasible block patterns.

Consider for example a 20×20 matrix with the load condition $(1, 19, 1, 19)$. The number of feasible block patterns are

$$2^{40} - 2^{21} - 2^{21} + 4 \approx 10^{12},$$

which is a huge number for a relatively small matrix.

In the following we introduce the integer program IP_{CG} according to some instance MinAf$(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}.\ell^{\mathcal{C}}, u^{\mathcal{C}})$. The used notation is set up afterwards.

$$(IP_{CG}) \quad \begin{array}{llrr}
\text{Minimize} & \displaystyle\sum_{i=1}^{m} x_i^R + \sum_{j=1}^{n} x_j^C & & \\
\text{subject to} & \displaystyle\sum_{t=1}^{k} y_{ti}^R + x_i^R = 1, & \text{for } i \in [m]; & (C1R) \\
& \displaystyle\sum_{t=1}^{k} y_{tj}^C + x_j^C = 1, & \text{for } j \in [n]; & (C1C) \\
& \displaystyle\sum_{p\in P_t} a_i^p z_t^p - y_{ti}^R = 0, & \text{for } t \in [k],\ \text{for } i \in [m]; & (C2R) \\
& \displaystyle\sum_{p\in P_t} b_j^p z_t^p - y_{ti}^C = 0, & \text{for } t \in [k],\ \text{for } j \in [n]; & (C2C) \\
& \displaystyle\sum_{p\in P_t} \bar{a}_i^p z_t^p - x_i^R \leq 0, & \text{for } t \in [k],\ \text{for } i \in [m]; & (C3R) \\
& \displaystyle\sum_{p\in P_t} \bar{b}_j^p z_t^p - x_i^C \leq 0, & \text{for } t \in [k],\ \text{for } j \in [n]; & (C3C) \\
& \displaystyle\sum_{p\in P_t} z_t^p = 1, & \text{for } t \in [k]; & (C4) \\
& y_{ti}^R, y_{ti}^C, x_i^R, x_j^C \in \{0,1\}, & \text{for } t \in [k],\ i \in [m],\ j \in [n]; & (C5) \\
& z_t^p \in \{0,1\}, & \text{for } t \in k,\ p \in P_t. & (C6)
\end{array}$$

Let us denote by P_t the set of all feasible block patterns for A according to $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ that can be assigned to block t for $t \in [k]$. For such a feasible block pattern $p = (R, C)$ with neighborhood $\bar{p} = (\bar{R}, \bar{C})$, a row $i \in [m]$, and a column $j \in [n]$, we define the parameters $a_i^p, b_j^p, \bar{a}_i^p, \bar{b}_j^p \in \{0, 1\}$ such that

- $a_i^p = 1$ if and only if $i \in R$,
- $b_j^p = 1$ if and only if $j \in C$,
- $\bar{a}_i^p = 1$ if and only if $i \in \bar{R}$, and
- $\bar{b}_j^p = 1$ if and only if $j \in \bar{C}$.

Moreover, our model contains, additionally to the variables of IP_A, a binary variable z_t^p, called *pattern variable*, for every block $t \in [k]$ and every feasible block pattern $p \in P_t$. This variable reaches value one if and only if block t contains exactly those rows and columns that are contained in block pattern p.

According to the constraints $(C4)$, for every block t exactly one block pattern is chosen. Therefore, the constraints $(C2R)$ ensure that for all $i \in [m]$ and $t \in [k]$ the variable y_{ti}^R takes value of one if and only if the chosen block pattern for block t contains row i. Similarly, the constraints $(C2C)$ assure that for all $j \in [n]$ and $t \in [k]$ the variable y_{tj}^C reaches a value of one if and only if the chosen block pattern for block t contains column j. Furthermore, the constraints $(C1R)$ and $(C1C)$ guarantee that each row and each column, respectively, is either assigned to exactly one block or the respective border.

Let $A \in \mathbb{R}^{m \times n}$ be a matrix, let $k, u^{\mathcal{R}}, u^{\mathcal{C}} \in \mathbb{N}$ be positive integers, and let $\ell^{\mathcal{R}}, \ell^{\mathcal{C}} \in \mathbb{N}_0$ be nonnegative integers. In the following, we want to show how a feasible solution of $IP_{CG}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ can be transformed to a feasible solution of the problem $\textsc{MinAf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, and vice versa. Furthermore, it will turn out that the corresponding objective function values are equal.

On account of the constraints $(C4)$, a feasible variable assignment for IP_{CG} includes for every $t \in [k]$ exactly one variable z_t^p that attains a value of 1. Let $p_t = (R_t, C_t)$ be the feasible block pattern with $z_t^{p_t} = 1$ for $t \in [k]$. Define $R^* := [m] \setminus \bigcup_{t=1}^{k} R_t$ and $C^* := [n] \setminus \bigcup_{t=1}^{k} C_t$. By setting $\mathcal{R}_t = R_t$ and $\mathcal{C}_t = C_t$ for $t \in [k]$, plus $\mathcal{R}_B = R^*$ and $\mathcal{C}_B = C^*$, we obtain a k-decomposition $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ from a feasible assignment. Since the block pattern p_t for $t \in [k]$ is feasible under $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, $\mathcal{D}$ fulfills the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Consider $\bar{p}_t = (\bar{R}, \bar{C})$ the neighborhood of block pattern p_t for $t \in [k]$. The constraints $(C3R)$ ensures that every row being a neighbor row of some chosen block pattern is assigned to the row border. Similarly, the constraints $(C3C)$ guarantee that every column being a neighbor column of some chosen block pattern is assigned to the column border. Hence, for all $t \in [k]$ we have $\bar{R}_t \subseteq R^*$ and $\bar{C}_t \subseteq C^*$. By Lemma 5.3.4, it follows that $\mathcal{D}$ fulfills the block condition. Hence, $\mathcal{D}$ is feasible for $\textsc{MinAf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Furthermore, $\mathcal{D}$ and the assignment of the variables has the same objective function values.

On the other hand, consider a k-decomposition $\mathcal{D} = (\mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{R}_B; \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_B)$ that is feasible for $\textsc{MinAf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. We can deduce a feasible variable assignment for the IP_{CG} according to $\textsc{MinAf}(A, k, \ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ from $\mathcal{D}$ whose objective function value equals the objective function value of $\mathcal{D}$. For $t \in [k]$ we define $p_t = (\mathcal{R}_t, \mathcal{C}_t)$ the block patterns that are chosen. Since $\mathcal{D}$ fulfills the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$, p_t is a feasible block pattern for A under the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ and hence $p_t \in P_t$ for $t \in [k]$. We set $z_t^{p_t} = 1$ and $z_t^p = 0$ for $t \in [k]$, $p \in P_t \setminus \{p_t\}$. Thus, the constraints $(C4)$ are respected. Moreover, for $t \in [k]$ we set $y_{ti}^R = 1$ if and only if $i \in \mathcal{R}_t$. Also, we set $y_{tj}^C = 1$ if and only if $j \in \mathcal{C}_t$. Furthermore, we set $x_i^R = 1$ if and only if $i \in \mathcal{R}_B$, and we assign $x_j^C = 1$ if and only if $j \in \mathcal{C}_B$. It follows immediately that the constraints $(C2R)$ and $(C2C)$ are fulfilled and that the objective function values of $\mathcal{D}$ and the deduced assignment for IP_{CG} are equal. Due to the fact that $\mathcal{D}$ is a k-decomposition of A, the constraints $(C1R)$ and $(C1C)$ are also fulfilled. Consider the neighborhood $\bar{p}_t = (\bar{R}_t, \bar{C}_t)$ of p_t for $t \in [k]$. Since $\mathcal{D}$ fulfills the block condition, Lemma 5.3.4 shows that for all $t \in [k]$, $\bar{R}_t \subseteq \mathcal{R}_B$ and $\bar{C}_t \subseteq \mathcal{C}_B$. Hence, for all $t \in [k]$, for all rows $i \in \bar{R}_t$ and for all $j \in \bar{C}_t$, we have $x_i^R = 1$ and $x_j^C = 1$. Therefore, the constraints $(C3R)$ and $(C3C)$ are fulfilled. Consequently, the deduced assignment fulfills all constraints and its objective function value equals the objective function value of $\mathcal{D}$.

It is worth pointing out that if we fix the variables x_j^C to zero for $j \in [n]$ we can use IP_{CG} to solve MinBf. This can be seen by the same reasoning as in Remark 13.

5.3.1 Solving the LP-relaxation of IP_{CG}

We have seen that the number of feasible block patterns and thus the number of the variables z_t^p can be large even for small instances. Instead of generating all variables in advance, we only generate them when needed. In order to do so, we utilize the fact that the simplex algorithm can solve an LP without keeping track of all possible variables. It is not our purpose to study the simplex algorithm in detail. For a deeper discussion of the simplex algorithm we refer the reader to [10] or [14].

For our implementation this means, that the set P_t does not contain all variables z_t^p. To be more precise, we initialize it in the following way:

$$P_t := \{z_t^p | p = (\{i\}, \{j\}), i \in [m], j \in [n], a_{ij} \neq 0\}$$

for all $t \in [k]$. Thus, P_t consists of artificial pattern variables that each correspond to a nonzero entry of A. We set the objective cost coefficient of an artificial z_t^p to $m + n + 1$ if pattern p is not feasible according to $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. Hence, in an optimal feasible solution the pattern variables that belong to infeasible patterns attains a value of 0.

In every iteration of the simplex algorithm we search a variable with so-called negative reduced cost. In general, the reduced costs for a variable v can be calculated using the objective function coefficient of v and the dual values of the constraints containing v in the current simplex iteration. For all $t \in [k]$ and $i \in [m]$, we denote the dual value of the constraint $(C2R)$ for (t, i) by α_{ti}, and the dual value of the constraint $(C3R)$ for (t, i) by $\bar{\alpha}_{ti}$. Moreover, for all $t \in [k]$ and $j \in [n]$, we denote the dual value of the constraint $(C2C)$ for (t, j) by β_{tj} and the dual value of the constraint $(C3C)$ for (t, j) by $\bar{\beta}_{tj}$. Furthermore, for $t \in [k]$ the dual value of the constraint $(C4)$ for t is denoted by γ_t. We thus obtain the reduced costs $\bar{c}_t^p$ for the variable z_t^p with $t \in [k]$, $p = (R, C) \in P_t$

and $\bar{p} = (\bar{R}, \bar{C})$, the neighborhood of p, by

$$\begin{aligned} \bar{c}_t^p &= 0 - \left(\sum_{i=1}^m a_i^p \alpha_{ti} + \sum_{j=1}^n b_j^p \beta_{tj} + \sum_{i=1}^m \bar{a}_i^p \bar{\alpha}_{ti} + \sum_{j=1}^n \bar{b}_j^p \bar{\beta}_{tj} + \gamma_t \right) \\ &= - \left(\sum_{i \in R} \alpha_{ti} + \sum_{j \in C} \beta_{tj} + \sum_{i \in \bar{R}} \bar{\alpha}_{ti} + \sum_{j \in \bar{C}} \bar{\beta}_{tj} + \gamma_t \right). \end{aligned}$$

.

To find variables with negative reduced costs explicitly, we can iterate over all pairs (t, p) of blocks $t \in [k]$ and feasible block patterns $p \in P_t$, and stop if we have found a variable with negative reduced cost. Of course, if there are too many feasible block patterns, this procedure is not efficient. Instead of doing that, we solve an optimization problem for every block $t \in [k]$. In fact, given a block $t \in [k]$ we look for a feasible block pattern p such that $\bar{c}_t^p$ is minimal. For fixed $t \in k$, we observe that γ_t is a constant and therefore minimizing $\bar{c}_t^p$ is the same as maximizing

$$s_t^p := \sum_{i \in R} \alpha_{ti} + \sum_{j \in C} \beta_{tj} + \sum_{i \in \bar{R}} \bar{\alpha}_{ti} + \sum_{j \in \bar{C}} \bar{\beta}_{tj},$$

over all feasible block patterns $p = (R, C) \in P_t$ with neighborhood $\bar{p} = (\bar{R}, \bar{C})$. Furthermore, we observe that the dual variables $\bar{\alpha}_{ti}$ and $\bar{\beta}_{tj}$ are nonpositive for $i \in [m], j \in [n]$ because the corresponding constraints $(C3R)$ and $(C3C)$ are “less-or-equal” constraints and the problem is a minimization problem.

Thus, we solve the following problem for every block $t \in [k]$:

Maximum Capacitated Block Pattern Score

Instance: Matrix $A \in \mathbb{R}^{m \times n}$, $\ell^R, \ell^C \in \mathbb{N}_0$ and $u^R, u^C \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$ and $\bar{\alpha}_i \in \mathbb{R}_{\leq 0}$ for $i \in [m]$, and $\beta_j \in \mathbb{R}$ and $\bar{\beta}_j \in \mathbb{R}_{\leq 0}$ for $j \in [n]$

Solution: A feasible block pattern $p = (R.C)$ (with neighborhood $\bar{p} = (\bar{R}, \bar{C})$) for A under the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$

Objective: Maximize $\sum_{i \in R} \alpha_i + \sum_{j \in C} \beta_j + \sum_{i \in \bar{R}} \bar{\alpha}_i + \sum_{j \in \bar{C}} \bar{\beta}_j$

To shorten notation, we write MCBPS for the Maximum Capacitated Block Pattern Score problem. If $\ell^{\mathcal{R}} = \ell^{\mathcal{C}} = 0$, $u^{\mathcal{R}} = m$ and $u^{\mathcal{C}} = n$, then we call it the Maximum Uncapacitated Block Pattern Score problem and write MUBPS.

After solving MCBPS for $A, \ell^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{R}}, u^{\mathcal{C}}$ and $\{\alpha_i, \bar{\alpha}_i, \beta_j, \bar{\beta}_j | i \in [m], j \in [n]\}$, for a $t \in [k]$, we add the variable z_t^p if the reduced costs $\bar{c}_t^p$ are negative. To be more precise, the variable z_t^p is added if $val(p) + \gamma_t > 0$ where $val(p)$ denotes the objective function value of p for MCBPS.

In our implementation we solve the MCBPS problem for every block $t \in [k]$. In every pricing round we add variables up to a fixed number to P_t for all $t \in [k]$. We solve the MCBPS problem with the integer program IP_P indicated at the end of the next subsection.

5.3.2 The Pricing Problem

At first, we want to show that the uncapacitated version of the problem ($\ell^{\mathcal{R}} = \ell^{\mathcal{C}} = 0$, $u^{\mathcal{R}} = m$ and $u^{\mathcal{C}} = n$) can be solved in polynomial time in the input size. One question that is still unanswered is whether this is true for the capacitated version of the pricing problem. Finally, we introduce an integer program that solves MCBPS.

Complexity of the Uncapacitated Version

We show that the uncapacitated version of MCBPS can be solved in polynomial time in the input size. In order to do so, we transform an instance of MUBPS to an instance of the so-called Maximum s-Excess problem that we introduce presently. This problem, going back to the work of Hochbaum [21], was shown to be equivalent to the well-known Minimum Cut problem.

Maximum s-Excess

Instance: A directed graph $G = (V, \mathcal{A})$, node weights $w_i \in \mathbb{R}$ for all $i \in V$ and arc weights $c_{ij} \in \mathbb{R}_{\geq 0}$ for all $(i, j) \in \mathcal{A}$

Solution: A subset of nodes $S \subseteq V$

Objective: Maximize $\sum\limits_{i \in S} w_i - \sum\limits_{i \in S, j \in \bar{S}} c_{ij}$, with $\bar{S} = V \setminus S$

For every subset of nodes $S \subseteq V$ we call an arc $(i, j) \in \mathcal{A}$ with $i \in S$ and $j \notin S$ a *cut arc* of S and $\delta^-(S) := \{(i, j) \in \mathcal{A} | i \in S, j \notin S\}$ the *set of cut arcs* of S.

Proposition 5.3.6
The Maximum s-Excess *problem can be solved in polynomial time in the input size.*

Proof: The proof was done by Hochbaum in [21] and is omitted.

We make use of this result to show that one can solve MUBPS in polynomial time.

Proposition 5.3.7
The MUBPS *problem can be solved in polynomial time in the input size.*

Proof: Consider an instance I_{MU} of the MUBPS problem, i.e. a matrix $A \in \mathbb{R}^{m \times n}$, real numbers $\alpha_i, \beta_j \in \mathbb{R}$ for all $i \in [m], j \in [n]$ and real non-positive numbers $\bar{\alpha}_i, \bar{\beta}_j \in \mathbb{R}_{\leq 0}$ for $i \in [m], j \in [n]$.

We are going to construct an instance I_{MA} of the Maximum s-Excess problem for I_{MU} that has the following two properties: On the one hand, every solution of I_{MU} can be transformed to a solution of I_{MA} with the same objective function value. On the other hand, there is an optimal solution S^* of I_{MA} that can be obtained by this transformation from a solution p^* of I_{MU}. Moreover, we can obtain p^* from S^*. Therefore, p^* is optimal for I_{MU}. It will turn out that the construction and transformations can be accomplished in polynomial time in the input size, and hence we can use I_{MA} to solve I_{MU} in polynomial time in the input size.

We start by constructing such an instance I_{MA} of the Maximum s-Excess problem from I_{MU}. Consider the following directed graph $G = (V, \mathcal{A})$. For every row $i \in [m]$ of A, G includes two vertices s_i and $\bar{s}_i$ with weights $w_{s_i} = \alpha_i$ and $w_{\bar{s}_i} = 0$ that are connected by an arc $(\bar{s}_i, s_i)$ with weight $c_{\bar{s}_i s_i} = -\bar{\alpha}_i \geq 0$. Analogously, G includes two vertices t_j and $\bar{t}_j$ with weights $w_{t_j} = \beta_j$ and $w_{\bar{t}_j} = 0$ that are connected by an arc $(\bar{t}_j, t_j)$ with weight $c_{\bar{t}_j t_j} = -\bar{\beta}_j \geq 0$ for every column $j \in [n]$ of A. Furthermore, there are two arcs $(s_i, \bar{t}_j)$ and $(t_j, \bar{s}_i)$ for every nonzero entry $a_{ij} \neq 0$ of A with weights $c_{s_i \bar{t}_j} = c_{t_j \bar{s}_i} = M$ with

$$M := 1 + \sum_{i \in [m], \alpha_i > 0} \alpha_i + \sum_{j \in [n], \beta_j > 0} \beta_j \geq 0.$$

Thus, we constructed the directed graph $G = (V, \mathcal{A})$ with

$$V = \{s_i | i \in [m]\} \cup \{\bar{s}_i | i \in [m]\} \cup \{t_j | j \in [n]\} \cup \{\bar{t}_j | j \in [n]\}$$

and

$$\begin{aligned} \mathcal{A} = & \{(\bar{s}_i, s_i) | i \in [m]\} \cup \{(\bar{t}_j, t_j) | j \in [n]\} \cup \{(s_i, \bar{t}_j) | i \in [m], j \in [n], a_{ij} \neq 0\} \\ & \cup \{(t_j, \bar{s}_i) | i \in [m], j \in [n], a_{ij} \neq 0\}. \end{aligned}$$

G is sketched in Figure5.1. For $i \in [m]$ and $j \in [n]$ with $a_{ij} \neq 0$ a pair of edges with weight M is displayed. However, the fact that there are more arcs with weight M is adumbrated by the dashed arrows.

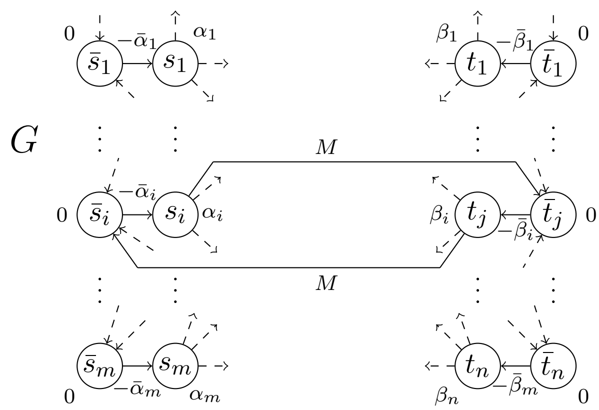


Figure 5.1: Sketch of the constructed instance I_{MA} of the Maximum s-Excess problem

Consider a feasible solution for I_{MA}, i.e. a block pattern $p = (R, C)$ with neighborhood $\bar{p} = (\bar{R}, \bar{C})$ and objective function value:

$$\sum_{i \in R} \alpha_i + \sum_{j \in C} \beta_j + \sum_{i \in \bar{R}} \bar{\alpha}_i + \sum_{j \in \bar{C}} \bar{\beta}_j.$$

We define

$$\begin{aligned} S(p) :=& \{s_i \in V | i \in R\} \cup \{\bar{s}_i \in V | i \in R \vee i \in \bar{R}\} \\ & \cup \{t_j \in V | j \in C\} \cup \{\bar{t}_j \in V | j \in C \vee j \in \bar{C}\}, \end{aligned}$$

and obtain

$$\begin{aligned} \delta^-(S(p)) =& \{(v, v') \in \mathcal{A} | v \in S(p), v' \notin S(p)\} \\ =& \{(\bar{s}_i, s_i) | i \in [m], \bar{s}_i \in S(p), s_i \notin S(p)\} \cup \{(\bar{t}_j, t_j) | j \in [n], \bar{t}_j \in S(p), t_j \notin S(p)\} \\ & \cup \{(s_i, \bar{t}_j) | i \in [m], j \in [n], a_{ij} \neq 0, s_i \in S(p), \bar{t}_j \notin S(p)\} \\ & \cup \{(t_j, \bar{s}_i) | i \in [m], j \in [n], a_{ij} \neq 0, t_j \in S(p), \bar{s}_i \notin S(p)\} \\ =& \{(\bar{s}_i, s_i) | i \in \bar{R}\} \cup \{(\bar{t}_j, t_j) | j \in \bar{C}\} \\ & \cup \{(s_i, \bar{t}_j) | i \in R, j \in [n] \setminus C, j \in [n] \setminus \bar{C}, a_{ij} \neq 0\} \\ & \cup \{(t_j, \bar{s}_i) | j \in C, i \in [m] \setminus R, i \in [m] \setminus \bar{R}, a_{ij} \neq 0\} \\ =& \{(\bar{s}_i, s_i) | i \in \bar{R}\} \cup \{(\bar{t}_j, t_j) | j \in \bar{C}\} \cup \emptyset \cup \emptyset, \end{aligned} \tag{5.1}$$

from the definition of neighborhood.
Therefore, the objective function value of $S(p)$ is

$$\begin{aligned} \sum_{v \in S(p)} w_v - \sum_{(v,v') \in \delta^-(S(p))} c_{vv'} \overset{5.1}{=} & \sum_{s_i \in S(p)} w_{s_i} + \sum_{\bar{s}_i \in S(p)} w_{\bar{s}_i} + \sum_{t_j \in S(p)} w_{t_j} + \sum_{\bar{t}_j \in S(p)} w_{\bar{t}_j} \\ & - \left(\sum_{i \in \bar{R}} c_{s_i \bar{s}_i} + \sum_{j \in \bar{C}} c_{t_j \bar{t}_j} \right) \\ =& \sum_{i \in R} \alpha_i + 0 + \sum_{j \in C} \beta_j + 0 - \left(\sum_{i \in \bar{R}} -\bar{\alpha}_i \right) - \left(\sum_{j \in \bar{C}} -\bar{\beta}_j \right) \\ =& \sum_{i \in R} \alpha_i + \sum_{j \in C} \beta_j + \sum_{i \in \bar{R}} \bar{\alpha}_i + \sum_{j \in \bar{C}} \bar{\beta}_j. \end{aligned}$$

Hence, the objective function values of p and $S(p)$ are equal.

Now, let us consider an optimal solution S^* for I_{MA}. We can assume w.l.o.g. that if $s_i \in S^*$ for some $i \in [m]$, then it is $\bar{s}_i \in S^*$, and analogously, if $t_j \in S^*$ for some $j \in [n]$, then it holds that $\bar{t}_j \in S^*$, otherwise we add the respective nodes. This does not decrease the objective function value because for all $i \in [m], j \in [n]$ we have $w_{\bar{s}_i} = w_{\bar{t}_j} = 0$, and the outgoing arc of $\bar{s}_i$ (and $\bar{t}_j$) cannot be a cut arc since $s_i \in S^*$ (and $t_j \in S^*$). Moreover, again w.l.o.g. we can assume that if $s_i \notin S^*$ and there is no $j \in [n]$ with $a_{ij} \neq 0$ and $t_j \in S^*$, then $\bar{s}_i \notin S^*$ holds. Otherwise we delete $\bar{s}_i$ from S^* without decreasing the objective function value since all incoming arcs of $\bar{s}_i$ are no cut arcs anyway and the weight of $\bar{s}_i$ is zero. Analogously, we can assume that if $t_j \notin S^*$ and there is no $i \in [m]$ with $a_{ij} \neq 0$ and $s_i \in S^*$, then we have $\bar{t}_j \notin S^*$.

We make use of the following claims: For all $i \in [m]$ holds:

$$\bar{s}_i \in S^* \Leftrightarrow s_i \in S^* \vee \exists j \in [n], a_{ij} \neq 0, t_j \in S^* \tag{5.2}$$

and for all $j \in [n]$ holds:

$$\bar{t}_j \in S^* \Leftrightarrow t_j \in S^* \vee \exists i \in [m], a_{ij} \neq 0, s_i \in S^* \tag{5.3}$$

We only proof Claim 5.2. Claim 5.3 follows with analogous arguments.

Let $i \in [m]$ be an arbitrary row. At first, we show " $\Rightarrow$" by contraposition. Assume that $s_i \notin S^*$ and for all $j \in [n]$ with $a_{ij} \neq 0$ holds $t_j \notin S^*$. By the assumptions seen above, we have $\bar{s}_i \notin S^*$ which proofs the statement. Now we show " $\Leftarrow$". We distinguish two cases:

1. If $s_i \in S^*$ we obtain $\bar{s}_i \in S^*$ by the assumption seen above.
2. If $s_i \notin S^*$ and there is a $j \in [n]$ such that $a_{ij} \neq 0$ and $t_j \in S^*$, then we assume by contradiction that $\bar{s}_i \notin S^*$. Then the arc $(t_j, \bar{s}_i)$ would be a cut arc with weight $M = 1 + \sum_{i \in [m], \alpha_i > 0} \alpha_i + \sum_{j \in [n], \beta_j > 0} \beta_j$. Hence, the objective function value of S^* would be smaller than zero, which contradicts the optimality of S^* since the empty set $\emptyset \subseteq V$ is a solution of I_{MA} with objective function value zero. Therefore, we have $\bar{s}_i \in S^*$.

Thus, we have proven Claim 5.2.

In the following we construct a block pattern p^* from S^*. We set $p^* = (R^*, C^*)$ with $R^* = \{i \in [m] | s_i \in S^*\}$ and $C^* = \{j \in [n] | t_j \in S^*\}$. We obtain the neighborhood $\bar{p} = (\bar{R}^*, \bar{C}^*)$ of p^* with:

$$\bar{R}^* = \{i \in [m] | i \notin R^* \wedge \exists j \in C^* : a_{ij} \neq 0\} \tag{5.4}$$

$$\bar{C}^* = \{j \in [n] | j \notin C^* \wedge \exists i \in R^* : a_{ij} \neq 0\} \tag{5.5}$$

It remains to prove that $S(p^*) = S^*$. It holds that

$$\begin{aligned}
S(p^*) &\overset{\text{Def.}}{=} \{s_i \in V | i \in R^*\} \cup \{\bar{s}_i \in V | i \in [m] : i \in R^* \vee i \in \bar{R}^*\} \\
&\quad \cup \{t_j \in V | j \in C^*\} \cup \{\bar{t}_j \in V | j \in [n] : j \in C^* \vee j \in \bar{C}^*\} \\
&\overset{5.45.5}{=} \{s_i \in S^* | i \in [m]\} \cup \{\bar{s}_i \in V | i \in [m] : i \in R^* \vee \exists j \in C^* : a_{ij} \neq 0\} \\
&\quad \cup \{t_j \in S^* | j \in [n]\} \cup \{\bar{t}_j \in V | j \in [n] : j \in C^* \vee \exists i \in R^* : a_{ij} \neq 0\} \\
&\overset{5.25.3}{=} \{s_i \in S^* | i \in [m]\} \cup \{\bar{s}_i \in S^* | i \in [m]\} \\
&\quad \cup \{t_j \in S^* | j \in [n]\} \cup \{\bar{t}_j \in S^* | j \in [n]\} \\
&= S^*.
\end{aligned}$$

Therefore, p^* is optimal for I_{MU}. The constructions and transformations can be accomplish in polynomial time in the input size, and hence we can use I_{MA} to solve I_{MU} in polynomial time in the input size.

□

An Integer Program for Solving MCBPS

In the following, we introduce the integer program IP_P that can be used to solve the Maximum Capacitated Block Pattern Score problem. In our implementation we have used IP_P to solve the pricing problem. Consider an instance of MCBPS given by a matrix $A \in \mathbb{R}^{m\times n}$, $\ell^R, \ell^C \in \mathbb{N}_0$, and $u^R, u^C \in \mathbb{N}$, $\alpha_i, \beta_j \in \mathbb{R}$ and $\bar{\alpha}_i, \bar{\beta}_j \in \mathbb{R}_{\leq 0}$ for $i \in [m]$, $j \in [n]$. We look for a feasible block pattern $p = (R.C)$ (with neighborhood $\bar{p} = (\bar{R}, \bar{C})$) for A under the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$ such that

$$\sum_{i\in R} \alpha_i + \sum_{j\in C} \beta_j + \sum_{i\in \bar{R}} \bar{\alpha}_i + \sum_{j\in \bar{C}} \bar{\beta}_j$$

is maximized.

We introduce a binary variable y_i^R for every row $i \in [m]$ that attains value one if and only if $i \in R$, and analogously, a binary variable y_j^C for every column $j \in [n]$ that attains value one if and only if $j \in C$. Moreover, we add a binary variables x_i^R for every row and x_j^C for every column $j \in [n]$ that attains value one if and only if $i \in \bar{R}$ and $j \in \bar{C}_j$, unless $\bar{\alpha}_i \neq 0$ and $\bar{\beta}_j \neq 0$, respectively. If $\bar{\alpha}_i = 0$ (or $\bar{\beta}_j = 0$) for some $i \in [m](,j \in [n])$, then it does not matter whether i (or j, respectively,) is a neighbor row (or neighbor column).

$$\begin{array}{llll}
 & \text{Maximize} & \displaystyle\sum_{i=1}^{m} \alpha_i y_i^R + \sum_{j=1}^{n} \beta_j y_j^C + \sum_{i=1}^{m} \bar{\alpha}_i x_i^R + \sum_{j=1}^{n} \bar{\beta}_j x_j^C & \\
 & \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i^R \geq \ell^{\mathcal{R}}; & (PR\ell) \\
 & & \displaystyle\sum_{j=1}^{n} y_j^C \geq \ell^{\mathcal{C}}; & (PC\ell) \\
IP_P & & \displaystyle\sum_{i=1}^{m} y_i^R \leq u^{\mathcal{R}}; & (PRu) \\
 & & \displaystyle\sum_{j=1}^{n} y_j^C \leq u^{\mathcal{C}}; & (PCu) \\
 & & y_i^R - y_j^C - x_j^C \leq 0, \quad \text{for } i \in [m], j \in [n] \text{ with } a_{ij} \neq 0; & (PRN) \\
 & & y_j^C - y_i^R - x_i^R \leq 0, \quad \text{for } i \in [m], j \in [n] \text{ with } a_{ij} \neq 0; & (PRC) \\
 & & y_i^R, y_j^C, x_i^R, x_j^C \in \{0,1\}, \quad \text{for } i \in [m], j \in [n]. & (PB)
\end{array}$$

It easily can be seen that IP_P solve the MCBPS problem. According to the constraints $(PR\ell)$, $(PC\ell)$, (PRu) and (PCu), the obtained block pattern is feasible under the load condition $(\ell^{\mathcal{R}}, u^{\mathcal{R}}, \ell^{\mathcal{C}}, u^{\mathcal{C}})$. The constraints (PRN) ensure that if row $i \in [m]$ is assigned to the block pattern $p = (R, C)$, then for every column $j \in [n]$ with $a_{ij} \neq 0$, j is assigned to C or $\bar{C}$. Analogously, the constraints (PCN) ensure that if column $j \in [n]$ is assigned the block pattern $p = (R, C)$, then for every row $i \in [m]$ with $a_{ij} \neq 0$, i is assigned to R or $\bar{R}$. Notice that in an optimal solution of IP_P the constraints (PRN) guarantee that if $y_i^R = 1$ for some $i \in [m]$, then for every column $j \in [n]$ with $a_{ij} \neq 0$ holds either $x_j^C = 1$ or $y_j^C = 1$, unless $\bar{\beta} = 0$. Analogously, the constraints (PRC) guarantee that if $y_j^C = 1$ for some $j \in [n]$, then for every row $i \in [m]$ with $a_{ij} \neq 0$ holds either $x_i^R = 1$ or $y_i^R = 1$, unless $\bar{\alpha} = 0$. Hence, one can obtain an optimal solution for MCBPS from an optimal solution of IP_P and thus solve the MCBPS problem.

In the next chapter we will examine in Section 6.2.4 whether the LP-relaxation of IP_{CG} provides a better lower bound than the trivial one obtained by IP_A.

6 Computational Experiments

In this chapter we are going to present the results of the computational experiments that we obtained with the matrix decomposition tool *Decomp* that was implemented in the course of this thesis. After giving some information about the testing machine and our implementation, we illustrate the results of our experiments for the heuristic methods in Section 6.1. Finally, we show our computational results for the exact methods.

Implementational issues

We have implemented the matrix decomposition tool *Decomp* that is capable of parsing the coefficient matrix of a mixed integer program given in MPS or LP file format and solving the problems MinBf and MinAf on this matrix. *Decomp* was implemented over some months and contains roughly 18000 lines of code. The visualization of the decomposed matrices uses *gnuplot* 4.4.

Machine

All tests were executed on a machine using one Intel(R) Pentium(R) 4 CPU with a clock speed of 3.20GHz. The 32 bit CPU was supported by 1 GB of RAM. A Linux 2.6.34.10 kernel was running on the machine and all software was compiled using a GCC 4.3 compiler.

6.1 Results for Heuristic Methods

In this section we present our obtained computational results for the algorithms introduced in Chapter 4. At first, we want to give a brief introduction to Metis and hMetis, the graph partitioning software that we will use to solve the HES problem. Secondly, we summarize all parameters. Afterwards, an overview about our test instances is given. Our first goal is to determine the best parameter settings by testing on 3 different test sets. The found settings will be applied to compare our results with the results obtained by Ferris and Horn[17]. Moreover, we apply these settings to find decompostions on matrices from a large mixed integer programming problem library, namely the MIPLIB 2010[30].

Metis

The solving of the HES problem was treated as 'black box' in our theoretical examinations in Chapter 4. In our implementation we used the methods that are provided by *Metis* 4.0.1 [27, 26] and *hMetis* 1.5.3 [29, 25] to solve HES on graphs and hypergraphs, respectively. These are for Metis the methods *METIS_PartGraphRecursive* and *METIS_PartGraphKway*, and for hMetis the methods *HMETIS_PartRecursive* and *HMETIS_PartKway*. We used the default settings of METIS. For hMetis we also used the default settings with two exceptions: For the method HMETIS_PartRecursive the parameter *UBfactor* is set to 2 and for the method HMETIS_PartKway it is set to 5, for both methods the parameter *nRuns* is set to 5. For a deeper discussion of the methods METIS_PartGraphRecursive [27], METIS_PartGraphKway[26], HMETIS_PartKway [29] and HMETIS_PartRecursive [25] we refer the reader to the indicated literature and the manuals of Metis and hMetis. For the sake of simplicity, we will denote the methods METIS_PartGraphRecursive and HMETIS_PartRecursive by *RECURSIVE*. It will be clear which one we use, since graphs are only used in the bipartite decomposing algorithm. For the same reason, we denote the methods METIS_PartGraphKway and HMETIS_PartKway by *KWAY*. We introduce the parameter *metisMethod* that is either RECURSIVE or KWAY.

Dummy Nodes

The tests of Ferris and Horn[17] suggest that it may be profitable to add *dummy nodes* to the graph. Dummy nodes are not adjacent to other nodes. Furthermore, they are deleted after obtaining the partition. This may leave some parts of the partition empty. However, we also test this approach by optionally adding $0.2N$ dummy nodes with N the number of vertices of the graph. We introduce the parameter *dummyRatio* that is either '0%' if no dummy nodes are added, or '20%' if $0.2N$ dummy nodes are added.

6.1.1 Parameters

We have introduced two general parameters so far: The parameter *dummyRatio* and the parameter *metisMethod*. In addition to that, we can choose between two weighting schemes. (As seen in Section 4.3.1 we do not have this choice for the hyperrow decomposing algorithm). To be more precise,

- if we apply the hypercolum or the hypercolrow algorithm, we can choose either the unary (un) or the prop size (ps) weighting scheme.
- If we apply the bipartite algorithm, we can choose either the unary (un) or the aprop degree (ad) weighting scheme.
- If we use the hyperrow algorithm, there is no choice and we have to use the unary (un) weighting scheme.

In this way, we will use the parameter *weightingScheme* that can attain one of the values *un*, *ps* and *ad*.

6.1.2 Instances

In this subsection we want to give some information about our test instances. We introduce our parameter test set that we have used to find promising settings of our parameters. In order to determine good parameter settings, we will test our algorithms on the three following test sets from the Miplib 2003 [3]. The first set consists of rather small instances whose number of nonzero entries is smaller than 10000. It is displayed in Table 6.1c. Furthermore, we introduce the medium-size instances whose number of nonzeros is between 10000 and 30000, and a set of big instances with 30000 or more nonzero entries displayed in Table 6.1b and Table 6.1a, respectively.

instance	#rows	#edges	#nonzeros
air04	823	8904	72965
air05	426	7195	52121
cap6000	2176	6000	48243
dano3mip	3202	13873	79655
disctom	399	10000	30000
msc98-ip	15850	21143	92918
net12	14021	14115	80384
seymour	4944	1372	33549
swath	884	6805	34965

(a) big-size instances from MIPLIB 2003

instance	#rows	#edges	#nonzeros
a1c1s1	3312	3648	10178
arki001	1048	1388	20439
liu	2178	1156	10626
manna81	6480	3321	12960
mkc	3411	5325	17038
mod011	4480	10958	22254
protfold	2112	1835	23491
roll3000	2295	1166	29386

(b) medium-size instances from MIPLIB 2003

instance	#rows	#columns	# nonzeros
aflow30a	479	842	2091
aflow40b	1442	2728	6783
danoint	664	521	3232
fiber	363	1298	2944
fixnet6	478	878	1756
gesa2	1392	1224	5064
gesa2-o	1248	1224	3672
glass4	396	322	1815
harp2	112	2993	5840
modglob	291	422	968
noswot	182	128	735
opt1217	64	769	1542
p2756	755	2756	8937
pk1	45	86	915
pp08a	136	240	480
pp08aCUTS	246	240	839
qiu	1192	840	3432
timtab1	171	397	829
timtab2	294	675	1482
tr12-30	750	1080	2508
vpm2	234	378	917

(c) small-size instances from MIPLIB 2003

Table 6.1: Instances from Miplib2003

In the following subsections we want to determine the best algorithm-setting combination to solve the problems MinAf and MinBf. Instead of comparing the objective function values, we will compare the values of the three quality measure. We have seen in Section 2.4 that we can consider the value of μ_{boN} in order to compare the solutions by their objective function value. We start with the MinAf problem. Every test run actually consists of 5 runs of the current algorithm-setting combination. The decomposition with the highest measure is chosen.

6.1.3 MinAf

In this subsection we present our results concerning the heuristic algorithms designed to solve the MinAf problem. These are the hypercolrow decomposing algorithm and the bipartite decomposing algorithm. We compare the algorithm-settings combinations with each other measure-wise starting with the border number measure. In order to describe our testing policy, we show the complete results for the medium-size instances for the border number measure in Table 8.2. The complete results for the small-size instances and the big-size instances can be found in Table 8.1 and Table 8.3, respectively. The table is organized as follows: Every column (except the first 2) stands for an algorithm-settings combination and every row (except the first 5) stands for a combination of matrix and number of blocks. We have tested three different block numbers, namely 4, 16 and $b(A)$ with $b(A) = \min(m, n)^{0.3}$ for a matrix $A \in \mathbb{R}^{m \times n}$. The table shows the best value of five test runs. If no feasible solution could be found, the respective entry is empty and the value is treated as 0.

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
10teams	4	0.90	0.94						0.93	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
	16																
	5	0.91								0.94	0.94	0.94	0.94	0.94	0.94	0.95	0.95
a1c1s1	4	0.96	0.95	0.96	0.95	0.97	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.88	0.93	0.89	0.93	0.89	0.93	0.89	0.92	0.96	0.92	0.96	0.92	0.96	0.89	0.95	0.90
	11	0.90	0.93	0.90	0.93	0.90	0.94	0.90	0.93	0.96	0.92	0.96	0.92	0.96	0.91	0.96	0.92
arki001	4	0.96	0.97	0.97	0.97	0.96	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.94	0.94	0.94	0.94	0.89	0.93	0.94	0.95	0.95	0.95	0.95	0.95	0.96	0.96	0.96	0.96
	8	0.93	0.94	0.93	0.93	0.92	0.94	0.98	0.98	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
liu	4	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	8	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
manna81	4	0.95	0.99	0.91	0.98	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.87	0.99	0.86	0.98	0.97	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	11	0.91	0.99	0.88	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
mkc	4	0.97	0.99	0.97	0.99	0.95	0.99	0.96	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.97				0.95	0.99	0.95	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
	11	0.96	0.99	0.97	0.99	0.96	0.99	0.96	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
mod011	4	0.99	0.99	0.98	0.99	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.96	0.97	0.95	0.97	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	12	0.97	0.98	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1.00	1.00	1.00	0.99
protfold	4	0.72	0.75	0.76	0.75	0.73	0.72	0.72	0.73	0.83	0.76	0.83	0.77	0.84	0.76	0.83	0.75
	16	0.59	0.66	0.61	0.65	0.58	0.62	0.57	0.63	0.74	0.68	0.74	0.73	0.75	0.58	0.76	0.59
	9	0.64	0.67	0.64	0.69	0.60	0.66	0.61	0.65	0.79	0.73	0.78	0.76	0.79	0.63	0.80	0.63
roll3000	4	0.93	0.94	0.93	0.94	0.92	0.93	0.91	0.94	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.88
	16	0.88	0.87	0.87	0.86	0.86	0.86	0.88	0.88	0.85	0.85	0.85	0.86	0.86	0.86	0.86	0.86
	8	0.93	0.93	0.91	0.93	0.91	0.92	0.90	0.93	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88
arithm.mean		0.87	0.82	0.77	0.79	0.81	0.82	0.81	0.86	0.91	0.90	0.91	0.90	0.91	0.89	0.91	0.89
quadr.mean		0.89	0.88	0.84	0.86	0.86	0.88	0.87	0.90	0.93	0.92	0.93	0.92	0.93	0.91	0.93	0.91

Table 6.2: Results for medium inst. to arrowhead conc. μ_{boN}

The aggregated results of our tests concerning the border number measure μ_{boN} are stated in Table 6.3. We have decided to consider the quadratic mean values because the failed runs would have too much impact if we considered the arithmetic mean values. For the sake of simplicity, we aggregate the aggregated values by using the arithmetic mean of the quadratic mean values.

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
MetisMethod		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un	ad	un	ad	un	ad	un	ad	un	ps	un	ps	un	ps	un	ps
Testset																	
small	quadr.mean	0.86	0.90	0.85	0.89	0.87	0.89	0.84	0.90	0.93	0.93	0.94	0.94	0.94	0.93	0.94	0.94
medium	quadr.mean	0.89	0.88	0.84	0.86	0.86	0.88	0.87	0.90	0.93	0.92	0.93	0.92	0.93	0.91	0.93	0.91
big	quadr.mean	0.81	0.83	0.75	0.87	0.81	0.88	0.80	0.84	0.92	0.94	0.94	0.94	0.91	0.94	0.89	0.92
total	arithm.mean	0.85	0.87	0.81	0.87	0.85	0.88	0.83	0.88	0.94	0.93	0.94	0.93	0.93	0.93	0.92	0.92

Table 6.3: aggregated results for μ_{boN}

One can see that in general, the hypercolrow decomposing algorithm yields better decompositions in terms of the border number measure than the bipartite decomposing algorithm. In particular, the settings with $metisMethod = PKW$ and $weightingScheme = un$ yield decompositions with best aggregated values.

Now, we want to study the aggregated results for the border area measure μ_{boA} indicated in Table 6.4. The complete results can be found in the Tables 8.4, 8.5 and 8.6 in the appendix. We aggregate the aggregated values again for the sake of simplicity.

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
MetisMethod		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un	ad	un	ad	un	ad	un	ad	un	ps	un	ps	un	ps	un	ps
Testset																	
small	quadr.mean	0.77	0.82	0.77	0.81	0.79	0.83	0.77	0.83	0.87	0.86	0.87	0.86	0.87	0.86	0.87	0.86
medium	quadr.mean	0.78	0.80	0.76	0.80	0.79	0.82	0.80	0.83	0.84	0.83	0.84	0.83	0.85	0.83	0.85	0.83
big	quadr.mean	0.63	0.65	0.56	0.65	0.67	0.68	0.67	0.68	0.71	0.72	0.71	0.72	0.72	0.72	0.71	0.72
total	arithm.mean	0.73	0.76	0.70	0.75	0.75	0.78	0.75	0.78	0.81	0.80	0.81	0.80	0.81	0.80	0.81	0.80

Table 6.4: aggregated results for μ_{boA}

The aggregated values show again that the hypercolrow decomposing algorithm obtains better decompositions in terms of the border area measure than the bipartite decomposing algorithm in general. Especially, the settings with $weightingScheme = un$ yield decompositions with best aggregated values.

Finally, we examine the aggregated results for the block balance measure μ_{blB} presented in Table 6.5. Again, the complete results can be found in the appendix in the Tables 8.7, 8.8 and 8.9.

The results show again that the hypercolrow decomposing algorithm finds better decompositions in terms of the block balance measure than the bipartite decomposing algorithm. In particular, the settings with $metisMethod = PKW$, $weightingScheme = un$, and $dummyRatio = 20\%$ and the settings with $weightingScheme = ad$ finds decompositions with best aggregated values in terms of the block balance measure.

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
MetisMethod		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un	ad	un	ad	un	ad	un	ad	un	ad	un	ad	un	ad	un	ad
Testset																	
small	quadr.mean	0.79	0.80	0.76	0.78	0.82	0.85	0.73	0.77	0.90	0.89	0.89	0.89	0.89	0.89	0.87	0.87
medium	quadr.mean	0.75	0.77	0.71	0.72	0.75	0.78	0.67	0.69	0.85	0.85	0.85	0.84	0.84	0.84	0.84	0.84
big	quadr.mean	0.43	0.49	0.42	0.50	0.54	0.66	0.47	0.55	0.76	0.78	0.78	0.79	0.74	0.78	0.73	0.76
total	arithm.mean	0.65	0.69	0.63	0.67	0.71	0.76	0.63	0.67	0.83	0.84	0.84	0.84	0.82	0.84	0.82	0.84

Table 6.5: aggregated results for μ_{blB}

Conclusion

We have seen that the hypercolrow decomposing algorithm obtains better decompositions in terms of all three block measures. We believe that one reason for that is that the HVS problem is solved indirectly and that a more elaborate approach to solve HVS would yield better decompositions. However, there is one algorithm-setting combination that yield decomposition with best aggregated values for all measures. This is the hypercolrow algorithm with $\mathit{metisMethod} = PKW$, $\mathit{weightingScheme} = \mathit{un}$ and $\mathit{dummyRatio} = 20\%$. Therefore, throughout the remaining tests, we will use this setting whenever we solve MinAf.

6.1.4 MinBf

In this subsection we state and compare the results for the heuristic algorithms designed to solve the MinBf problem, these are the hyperrow decomposing algorithm and the hypercol decomposing algorithm. The structure is the same as in the last subsection: We compare the algorithm-setting combinations for each measure starting with the border number measure. The tables are organized in the same way as above: Every column (except the first 2) stands for an algorithm-settings combination and every row (except the first 5) stands for a combination of matrix and number of blocks.

The aggregated results of our tests with respect to the border number measure μ_{boN} are stated in Table 6.6. The complete test results can be found in the Tables 8.10, 8.11 and 8.10 in the appendix.

Algorithm		HYPERROW DEC.				HYPERCOL DECOMPOSING							
Metis method		PKW		REC.		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un	un	un	un	un	ps	un	ps	un	ps	un	ps
Testset													
small	quadr.mean	0.89	0.90	0.88	0.89	0.85	0.86	0.84	0.84	0.87	0.85	0.82	0.83
medium	quadr.mean	0.86	0.85	0.86	0.85	0.73	0.75	0.74	0.75	0.74	0.74	0.74	0.73
big	quadr.mean	0.86	0.87	0.87	0.85	0.66	0.63	0.67	0.68	0.66	0.70	0.66	0.66
total	arithm.mean	0.87	0.87	0.87	0.87	0.75	0.75	0.75	0.75	0.76	0.76	0.74	0.74

Table 6.6: Aggregated results for solving MinBf in terms of μ_{boN}

One can see that in general, the hyperrow decomposing algorithm yields better decompositions in terms of the border number measure than the hypercolumn decomposing algorithm. In particular, all settings for the hyperrow decomposing algorithm yield decompositions with best aggregated values.

Now we want to study the aggregated results for the border area measure μ_{boA} stated in Table 6.7. For the sake of simplicity, we aggregate the aggregated values. The complete results can be found in the Tables 8.13, 8.14 and 8.15 in the appendix.

Algorithm		HYPERROW DEC.				HYPERCOL DECOMPOSING							
Metis method		PKW		REC.		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un	un	un	un	un	ps	un	ps	un	ps	un	ps
Testset													
small	quadr.mean	0.80	0.81	0.80	0.81	0.74	0.74	0.73	0.73	0.74	0.73	0.72	0.72
medium	quadr.mean	0.77	0.77	0.78	0.78	0.67	0.68	0.68	0.68	0.66	0.66	0.67	0.65
big	quadr.mean	0.58	0.60	0.61	0.64	0.46	0.46	0.49	0.49	0.47	0.48	0.48	0.49
total	arithm.mean	0.72	0.73	0.73	0.74	0.63	0.63	0.63	0.63	0.63	0.63	0.62	0.62

Table 6.7: aggregated results for μ_{boA}

The aggregated values show that the hyperrow decomposing algorithm yields better decompositions in terms of the border area measure than the hypercolumn decomposing algorithm. Especially, the settings with *dummyRatio* $= 20\%$ for the hyperrow decomposing algorithm yield decompositions with best aggregated values.

Finally, we examine the aggregated results for the block balance measure μ_{blB} indicated in Table 6.8. Again, the complete results can be found in the appendix in the Tables 8.16, 8.16 and 8.16.

Algorithm		HYPERROW DEC.				HYPERCOL DECOMPOSING							
Metis method		PKW		REC.		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un	un	un	un	un	ps	un	ps	un	ps	un	ps
Testset													
small	quadr.mean	0.84	0.81	0.84	0.80	0.76	0.76	0.69	0.67	0.73	0.72	0.65	0.61
medium	quadr.mean	0.82	0.77	0.82	0.78	0.63	0.62	0.60	0.61	0.60	0.59	0.57	0.56
big	quadr.mean	0.69	0.65	0.61	0.58	0.41	0.40	0.37	0.39	0.37	0.38	0.30	0.32
total	arithm.mean	0.78	0.74	0.76	0.72	0.60	0.60	0.55	0.56	0.57	0.56	0.51	0.49

Table 6.8: aggregated results for μ_{blB}

One can see again that the hyperrow decomposing algorithm yields better decompositions in terms of the block balance measure than the hypercolumn decomposing algorithm. In particular, the setting with *dummyRatio* $= 0\%$ and *metisMethod* $= PKW$ for the hyperrow decomposing algorithm yield decompositions with best aggregated values.

Conclusion

We have seen that the hyperrow decomposing algorithm obtains better decompositions in terms of all three block measures. We guess, like for MinAf, that one reason for this is that the HVS problem is solved indirectly and that a more elaborate approach to solve HVS would yield better decompositions. However, there are several algorithm-setting combinations that yield decompositions with almost best aggregated values for all measures, for instance, the hyperrow algorithm with $metisMethod = PKW$, $weightingScheme = un$ and $dummyRatio = 20\%$. Hence, throughout the remaining tests, we will use this setting whenever we solve MinBf.

6.1.5 Comparison to Ferris and Horn's Results

Ferris and Horn [17] developed an approach to solve MinAf. They suggest the quality measure $\mu^* := 0.1\mu_{blB} + 0.9\mu_{boA}$. In the following we call this measure the *star measure*. One disadvandage of their stated results is that they allow blocks to be empty and do note indicate how many blocks the decompositions found by their approach really have. They only present the number of requested blocks and, in particular, found a decomposition for the instance 'afiro' that has 27 rows while actually requesting 64 blocks. From a theoretical point of view one could define the whole matrix to be one single block leaving all other blocks empty. This decomposition would have measure value of one for all three measures. However, in order to compare our best algorithm-setting combination with their algorithm, we follow them and make this concession. The complete comparison, including 89 instances, can be found in Table 8.19 and Table 8.20 in the appendix. Here we will show an excerpt from these tests in Table 6.9 and, of course, indicate the aggregated values in Table 6.10.

The structure of the table is the following: The first four columns include information about the instance. The remaining columns contain information about the found decompositions. There are 8 main columns, one for each $k \in \{2, 4, 8, 16, 32, 64, 128, 256\}$ with k is the number of requested blocks. Each of the 8 main columns contain 3 subcolumns. The first one indicates by its color which algorithm found a better decomposition. If Ferris and Horn found the better one, the color is red. If *Decomp* found the better one, it is green. If no decomposition could be found by both algorithms, the first subcolumn is empty. The next two subcolumns show the star measure value of the found decomposition.

number of blocks requested				2			4			8			16			32			64			128			256		
instance	#rows	#cols	#nonz		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.
25fv47	821	1571	10400		0.97	0.91		0.88	0.855		0.65	0.762		0.73	0.611		0.69	0.524		0.58	0.473		0.45	0.369		0.27	0.285
80bau3b	2262	9799	21002		0.78	0.937		0.62	0.91		0.6	0.88		0.59	0.851		0.56	0.816		0.52	0.77		0.43	0.719		0.41	0.642
adlittle	56	97	383		0.95	0.83		0.59	0.74		0.5	0.633		0.47	0.573		0.35	0.421		0.18	0.273		0.1	0.11		0.03	0.033
afiro	27	32	83		1	0.824		0.77	0.754		0.56	0.616		0.51	0.43		0.33	0.362		0.16	0.166		0	0		0	0
agg	488	163	2410		0.8	0.886		0.74	0.734		0.65	0.474		0.44	0.321		0.27	0.255		0.19	0.202		0.19	0.197		0.18	0.178
agg2	516	302	4284		1	0.852		0.89	0.69		0.71	0.54		0.76	0.48		0.59	0.38		0.53	0.302		0.48	0.25		0.45	0.18
agg3	516	302	4300		1	0.865		1	0.694		0.83	0.534		0.78	0.471		0.66	0.38		0.57	0.318		0.53	0.233		0.5	0.179
bandm	305	472	2494		0.91	0.938		0.77	0.831		0.71	0.782		0.66	0.712		0.58	0.566		0.43	0.516		0.34	0.443		0.34	0.379
beaconfd	173	262	3375		0.73	0.796		0.49	0.742		0.48	0.717		0.46	0.689		0.44	0.636		0.44	0.471		0.38	0.341		0.28	0.278
blend	74	83	491		0.75	0.831		0.75	0.706		0.62	0.622		0.46	0.411		0.29	0.343		0.18	0.251		0	0.13		0	0.075
bnl1	643	1175	5121		0.84	0.906		0.75	0.847		0.69	0.802		0.62	0.746		0.58	0.668		0.52	0.63		0.48	0.527		0.37	0.453
bnl2	2324	3489	13999		0.87	0.928		0.83	0.866		0.78	0.83		0.71	0.787		0.66	0.75		0.6	0.713		0.58	0.673		0.52	0.614
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 6.9: Excerpt from comparison with results of Ferris and Horn

number of blocks requested	2			4			8			16			32			64			128			256		
		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.
quadratic mean		0.87	0.89		0.76	0.82		0.67	0.75		0.59	0.67		0.52	0.61		0.45	0.55		0.39	0.49		0.33	0.42

Table 6.10: Comparison to the results of Ferris and Horn (aggregated)

By taking the arithmetic mean of these values we obtain a value of 0.5725 for the decompositions found by the algorithm of Ferris and Horn, and a value of 0.65 for *Decomp*. Hence, in terms of the star measure for aggregated values, *Decomp* is better by 13.5%. Overall, Ferris and Horn' algorithm found a better decomposition for 221 instances and *Decomp* found a better decomposition for 485 instances.

Conclusion

We have seen that the hypercolrow algorithm performs better than the algorithm of Ferris and Horn, at least under the conditions used by Ferris and Horn. Unfortunately, on the one hand, these conditions make the results rather less significant, and on the other hand, they used older graph partitioning software, namely Metis 2.0.

In the following, we give results that are easier to compare since empty blocks are forbidden.

6.1.6 Performance with Forbidden Empty Blocks

In this subsection, we want to study if our algorithms find decompositions to arrowhead and bordered block diagonal form for coefficient matrices of state of the art mixed integer programs with forbidden empty blocks. In order to do so, we have tested our best algorithm-setting combinations for each instance of the 201 instances from Miplib2010 [30] that has less than 100000 nonzero entries. Again, we have tested for 8 different block numbers k with $k \in \{2, 4, 8, 16, 32, 64, 128, 256\}$ but this time empty blocks are forbidden.

The complete results for the hypercolrow decomposing algorithm that should solve the MinAf problem can be found in the Tables 8.21, 8.22 and 8.23 in the appendix. These tables are structured in a similar way as in the last subsection. The first four columns include information about the matrix, and the remaining 8 columns contain the star measure value of the obtained decomposition or are empty if no decomposition could be found. Moreover, the complete results for the hyperrow algorithm that approaches the MinBf problem are indicated in the Tables 8.24, 8.25 and 8.26 in the appendix.

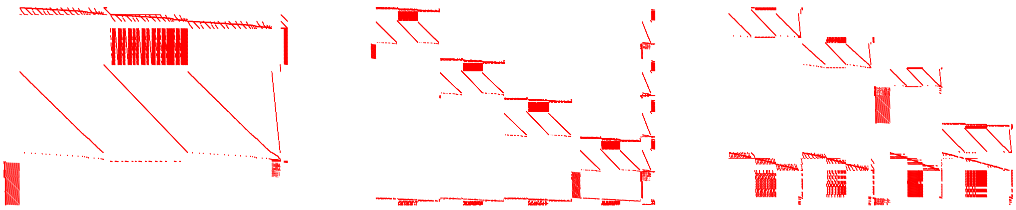
The aggregated values of the test runs for MinAf and MinBf on all 201 instances are shown in Table 6.11.

As expected, the aggregated values suggest that the quality of the decompositions to bordered block diagonal form is worse than those that are in arrowhead form. Furthermore, one can see that the more blocks are requested the quality of the decompositions decreases. It seems that the quality for decompositions to bordered block diagonal form decreases faster than for arrowhead form.

number of blocks	2	4	8	16	32	64	128	256
quadr. mean of μ^* for MinAf	0.86	0.82	0.78	0.75	0.70	0.65	0.59	0.51
quadr. mean of μ^* for MinBf	0.85	0.79	0.73	0.68	0.63	0.52	0.45	0.36

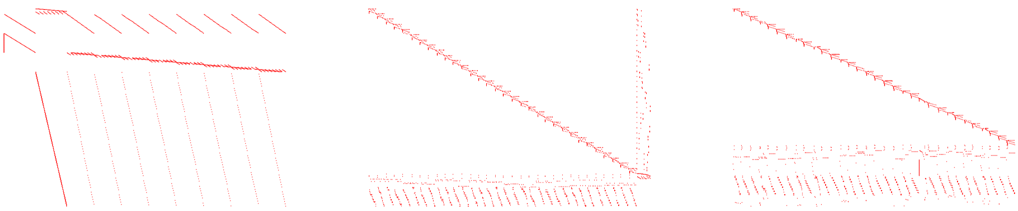
Table 6.11: Results for miplib2010 (aggregated)

Finally, we present some decomposed matrices from the Miplib2010 in arrowhead and in bordered block diagonal form.



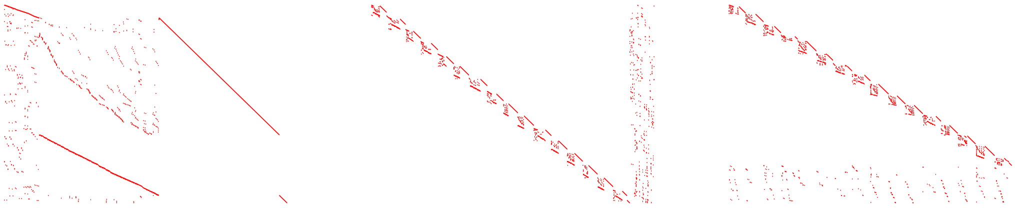
(a) original (b) 4-arrowhead form (c) bordered 4-block diagonal form

Figure 6.1: Coefficient matrix of satellites1-25



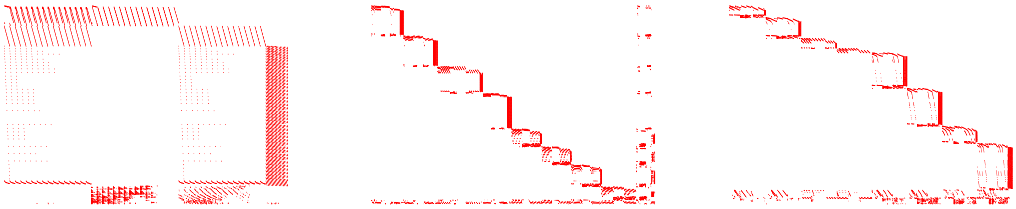
(a) original (b) 32-arrowhead form (c) bordered 32-block diagonal form

Figure 6.2: Coefficient matrix of bienst2



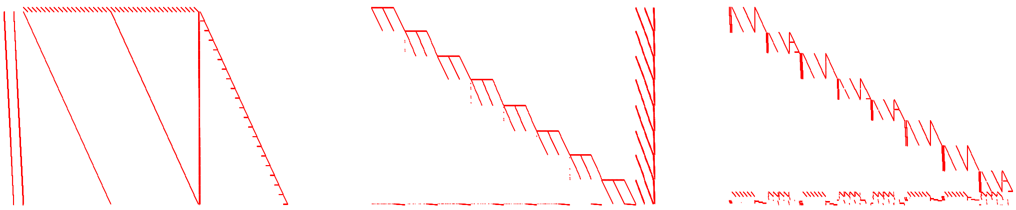
(a) original (b) 16-arrowhead form (c) bordered 16-block diagonal form

Figure 6.3: Coefficient matrix of ic97_potential



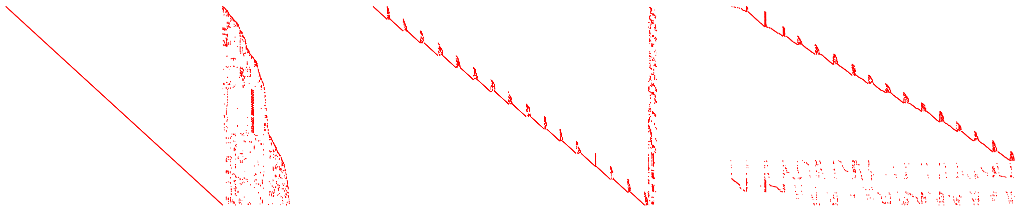
(a) original (b) 8-arrowhead form (c) bordered 8-block diagonal form

Figure 6.4: Coefficient matrix of dg012142



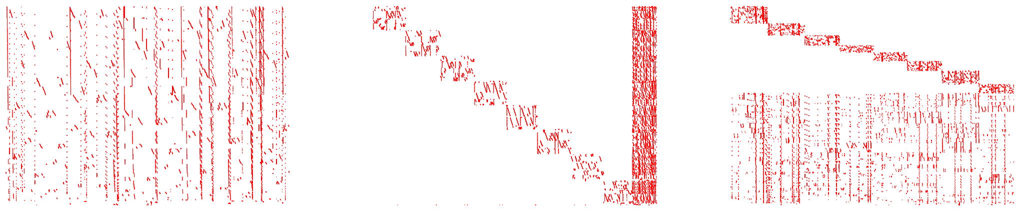
(a) original (b) 8-arrowhead form (c) bordered 8-block diagonal form

Figure 6.5: Coefficient matrix of atm20-100



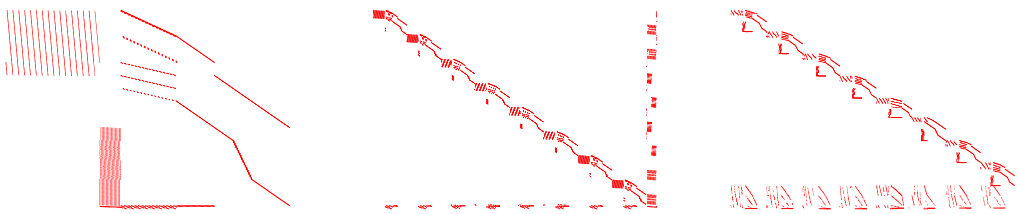
(a) original (b) 16-arrowhead form (c) 16-bordered block diagonal form

Figure 6.6: Coefficient matrix of toll-like



(a) original (b) 8-arrowhead form (c) bordered 8-block diagonal form

Figure 6.7: Coefficient matrix of nag



(a) original (b) 8-arrowhead form (c) bordered 8-block diagonal form

Figure 6.8: Coefficient matrix of b2c1s1

6.2 Results for Exact Methods

In the following section we give the computational results of our implemented exact approach. This is the integer program IP_A introduced in Section 5.2. We have implemented this integer program in *SCIP*.

6.2.1 Scip

SCIP [2] is a framework developed for solving *constraint integer programs*. Constraint integer programming is a generalization of mixed integer programming (see Subsection 2.2.4). It provides the fundamental parts to implement branch-and-bound based search algorithms. Moreover, the user can replace these parts with own implementations (plugins). We use *SCIP* with an extern linear programming solver, namely *CPLEX* 12.1.0 [1].

6.2.2 Instances

We test our approach on a set of very small instances indicated in Table 6.12.

instance	#rows	#edges	#nonZeros	origin
bell3a	123	133	347	MipLib 3.0
bell5	91	104	266	MipLib 3.0
bm23	20	27	478	MipLib 2.0
egout	98	141	282	MipLib 3.0
enigma	21	100	289	MipLib 3.0
fixnet3	478	878	1756	MipLib 2.0
flugpl	18	18	46	MipLib 3.0
gt2	29	188	376	MipLib 3.0
khb05250	101	1350	2700	MipLib 3.0
lseu	28	89	309	MipLib 3.0
markshare1	6	62	312	MipLib2003
markshare2	7	74	434	MipLib2003
misc01	54	83	745	MipLib 2.0
mod008	6	319	1243	MipLib 3.0
neos858960	132	160	2770	MipLib2010
noswot	182	128	735	MipLib2010
p0033	16	33	98	MipLib 3.0
p0040	23	40	110	MipLib 2.0
pipex	25	48	192	MipLib 2.0
pk1	45	86	915	MipLib2003
pp08a	136	240	480	MipLib2003
rgn	24	180	460	MipLib 3.0
sample2	45	67	146	MipLib 2.0
stein9	13	9	45	MipLib 2.0
stein15	36	15	120	MipLib 2.0
stein27	118	27	378	MipLib 3.0
stein45	331	45	1034	MipLib 3.0
timtab1	171	397	829	MipLib2010
vpm1	234	378	749	MipLib 3.0

Table 6.12: Test instances for exact approaches

6.2.3 Results for the Assignment Approach

In order to test the approach for different load conditions, we introduce the parameter $lc \in \{LO, ME, TI\}$. This paramter should achieve that the load condition can be rather loose, medium or tight, respectively. Its influence on the load condition of the instance is illustrated in Table 6.13. The values *avr* and *avc* are the average number of rows per block and the average number of columns per block, respectively.

lc	$\ell^{\mathcal{R}}$	$\ell^{\mathcal{C}}$	$u^{\mathcal{R}}$	$u^{\mathcal{C}}$
LO	1	1	m	n
ME	$0.5avr$	$0.5avc$	$1.5avr$	$1.5avc$
TI	$0.9avr$	$0.9avc$	$1.1avr$	$1.1avc$

Table 6.13: Table for parameter lc

We tested the three integer programs IP_A, IP_{AR} and IP_{AC} introduced in Section 5.2 for the block numbers 2 and 4. The complete test results for 2 and 4 blocks can be found in Table 8.27 and Table 8.28, respectively, in the appendix. In order to describe the structure of the tables, we give a short excerpt in Table 6.14. All test runs have a timelimit of 1800 seconds.

		IP_A			IP_{AR}			IP_{AC}		
Instance	lc	gap	nNodes	time	gap	nNodes	time	gap	nNodes	time
"bell3a"	LO	0%	2	2.52	0%	1	0.44	0%	1	0.49
	ME	0%	19	8.18	0%	15	8.96	0%	9	4.27
	TI	0%	73	18.88	0%	46	21.76	0%	58	19.85
"bell5"	LO	0%	2	1.83	0%	1	0.42	0%	1	1.59
	ME	0%	3	2.41	0%	12	4.55	0%	21	5.49
	TI	0%	6	2.72	0%	21	6.71	0%	23	7.27
"bm23"	LO	0%	462	15.64	0%	345	17.19	0%	109	15.18
	ME	infeas.	1	0.06	infeas.	1	0.08	infeas.	1	0.07
	TI	infeas.	1	0.07	infeas.	1	0.04	infeas.	1	0.06
"egout"	LO	0%	13	4.66	0%	7	2.7	0%	12	5.57
	ME	0%	14	4.8	0%	9	4.45	0%	11	5.32
	TI	0%	10	3.93	0%	13	5.13	0%	12	8.24
"enigma"	LO	0%	2913	20.86	0%	54	8.26	0%	804	13.14
	ME	0%	37	6.05	0%	27	3.96	0%	26	5.65
	TI	0%	17	5.89	0%	21	6.73	0%	25	9.89
"fixnet3"	LO	0%	1	42.41	0%	1	27.67	0%	1	48.87
	ME	0%	1939	649.17	0%	1638	885.26	0%	5816	1733.59
	TI	0%	2564	1151.11	0%	1119	498.19	0%	2278	734.75
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 6.14: Results for exact solving (2Blocks)

Every row stands for test run including a matrix and a load condition. There are three main columns containing the information about the results for one integer program. Each main column contains three subcolumns including information about the optimality gap in percent, the number of nodes and the needed time. The gap column may contain “infeas.” then *SCIP* found out that the instance is infeasible. In order to compare the performance of the three integer programms we aggregate the values over all instance-load condition combinations by calculating the geometric and arithmetic mean. The aggregated numbers for 2 and 4 blocks are indicated in Table 6.15 and Table 6.16, respectively. Additionally, we indicate the best gain that could achieve for the geometric mean.

	IP_A		IP_{AR}		IP_{AC}		best gain	
	nNodes	time	nNodes	time	nNodes	time	nNodes	time
geom. mean	47.10	6.64	35.49	5.77	32.35	5.79	31.3%	13.1%
arithm. mean	9490.15	211.81	4884.01	172.33	6239.78	182.82		

Table 6.15: Aggregated results for exact approach (2 blocks)

	IP_A		IP_{AR}		IP_{AC}		best gain	
	nNodes	time	nNodes	time	nNodes	time	nNodes	time
geom. mean	303.91	151.18	250.14	134.49	210.63	141.39	30.5%	18.6%
arithm. mean	4755.66	770.12	3038.63	723.33	2737.51	749.10		

Table 6.16: Aggregated results for exact approach (4 blocks)

One can see that for both block numbers the integer programs IP_{AR} and IP_{AC} achieve better aggregated values in terms of the number of needed nodes and in terms of needed time. Hence, both kind of constraints yield a significant improvement of the performance. Unfortunately, for bigger instances, with more than 3000 nonzero entries, this approach is still impracticable.

6.2.4 Results for I_{CG}

Finally, we want to indicate the results for I_{CG} the column generation model introduced in Section 5.3. We are only interested whether this model yields a stronger LP-relaxation. Unfortunately, our approach could solve the LP in the root node only for 4 instance-load condition combinations within a timelimt of 1800 seconds for two blocks. The results are indicated in Table 6.17.

instance	lc	lb	ub	known optimum
stein9	LO	4.32	18	6
	ME	8.40	12	9
stein15	LO	5.84	47	9
stein27	LO	8.85	141	15

Table 6.17: Results for IP_{CG}

We observe that the bounds are much better than the trivial one obtained with the model IP_A. Unfortunately, the number of instances we could solve is not meaningful. It is necessary to speed up the pricing process, e.g. by solving it heuristically whenever possible and thus add more variables in every pricing round.

7 Final Remarks

In this thesis, we presented the theoretical background, complexity analysis, heuristic and exact approaches, and computational results concerning the problem of decomposing a matrix into arrowhead and bordered block diagonal form.

To be more precise, we motivated the problems MinAf and MinBf in Chapter 2 with the help of numerous applications. Furthermore, we discussed how the quality of decomposition can be measured.

In Chapter 3, we studied the complexity of these problems. We have found out that it is $\mathcal{NP}$-hard to obtain solutions with objective function value within an additive factor of the optimal objective value, even for only two blocks and matrices with at most three nonzero entries in every row and at most two nonzero entries in every column.

We have introduced two heuristic approaches for each problem and described them in detail in Chapter 4. We proved that the solutions found by these heuristics fulfill the block condition. Moreover, some relevant examples of failed runs were indicated.

In Chapter 5, we introduced two integer programs that solves MinAf and easily can be adapted to MinBf. We presented a column generation approach to solve the last one of the integer programs. In order to solve the pricing problem, we stated another integer program and we proved that a special case of the pricing problem can be solved by a polynomial time algorithm. One question still unanswered is whether this is true for the general pricing problem.

The algorithms introduced in Chapter 4 and Chapter 5 were implemented and tested in Chapter 6. For each problem, we found a good algorithm-parameter combination. We compared the quality of the found decompositions with the results of Ferris and Horn. Furthermore, we tested our algorithms on a huge instance set consisting of coefficient matrices of state of the art mixed integer programs and found decompositions on most of them. Moreover, we have visualized some of these coefficient matrices in original and decomposed state. Finally, we indicated our results for the exact algorithms. We add constraints to reduce the impact of symmetry and showed the positive influence of these constraints on the performance. Last but not least, we demonstrated that the column generation promises better LP bounds, although its applicability is very limited at the moment.

Outlook

Finally, we want to emphasize two interesting directions of research arising directly from this thesis:

On the one hand, it is still open how a good decomposition for the coefficient matrix of a mixed integer program should look like, to improve the solving process of the mixed integer program. This question is at present far from being solved, at least for the author.

On the other hand, it seems to be interesting how to implement the column generation approach in a more elaborate way. For instance, one could solve the pricing problem heuristically whenever possible and thus add more pricing variables per pricing round. Furthermore, one could combine the polynomial algorithm for the uncapacitated special case of the pricing problem with a local-search heuristic. Moreover, it would be interesting to study the complexity of the general pricing problem in detail.

8 Appendix

8.1 Zusammenfassung (German Summary)

In der mathematischen Optimierung sind Matrizen ein unverzichtbares Hilfsmittel zum Modellieren von Problemen. Ein besonders wichtiges Beispiel sind *gemischt ganzzahlige Programme*, deren Koeffizientenmatrix die Struktur des zu lösenden Problems verschlüsselt. Die Nichtnulleinträge einer Matrix können so angeordnet sein, dass ein Teil dieser Problemstruktur sichtbar ist. In dieser Diplomarbeit werden wir zwei solcher Anordnungen genauer betrachten: Die *k-arrowhead form(arh)* und die *bordered k-block diagonal form(bbd).* Grob gesagt, zeichnen sich beide dadurch aus, dass die Nichtnulleinträge entlang der Diagonalen, in k nichtleeren Teilmatrizen von A, k Blöcke bilden und alle anderen Einträge Null sind. Ausnahme sind bei der *bbd* nur die letzten Zeilen und bei der *arh* die letzten Zeilen und Spalten der Matrix. Dieser Ausnahmebereich der Matrix wird *Border* genannt.

Die vorliegende Arbeit beschäftigt sich mit dem Problem des Findens von Permutationen der Zeilen und Spalten einer Matrix A, die A für vorgegebenes k in *bbd* oder *arh* überführen, wobei die Border möglichst wenig Zeilen (*bbd*) bzw. Zeilen und Spalten (*arh*) enthält. Dabei werden sechs Ziele verfolgt:

- Vorstellung von Problemen, bei deren Lösungsvorgang ausgenutzt werden kann, dass die Koeffizientmatrix in *arh* oder *bbd* ist.
- Charakterisierung der Permutationen der Zeilen und Spalten einer Matrix A, die zu *arh* oder *bbd* führen.
- Komplexitätsanalyse der zugehörigen Optimierungsprobleme des Findens solcher Permutationen.
- Detailierte Beschreibung jeweils zweier heuristischer Verfahren für beide Optimierunsprobleme.
- Vorstellung zweier exakter Verfahren für *arh* mithilfe von ganzzahligen Programmen (die leicht auf *bbd* angepasst werden können).
- Analyse der Rechenergebnisse für die Implementation der vorgestellten Verfahren, die im Rahmen dieser Diplomarbeit entstanden ist.

Wir hoffen, dass der Leser von diesen Resultaten profitieren kann.

8.2 Background

This lemma states rigorously that given a k_1-decomposition $\mathcal{D}$ that fullfills the block condition, but not the load condition $(1, m, 1, n)$, it is possible to obtain a k_2-decomposition that fulfills the block condition and the load condition $(1, v, 1, w)$ with

$$k_2 := k_1 - |\{i \in [k_1] : |\mathcal{R}_i| = 0 \vee |\mathcal{C}_i| = 0\}|.$$

Lemma 8.2.1
Let $A \in \mathbb{R}^{m \times n}$ be a matrix. Furthermore, let $v \in [m]$, $w \in [n]$ and $k_1 \in \mathbb{N}$. Let $\mathcal{D} = ((\mathcal{R}_1, \ldots, \mathcal{R}_{k_1}, \mathcal{R}_B); (\mathcal{C}_1, \ldots, \mathcal{C}_{k_1}, \mathcal{C}_B))$ be a k_1-decomposition of A that fulfills the block condition and the load condition $(0, v, 0, w)$. Moreover, define $k_2 := k_1 - q$ with $q = |\{i \in [k_1] : |\mathcal{R}_i| = 0 \vee |\mathcal{C}_i| = 0\}|$. Then there is a k_2-decomposition that fulfills the block condition and the load condition $(1, v, 1, w)$.

Proof: Suppose $A, v, w, k_1, \mathcal{D}, q$ and k_2 are as defined above. Consider the set of blocks $\mathcal{B}^* := \{i \in [k_1] : |\mathcal{R}_i| = 0 \vee |\mathcal{C}_i| = 0\}$ and $\mathcal{B} := [k_1] \setminus \mathcal{B}^*$ which has k_2 elements. We denote by $\mathcal{R}^* := \bigcup_{\ell \in \mathcal{B}^*} \mathcal{R}_\ell$ and $\mathcal{C}^* := \bigcup_{\ell \in \mathcal{B}^*} \mathcal{C}_\ell$ the rows and the columns, respectively, that are in "half-empty" blocks. These rows and columns could be part of any $\mathcal{R}_\ell$ and $\mathcal{C}_\ell$, respectively, for $\ell \in \mathcal{B}$, without violating the block condition. In order to see that, consider a row $i \in \mathcal{R}_{\ell^*}$ with $\ell^* \in \mathcal{B}^*$ such that $\mathcal{R}_{\ell^*} \neq \emptyset$. Obviously, ℓ^* has no nonzero entry with any column $j \in \mathcal{C}_\ell$, for $\ell \in [k_1] \setminus \{\ell^*\}$ since the block condition is fulfilled. Analogously, every column $j \in \mathcal{C}_{\ell'}$ with $\ell' \in \mathcal{B}^*$ such that $\mathcal{C}_{\ell'} \neq \emptyset$ have no nonzero entry with any row $i \in \mathcal{R}_\ell$ for $\ell \in [k_1] \setminus \{\ell'\}$ since $\mathcal{D}$ fulfills the block condition.

Therefore, we can add each of the rows in $\mathcal{R}^*$ to one of the sets $\mathcal{R}_\ell$, for $\ell \in \mathcal{B}$ without violating the block condition. We add consecutively each row in $\mathcal{R}^*$ to one of the sets $\mathcal{R}_\ell$, for $\ell \in \mathcal{B}$ such that the upper row load condition is not violated. If the next addition of a row would violate this condition, we add the remaining rows to the set $\mathcal{R}_B$. Now we add consecutively each column in $\mathcal{C}^*$ to one of the sets $\mathcal{C}_\ell$, for $\ell \in \mathcal{B}$, until the upper column load condition would be violated. The remaining columns are added to the set $\mathcal{C}_B$.

Since the set $\mathcal{B}$ has k_2 elements, we can denote its elements by $\mathcal{B} = \{\ell_1, \ldots, \ell_{k_2}\}$ and define $\mathcal{R}' := (\mathcal{R}_{\ell_1}, \ldots, \mathcal{R}_{\ell_{k_2}}, \mathcal{R}_B)$ which has become a weak partition of the rows (by adding the rows of $\mathcal{R}^*$) and $\mathcal{C}' := (\mathcal{C}_{\ell_1}, \ldots, \mathcal{C}_{\ell_{k_2}}, \mathcal{C}_B)$ which has become a weak partition of the columns (by adding the columns of $\mathcal{C}^*$). Hence, $P' = (\mathcal{R}', \mathcal{C}')$ is a k_2-decomposition that fulfills the block condition, the old upper row load condition and the old upper column load condition. Moreover, all sets that form the row partition and the column partition are nonempty. Therefore, it also fulfills the load condition $(1, v, 1, w)$.

$\square$

8.3 Computational Tests

In this part of the appendix we present the results of our computational tests in detail.

In the following we show our results for all settings and all sizes of test instances to find the best decomposition to k-arrowhead form concerning the current measure. Afterwards we present the remaining results in detail. Explanations can be found in the main part of this thesis.

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
aflow30a	4	0.97	0.97	0.97	0.97	0.96	0.97	0.97	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.95	0.96	0.96	0.96	0.95	0.96	0.97	0.96	0.97	0.96	0.96	0.97	0.97	0.97	0.97	0.97
	6	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.97	0.96	0.98	0.98	0.98	0.98
aflow40b	4	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.98	0.98	0.97	0.98	0.98	0.99	0.98	0.99	0.98	0.99	0.98	0.98	0.99	0.99	0.98	0.99
	8	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
danoint	4	0.77	0.83	0.78	0.84	0.85	0.82	0.86	0.85	0.90	0.91	0.90	0.91	0.90	0.91	0.90	0.91
	16	0.64	0.80	0.66	0.80	0.77	0.80	0.79	0.80	0.85	0.85	0.85	0.85	0.86	0.85	0.86	0.85
	6	0.74	0.83	0.77	0.84	0.81	0.83	0.82	0.85	0.89	0.89	0.88	0.89	0.89	0.89	0.88	0.89
fiber	4	0.92	0.96	0.95	0.97	0.93	0.97	0.92	0.97	0.97	0.96	0.97	0.96	0.96	0.96	0.97	0.96
	16	0.88	0.95	0.89	0.95	0.89	0.96		0.96	0.96	0.91	0.96	0.91	0.96	0.90	0.96	0.90
	5	0.94	0.96	0.93	0.96	0.93	0.97	0.93	0.97	0.97	0.95	0.97	0.95	0.96	0.95	0.97	0.95
fixnet6	4	0.94	0.99	0.95	0.99	0.97	0.99	0.97	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.95	0.98	0.95	0.98	0.96	0.98			0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	6	0.94	0.98	0.94	0.99	0.96	0.99	0.97	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
gesa2	4	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.91	0.91	0.90	0.91	0.91	0.91	0.91	0.91	0.94	0.92	0.94	0.93	0.94	0.91	0.94	0.91
	8	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.95	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
gesa2-o	4	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.99	0.98	0.99	0.98	0.99	0.98
	16	0.93	0.95	0.93	0.95	0.92	0.95	0.93	0.95	0.97	0.94	0.97	0.94	0.97	0.93	0.97	0.95
	8	0.95	0.96	0.95	0.96	0.96	0.96	0.96	0.97	0.98	0.96	0.98	0.96	0.98	0.96	0.98	0.96
glass4	4	0.97	0.97	0.97	0.97	0.96	0.97	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.88	0.96	0.90	0.95	0.91	0.96		0.96	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97
	5	0.97	0.96	0.96	0.97	0.95	0.98	0.97	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
harp2	4		0.97		0.98				0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16									0.99	0.99	0.98	0.98	0.99	0.99	0.99	0.99
	4		0.97		0.98				0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
modglob	4	0.95	0.97	0.96	0.96	0.97	0.97	0.98	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.91	0.92	0.91	0.93	0.92	0.93	0.93	0.94	0.95	0.95	0.96	0.95	0.95	0.94	0.96	0.95
	5	0.95	0.96	0.95	0.96	0.95	0.95	0.95	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
noswot	4	0.91	0.90	0.93	0.93	0.93	0.91	0.93	0.93	0.94	0.93	0.94	0.93	0.94	0.93	0.94	0.94
	16		0.69			0.76	0.75	0.76		0.81	0.79	0.79	0.78	0.80	0.76	0.80	0.78
	4	0.91	0.90	0.93	0.93	0.93	0.91	0.93	0.93	0.94	0.93	0.94	0.93	0.94	0.93	0.94	0.94
opt1217	4						0.97		0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16											0.96	0.96	0.95	0.96	0.96	0.97
	3				0.94				0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
p2756	4	0.98	0.99	0.98	0.97	0.98	0.99	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.94	0.96	0.98	0.94	0.97	0.98	0.97	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	7	0.99	0.99	0.97	0.97	0.98	0.99	0.98	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
pk1	4							0.88									
	16																
	3	0.87	0.88			0.88	0.88	0.88	0.88								
pp08a	4	0.96	0.98	0.95	0.96	0.96	0.98	0.96	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.91	0.94	0.93	0.94	0.92	0.94	0.92	0.93	0.93	0.93	0.95	0.95	0.94	0.94	0.95	0.95
	4	0.96	0.96	0.94	0.97	0.97	0.96	0.96	0.96	0.98	0.98	0.98	0.97	0.98	0.97	0.98	0.98
pp08aCUTS	4	0.94	0.97	0.91	0.96	0.91	0.96	0.93	0.96	0.98	0.96	0.98	0.98	0.96	0.94	0.98	0.96
	16	0.90	0.93	0.88		0.90	0.94			0.94	0.94	0.96	0.96	0.96	0.94	0.96	0.94
	5	0.94	0.97	0.91	0.96	0.91	0.96	0.93	0.96	0.98	0.96	0.98	0.96	0.96	0.94	0.98	0.96
qiu	4	0.88	0.97	0.89	0.97	0.93	0.97	0.91	0.97	0.98	0.98	0.98	0.98	0.98	0.97	0.98	0.97
	16	0.83	0.95	0.84	0.95	0.85	0.97	0.84	0.96	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.96
	7	0.88	0.97	0.87	0.96	0.89	0.97	0.88	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
timtab1	4	0.95	0.95	0.95	0.95	0.95	0.96	0.95	0.95	0.97	0.96	0.97	0.96	0.97	0.96	0.97	0.96
	16	0.93	0.93	0.94	0.93	0.92	0.93		0.94	0.94	0.94	0.95	0.94	0.94	0.94	0.95	0.93
	4	0.95	0.95	0.95	0.95	0.95	0.96	0.95	0.95	0.97	0.97	0.97	0.96	0.97	0.96	0.97	0.96
timtab2	4	0.96	0.97	0.96	0.97	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.94	0.94	0.93	0.94	0.93	0.94	0.93	0.94	0.95	0.95	0.95	0.95	0.95	0.94	0.95	0.95
	5	0.96	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
tr12-30	4	0.98	0.97	0.98	0.97	0.98	0.97	0.98	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.89	0.95	0.89	0.94	0.89	0.96	0.90	0.95	0.97	0.92	0.95	0.92	0.97	0.91	0.98	0.93
	7	0.95	0.96	0.95	0.96	0.96	0.96	0.95	0.96	0.96	0.96	0.98	0.96	0.97	0.96	0.98	0.96
vpm2	4	0.95	0.94	0.94	0.95	0.95	0.95	0.96	0.95	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.97
	16	0.88	0.91	0.88	0.90	0.91	0.91	0.92	0.91	0.94	0.93	0.94	0.92	0.94	0.86	0.94	0.91
	5	0.94	0.93	0.94	0.94	0.95	0.93	0.95	0.94	0.96	0.96	0.96	0.96	0.96	0.95	0.96	0.95
arithm.mean		0.79	0.86	0.78	0.83	0.81	0.84	0.76	0.85	0.90	0.90	0.92	0.91	0.92	0.91	0.92	0.91
quadr.mean		0.86	0.90	0.85	0.89	0.87	0.89	0.84	0.90	0.93	0.93	0.94	0.94	0.94	0.93	0.94	0.94

Table 8.1: Results for small inst. to arrowhead conc.μ_{boN}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
10teams	4	0.90	0.94						0.93	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
	16																
	5	0.91								0.94	0.94	0.94	0.94	0.94	0.94	0.95	0.95
a1c1s1	4	0.96	0.95	0.96	0.95	0.97	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.88	0.93	0.89	0.93	0.89	0.93	0.89	0.92	0.96	0.92	0.96	0.92	0.96	0.89	0.95	0.90
	11	0.90	0.93	0.90	0.93	0.90	0.94	0.90	0.93	0.96	0.92	0.96	0.92	0.96	0.91	0.96	0.92
arki001	4	0.96	0.97	0.97	0.97	0.96	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.94	0.94	0.94	0.94	0.89	0.93	0.94	0.95	0.95	0.95	0.95	0.95	0.96	0.96	0.96	0.96
	8	0.93	0.94	0.93	0.93	0.92	0.94	0.98	0.98	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
liu	4	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	8	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
manna81	4	0.95	0.99	0.91	0.98	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.87	0.99	0.86	0.98	0.97	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	11	0.91	0.99	0.88	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
mkc	4	0.97	0.99	0.97	0.99	0.95	0.99	0.96	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.97				0.95	0.99	0.95	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
	11	0.96	0.99	0.97	0.99	0.96	0.99	0.96	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
mod011	4	0.99	0.99	0.98	0.99	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.96	0.97	0.95	0.97	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	12	0.97	0.98	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1.00	1.00	1.00	0.99
protfold	4	0.72	0.75	0.76	0.75	0.73	0.72	0.72	0.73	0.83	0.76	0.83	0.77	0.84	0.76	0.83	0.75
	16	0.59	0.66	0.61	0.65	0.58	0.62	0.57	0.63	0.74	0.68	0.74	0.73	0.75	0.58	0.76	0.59
	9	0.64	0.67	0.64	0.69	0.60	0.66	0.61	0.65	0.79	0.73	0.78	0.76	0.79	0.63	0.80	0.63
roll3000	4	0.93	0.94	0.93	0.94	0.92	0.93	0.91	0.94	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.88
	16	0.88	0.87	0.87	0.86	0.86	0.86	0.88	0.88	0.85	0.85	0.85	0.86	0.86	0.86	0.86	0.86
	8	0.93	0.93	0.91	0.93	0.91	0.92	0.90	0.93	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88
arithm.mean		0.87	0.82	0.77	0.79	0.81	0.82	0.81	0.86	0.91	0.90	0.91	0.90	0.91	0.89	0.91	0.89
quadr.mean		0.89	0.88	0.84	0.86	0.86	0.88	0.87	0.90	0.93	0.92	0.93	0.92	0.93	0.91	0.93	0.91

Table 8.2: Results for medium inst. to arrowhead conc. μ_{boN}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
air04	4	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
	16		0.92							0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
	7	0.92	0.92	0.92	0.92		0.92			0.93	0.93	0.93	0.93		0.93		0.93
air05	4	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
	16																
	6				0.95		0.95				0.95	0.95	0.95		0.95		
cap6000	4	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.96	1.00			1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	10	0.91	1.00		0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
dano3mip	4	0.94	0.90	0.93	0.90	0.94	0.90	0.93	0.90	0.94	0.95	0.95	0.95	0.95	0.95	0.95	0.95
	16	0.90	0.90	0.90	0.90	0.91	0.90	0.92	0.90	0.92	0.93	0.93	0.93	0.94	0.94	0.94	0.94
	11	0.92	0.90	0.93	0.90	0.91	0.90		0.90	0.93	0.94	0.94	0.94	0.95	0.95	0.94	0.95
disctom	4				0.97		0.98		0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16									0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	6				0.97		0.97			0.97	0.97	0.97	0.97	0.97	0.97		0.97
msc98-ip	4	0.98	0.99	0.98	0.99	0.98	1.00	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.95	0.97	0.94	0.97	0.95	0.97	0.95	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97
	18	0.93	0.96	0.94	0.95	0.94	0.96	0.95	0.96	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.97
net12	4	0.85	0.96	0.85	0.96	0.92	0.96	0.94	0.95	0.97	0.97	0.96	0.97	0.97	0.97	0.97	0.97
	16	0.78	0.95	0.79	0.94	0.90	0.95	0.93	0.94	0.95	0.96	0.95	0.95	0.95	0.95	0.95	0.96
	17	0.77	0.95	0.77	0.93	0.91	0.94	0.93	0.94	0.95	0.96	0.95	0.95	0.95	0.96	0.95	0.96
seymour	4	0.93	0.94	0.92	0.94	0.93	0.94	0.94	0.94	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
	16	0.88	0.90	0.88	0.90	0.88		0.88	0.90	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91
	8	0.90	0.91	0.89	0.91	0.90	0.91	0.90	0.91	0.93	0.93	0.92	0.93	0.93	0.93	0.93	0.92
swath	4	0.97	0.98	0.97	0.97	0.98	0.98	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.97	0.97		0.98	0.97	0.98	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	7	0.97	0.98	0.97	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
arithm.mean		0.71	0.74	0.61	0.81	0.70	0.81	0.67	0.74	0.89	0.93	0.92	0.92	0.86	0.93	0.82	0.89
quadr.mean		0.81	0.83	0.75	0.87	0.81	0.88	0.80	0.84	0.92	0.94	0.94	0.94	0.91	0.94	0.89	0.92

Table 8.3: Results for big inst. arrowhead conc. μ_{boN}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
aflow30a	4	0.90	0.92	0.91	0.92	0.90	0.92	0.91	0.91	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
	16	0.88	0.89	0.89	0.90	0.87	0.90	0.90	0.90	0.90	0.91	0.90	0.91	0.91	0.91	0.91	0.91
	6	0.89	0.91	0.91	0.90	0.91	0.92	0.91	0.91	0.94	0.93	0.90	0.90	0.94	0.94	0.94	0.94
aflow40b	4	0.96	0.97	0.97	0.97	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.96	0.95	0.92	0.95	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.96	0.96	0.95	0.96
	8	0.96	0.96	0.97	0.96	0.95	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
danoint	4	0.56	0.70	0.57	0.72	0.71	0.68	0.73	0.72	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
	16	0.31	0.65	0.36	0.65	0.58	0.65	0.61	0.65	0.73	0.73	0.73	0.73	0.74	0.73	0.75	0.73
	6	0.53	0.69	0.56	0.72	0.65	0.70	0.67	0.73	0.79	0.79	0.78	0.79	0.80	0.79	0.79	0.79
fiber	4	0.66	0.84	0.77	0.85	0.71	0.86	0.64	0.86	0.86	0.82	0.86	0.83	0.82	0.82	0.87	0.83
	16	0.53	0.78	0.56	0.79	0.59	0.81		0.82	0.81	0.64	0.81	0.64	0.80	0.59	0.81	0.56
	5	0.72	0.81	0.68	0.80	0.70	0.85	0.71	0.87	0.87	0.83	0.87	0.81	0.80	0.81	0.85	0.80
fixnet6	4	0.84	0.96	0.85	0.96	0.93	0.96	0.91	0.96	0.96	0.97	0.97	0.97	0.96	0.96	0.96	0.96
	16	0.86	0.94	0.85	0.95	0.88	0.94			0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
	6	0.84	0.95	0.84	0.96	0.89	0.96	0.91	0.96	0.96	0.97	0.96	0.96	0.96	0.96	0.96	0.97
gesa2	4	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
	16	0.83	0.83	0.82	0.83	0.83	0.83	0.83	0.84	0.88	0.85	0.88	0.87	0.87	0.83	0.87	0.83
	8	0.93	0.92	0.93	0.92	0.93	0.93	0.92	0.91	0.93	0.93	0.93	0.93	0.92	0.93	0.92	0.93
gesa2-o	4	0.96	0.96	0.95	0.96	0.96	0.95	0.95	0.95	0.97	0.96	0.97	0.96	0.97	0.96	0.97	0.96
	16	0.85	0.90	0.86	0.89	0.85	0.90	0.86	0.90	0.94	0.89	0.94	0.89	0.94	0.86	0.94	0.90
	8	0.90	0.92	0.91	0.93	0.92	0.92	0.91	0.93	0.96	0.92	0.96	0.92	0.96	0.92	0.96	0.92
glass4	4	0.93	0.94	0.93	0.94	0.93	0.94	0.93	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.94	0.95
	16	0.77	0.91	0.82	0.90	0.83	0.92		0.92	0.93	0.92	0.93	0.92	0.93	0.93	0.93	0.94
	5	0.93	0.93	0.93	0.93	0.91	0.95	0.94	0.94	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
harp2	4		0.28		0.49				0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
	16									0.57	0.57	0.55	0.54	0.57	0.57	0.57	0.57
	4		0.28		0.49				0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
modglob	4	0.88	0.92	0.89	0.89	0.92	0.92	0.94	0.92	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
	16	0.79	0.81	0.80	0.82	0.81	0.83	0.82	0.84	0.88	0.88	0.89	0.89	0.89	0.87	0.89	0.89
	5	0.87	0.90	0.88	0.89	0.88	0.88	0.88	0.91	0.93	0.94	0.93	0.94	0.94	0.94	0.94	0.94
noswot	4	0.83	0.82	0.87	0.87	0.87	0.84	0.87	0.87	0.87	0.88	0.86	0.87	0.87	0.88	0.86	0.88
	16		0.44			0.55	0.55	0.59		0.59	0.61	0.59	0.60	0.61	0.58	0.60	0.60
	4	0.83	0.82	0.87	0.87	0.87	0.84	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.88	0.86	0.87
opt1217	4						0.61		0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
	16											0.50	0.52	0.39	0.44	0.53	0.56
	3				0.16				0.67	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
p2756	4	0.93	0.97	0.92	0.88	0.94	0.97	0.93	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.84	0.87	0.94	0.80	0.91	0.93	0.91	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	7	0.96	0.97	0.88	0.90	0.93	0.96	0.92	0.97	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98
pk1	4							0.66									
	16																
	3	0.64	0.66			0.66	0.66	0.66	0.66								
pp08a	4	0.92	0.97	0.90	0.92	0.92	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.82	0.88	0.86	0.88	0.85	0.88	0.85	0.87	0.87	0.87	0.90	0.90	0.89	0.89	0.91	0.91
	4	0.93	0.93	0.88	0.94	0.94	0.92	0.91	0.92	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
pp08aCUTS	4	0.83	0.92	0.76	0.89	0.76	0.90	0.80	0.88	0.94	0.88	0.94	0.94	0.88	0.88	0.94	0.89
	16	0.74	0.79	0.68		0.71	0.85			0.85	0.85	0.88	0.88	0.88	0.88	0.88	0.85
	5	0.83	0.92	0.76	0.89	0.76	0.90	0.80	0.88	0.94	0.89	0.94	0.88	0.88	0.88	0.94	0.89
qiu	4	0.79	0.94	0.81	0.93	0.89	0.94	0.85	0.94	0.95	0.94	0.94	0.95	0.95	0.94	0.94	0.94
	16	0.68	0.90	0.70	0.90	0.72	0.92	0.72	0.92	0.94	0.93	0.94	0.93	0.94	0.92	0.94	0.92
	7	0.77	0.93	0.76	0.91	0.80	0.93	0.78	0.93	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
timtab1	4	0.89	0.90	0.91	0.90	0.91	0.92	0.89	0.90	0.95	0.94	0.96	0.94	0.95	0.94	0.94	0.91
	16	0.86	0.84	0.89	0.85	0.86	0.88		0.88	0.91	0.90	0.92	0.92	0.91	0.89	0.92	0.87
	4	0.89	0.90	0.91	0.90	0.91	0.92	0.89	0.90	0.96	0.95	0.96	0.94	0.96	0.93	0.96	0.91
timtab2	4	0.91	0.93	0.91	0.93	0.92	0.91	0.92	0.92	0.96	0.96	0.96	0.95	0.96	0.96	0.96	0.95
	16	0.89	0.89	0.87	0.87	0.87	0.89	0.87	0.89	0.92	0.92	0.92	0.92	0.93	0.90	0.92	0.91
	5	0.90	0.91	0.90	0.91	0.91	0.92	0.89	0.90	0.95	0.95	0.95	0.95	0.95	0.94	0.95	0.95
tr12-30	4	0.94	0.93	0.94	0.93	0.94	0.93	0.95	0.94	0.96	0.96	0.96	0.97	0.95	0.96	0.95	0.97
	16	0.72	0.88	0.72	0.86	0.73	0.89	0.75	0.88	0.92	0.84	0.89	0.85	0.94	0.84	0.95	0.86
	7	0.87	0.90	0.88	0.89	0.89	0.91	0.89	0.90	0.92	0.93	0.96	0.93	0.93	0.93	0.95	0.93
vpm2	4	0.86	0.86	0.84	0.87	0.87	0.88	0.89	0.88	0.92	0.92	0.92	0.92	0.91	0.92	0.91	0.92
	16	0.71	0.76	0.70	0.74	0.75	0.77	0.79	0.77	0.83	0.81	0.83	0.80	0.84	0.74	0.84	0.80
	5	0.85	0.83	0.86	0.84	0.86	0.83	0.87	0.83	0.90	0.92	0.90	0.90	0.90	0.91	0.89	0.91
arithm.mean		0.71	0.77	0.70	0.75	0.74	0.78	0.69	0.78	0.84	0.83	0.85	0.84	0.84	0.83	0.85	0.83
quadr.mean		0.77	0.82	0.77	0.81	0.79	0.83	0.77	0.83	0.87	0.86	0.87	0.86	0.87	0.86	0.87	0.86

Table 8.4: Results for small inst. to arrowhead conc. μ_{boA}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
10teams	4	0.11	0.45						0.30	0.53	0.53	0.53	0.53	0.54	0.53	0.53	0.53
	16																
	5	0.12								0.40	0.40	0.40	0.40	0.43	0.46	0.54	0.47
a1c1s1	4	0.91	0.91	0.92	0.90	0.93	0.92	0.92	0.92	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94
	16	0.76	0.85	0.77	0.85	0.76	0.85	0.76	0.85	0.92	0.84	0.92	0.85	0.91	0.80	0.91	0.82
	11	0.80	0.87	0.79	0.85	0.80	0.88	0.79	0.87	0.93	0.85	0.93	0.85	0.92	0.84	0.91	0.84
arki001	4	0.92	0.94	0.94	0.94	0.91	0.93	0.96	0.96	0.94	0.94	0.94	0.94	0.95	0.95	0.94	0.95
	16	0.87	0.87	0.86	0.87	0.78	0.85	0.87	0.88	0.89	0.89	0.90	0.88	0.90	0.90	0.90	0.90
	8	0.86	0.87	0.86	0.86	0.83	0.88	0.95	0.95	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90
liu	4	0.91	0.92	0.92	0.92	0.92	0.91	0.92	0.92	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91
	16	0.92	0.92	0.91	0.92	0.91	0.91	0.92	0.92	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91
	8	0.92	0.92	0.91	0.92	0.91	0.91	0.92	0.92	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91
manna81	4	0.92	0.97	0.85	0.96	0.96	0.98	0.96	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.79	0.98	0.78	0.96	0.95	0.97	0.95	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	11	0.85	0.98	0.81	0.98	0.95	0.98	0.95	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
mkc	4	0.93	0.98	0.92	0.98	0.90	0.99	0.91	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.91				0.89	0.98	0.89	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	11	0.91	0.98	0.92	0.98	0.92	0.98	0.90	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
mod011	4	0.95	0.97	0.93	0.97	0.97	0.98	0.97	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.87	0.92	0.83	0.89	0.92	0.96	0.94	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.97	0.98
	12	0.90	0.93	0.91	0.92	0.95	0.97	0.96	0.97	0.97	0.98	0.98	0.98	0.99	0.99	0.99	0.99
protfold	4	0.48	0.54	0.53	0.54	0.51	0.46	0.48	0.50	0.69	0.58	0.69	0.59	0.69	0.55	0.69	0.55
	16	0.27	0.36	0.29	0.35	0.25	0.31	0.24	0.33	0.52	0.46	0.52	0.48	0.54	0.25	0.54	0.27
	9	0.35	0.39	0.34	0.42	0.28	0.38	0.28	0.37	0.60	0.49	0.60	0.53	0.61	0.34	0.61	0.36
roll3000	4	0.88	0.89	0.89	0.90	0.86	0.89	0.83	0.90	0.73	0.73	0.74	0.73	0.73	0.73	0.73	0.73
	16	0.73	0.72	0.72	0.72	0.72	0.70	0.75	0.72	0.63	0.65	0.63	0.66	0.65	0.66	0.65	0.66
	8	0.86	0.87	0.84	0.86	0.84	0.84	0.80	0.88	0.71	0.71	0.71	0.71	0.71	0.72	0.71	0.71
arithm.mean		0.73	0.74	0.68	0.72	0.73	0.76	0.73	0.78	0.81	0.80	0.81	0.80	0.81	0.78	0.82	0.79
quadr.mean		0.78	0.80	0.76	0.80	0.79	0.82	0.80	0.83	0.84	0.83	0.84	0.83	0.85	0.83	0.85	0.83

Table 8.5: Results for medium inst. arrowhead conc. μ_{boA}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
air04	4	0.09	0.16	0.11	0.16	0.07	0.14	0.08	0.13	0.23	0.24	0.23	0.24	0.21	0.22	0.22	0.23
	16		0.09							0.16	0.16	0.16	0.16	0.17	0.17	0.16	0.19
	7	0.04	0.12	0.04	0.12		0.10			0.20	0.20	0.20	0.20		0.22		0.22
air05	4	0.06	0.14	0.07	0.15	0.06	0.16	0.07	0.15	0.18	0.19	0.17	0.19	0.18	0.18	0.17	0.19
	16																
	6				0.14		0.14				0.17	0.16	0.17		0.18		
cap6000	4	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.94	0.99			1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	10	0.86	1.00		0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
dano3mip	4	0.73	0.50	0.70	0.51	0.71	0.52	0.68	0.51	0.71	0.75	0.73	0.75	0.72	0.75	0.72	0.75
	16	0.60	0.55	0.59	0.51	0.60	0.50	0.63	0.50	0.60	0.65	0.61	0.65	0.68	0.68	0.68	0.68
	11	0.66	0.52	0.69	0.50	0.61	0.49		0.50	0.64	0.71	0.68	0.71	0.74	0.73	0.70	0.73
disctom	4				0.30		0.37		0.39	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	16									0.24	0.24	0.25	0.24	0.24	0.25	0.25	0.25
	6				0.25		0.26			0.25	0.25	0.25	0.25	0.25	0.25		0.25
msc98-ip	4	0.95	0.98	0.96	0.98	0.95	0.99	0.96	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.89	0.94	0.88	0.93	0.89	0.94	0.90	0.93	0.96	0.95	0.96	0.96	0.96	0.95	0.96	0.95
	18	0.85	0.91	0.88	0.91	0.87	0.92	0.89	0.92	0.96	0.95	0.96	0.96	0.96	0.95	0.95	0.95
net12	4	0.71	0.92	0.70	0.92	0.84	0.93	0.88	0.90	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
	16	0.56	0.89	0.59	0.89	0.80	0.90	0.86	0.88	0.90	0.92	0.90	0.90	0.90	0.92	0.90	0.92
	17	0.56	0.90	0.55	0.87	0.83	0.89	0.87	0.89	0.90	0.93	0.90	0.90	0.91	0.93	0.90	0.93
seymour	4	0.73	0.74	0.70	0.76	0.74	0.76	0.74	0.76	0.77	0.77	0.77	0.78	0.76	0.77	0.76	0.78
	16	0.54	0.58	0.54	0.59	0.53		0.54	0.59	0.60	0.62	0.60	0.62	0.61	0.62	0.61	0.62
	8	0.60	0.63	0.58	0.64	0.63	0.64	0.62	0.65	0.67	0.67	0.66	0.68	0.66	0.68	0.67	0.67
swath	4	0.80	0.87	0.85	0.84	0.86	0.87	0.86	0.87	0.87	0.88	0.88	0.87	0.87	0.87	0.87	0.87
	16	0.81	0.86		0.87	0.86	0.87	0.83	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
	7	0.82	0.87	0.83	0.87	0.86	0.87	0.86	0.87	0.87	0.87	0.87	0.87	0.88	0.87	0.87	0.87
arithm.mean		0.51	0.52	0.42	0.54	0.54	0.56	0.53	0.57	0.62	0.64	0.63	0.63	0.62	0.64	0.61	0.63
quadr.mean		0.63	0.65	0.56	0.65	0.67	0.68	0.67	0.68	0.71	0.72	0.71	0.72	0.72	0.72	0.71	0.72

Table 8.6: Results for big inst. to arrowhead conc. μ_{boA}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
	4	0.93	0.95	0.96	0.95	0.95	0.99	0.96	0.95	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.99
aflow30a	16	0.77	0.76	0.75	0.71	0.91	0.91	0.70	0.69	0.94	0.95	0.94	0.94	0.94	0.94	0.94	0.94
	6	0.95	0.96	0.93	0.88	0.95	0.97	0.94	0.96	0.97	0.94	0.95	0.95	0.90	0.95	0.89	0.97
	4	0.94	0.94	0.93	0.93	0.98	0.99	0.95	0.98	0.92	0.97	0.94	0.97	0.96	0.97	0.96	0.97
aflow40b	16	0.89	0.87	0.79	0.67	0.97	0.97	0.85	0.85	0.98	0.96	0.92	0.92	0.98	0.96	0.97	0.98
	8	0.94	0.92	0.92	0.90	0.98	0.99	0.92	0.96	0.95	0.97	0.96	0.95	0.96	0.96	0.97	0.96
	4	0.90	0.85	0.88	0.96	0.95	0.94	0.90	0.84	1.00	0.99	1.00	0.99	1.00	0.99	0.98	0.99
danoint	16	0.53	0.76	0.44	0.81	0.72	0.81	0.55	0.66	0.94	0.94	0.94	0.91	0.95	0.91	0.96	0.94
	6	0.84	0.79	0.88	0.81	0.82	0.90	0.83	0.81	0.95	0.95	0.96	0.89	0.97	0.94	0.81	0.90
	4	0.88	0.79	0.88	0.89	0.86	0.82	0.90	0.86	0.95	0.83	0.91	0.94	0.90	0.94	0.83	0.94
fiber	16	0.57	0.52	0.62	0.61	0.60	0.58		0.50	0.81	0.69	0.83	0.57	0.76	0.55	0.81	0.50
	5	0.82	0.90	0.76	0.88	0.86	0.80	0.76	0.66	0.93	0.84	0.77	0.85	0.87	0.90	0.78	0.91
	4	0.92	0.87	0.91	0.91	0.97	0.98	0.72	0.70	0.96	0.96	0.98	0.96	0.99	0.98	0.99	0.97
fixnet6	16	0.81	0.80	0.80	0.77	0.88	0.90			0.92	0.92	0.94	0.91	0.96	0.93	0.96	0.94
	6	0.89	0.89	0.90	0.80	0.90	0.95	0.66	0.74	0.95	0.94	0.93	0.93	0.96	0.93	0.91	0.92
	4	0.99	0.99	0.86	0.99	1.00	1.00	0.96	0.98	0.99	1.00	1.00	0.99	1.00	1.00	1.00	1.00
gesa2	16	0.89	0.80	0.72	0.82	0.85	0.87	0.84	0.89	0.89	0.89	0.93	0.89	0.92	0.86	0.91	0.86
	8	0.98	0.93	0.99	0.87	1.00	1.00	0.93	0.97	0.99	0.98	0.99	0.99	1.00	0.99	1.00	0.99
	4	0.99	0.90	0.93	0.96	1.00	0.99	0.98	0.94	0.98	1.00	0.96	0.99	0.94	1.00	0.94	1.00
gesa2-o	16	0.89	0.92	0.83	0.87	0.94	0.92	0.87	0.81	0.94	0.90	0.94	0.90	0.92	0.93	0.94	0.94
	8	0.92	0.93	0.93	0.89	1.00	0.94	0.96	0.85	0.96	0.95	0.96	0.98	0.94	0.99	0.95	1.00
	4	0.94	0.91	0.93	0.95	0.92	0.95	0.69	0.82	0.93	0.93	0.94	0.92	0.93	0.92	0.94	0.94
glass4	16	0.49	0.59	0.54	0.69	0.66	0.80		0.64	0.80	0.78	0.82	0.81	0.85	0.79	0.80	0.74
	5	0.92	0.91	0.87	0.88	0.86	0.93	0.77	0.85	0.93	0.92	0.88	0.93	0.91	0.93	0.77	0.92
	4		0.47		0.55				0.69	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
harp2	16									0.95	0.95	0.93	0.89	0.96	0.96	0.96	0.96
	4		0.47		0.55				0.69	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
	4	0.96	0.92	0.90	0.89	0.94	0.96	0.92	0.87	0.98	0.97	0.96	0.95	0.98	0.98	0.96	0.97
modglob	16	0.84	0.65	0.69	0.71	0.81	0.80	0.70	0.72	0.81	0.88	0.79	0.75	0.86	0.81	0.75	0.70
	5	0.88	0.94	0.93	0.91	0.94	0.93	0.85	0.93	0.91	0.93	0.93	0.95	0.92	0.94	0.90	0.92
	4	0.90	0.89	0.88	0.92	0.87	0.88	0.93	0.92	0.93	0.95	0.93	0.92	0.93	0.93	0.92	0.90
noswot	16		0.41			0.65	0.65	0.61		0.59	0.76	0.58	0.65	0.65	0.71	0.65	0.57
	4	0.90	0.89	0.88	0.92	0.87	0.88	0.93	0.92	0.92	0.92	0.89	0.95	0.93	0.90	0.92	0.87
	4						0.80		0.73	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
opt1217	16											0.94	0.90	0.72	0.74	0.92	0.94
	3				0.38				0.82	1.00	1.00	1.00	1.00	0.89	1.00	0.79	0.89
	4	0.84	0.89	0.82	0.82	0.91	0.94	0.78	0.92	0.93	0.96	0.93	0.95	0.95	0.97	0.93	0.96
p2756	16	0.73	0.75	0.75	0.71	0.84	0.89	0.72	0.81	0.92	0.92	0.94	0.92	0.91	0.92	0.91	0.91
	7	0.90	0.88	0.73	0.82	0.86	0.92	0.81	0.91	0.84	0.95	0.90	0.93	0.92	0.88	0.82	0.92
	4							0.27									
pk1	16																
	3	0.40	0.41			0.41	0.40	0.43	0.30								
	4	0.95	0.96	0.93	0.89	0.92	0.96	0.85	0.96	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
pp08a	16	0.74	0.79	0.65	0.79	0.77	0.85	0.77	0.79	0.89	0.83	0.81	0.80	0.85	0.91	0.68	0.82
	4	0.95	0.90	0.91	0.94	0.96	0.95	0.88	0.75	0.94	0.94	0.94	0.93	0.88	0.90	0.88	0.88
	4	0.90	0.98	0.86	0.93	0.94	0.95	0.76	0.69	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94
pp08aCUTS	16	0.65	0.51	0.51		0.71	0.84			1.00	0.90	0.88	0.69	0.88	1.00	0.88	0.60
	5	0.90	0.98	0.86	0.93	0.94	0.95	0.76	0.69	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94
	4	0.88	0.94	0.89	0.92	0.98	0.98	0.88	0.94	0.99	0.95	0.99	0.98	0.94	0.97	0.93	0.96
qiu	16	0.72	0.84	0.74	0.83	0.84	0.93	0.58	0.81	0.94	0.98	0.95	0.98	0.95	0.90	0.95	0.87
	7	0.85	0.92	0.81	0.90	0.85	0.95	0.70	0.93	0.99	0.94	0.99	0.93	0.99	0.89	0.89	0.89
	4	0.90	0.91	0.87	0.87	0.94	0.93	0.90	0.88	0.87	0.90	0.92	0.89	0.84	0.88	0.89	0.87
timtab1	16	0.36	0.39	0.51	0.26	0.87	0.85		0.60	0.77	0.79	0.62	0.68	0.67	0.72	0.62	0.64
	4	0.90	0.91	0.87	0.87	0.94	0.93	0.90	0.88	0.88	0.89	0.87	0.88	0.87	0.87	0.88	0.90
	4	0.94	0.93	0.92	0.92	0.96	0.92	0.96	0.94	0.81	0.93	0.84	0.87	0.87	0.90	0.86	0.86
timtab2	16	0.60	0.75	0.54	0.62	0.80	0.84	0.63	0.69	0.77	0.76	0.76	0.78	0.74	0.78	0.72	0.73
	5	0.91	0.92	0.90	0.88	0.92	0.93	0.94	0.92	0.80	0.82	0.78	0.84	0.74	0.87	0.81	0.85
	4	0.97	0.93	0.94	0.95	0.93	0.93	0.95	0.96	0.96	0.96	1.00	0.95	0.98	0.97	0.99	0.97
tr12-30	16	0.72	0.90	0.78	0.87	0.79	0.92	0.63	0.66	0.95	0.88	0.90	0.81	0.97	0.77	0.97	0.78
	7	0.91	0.91	0.90	0.92	0.88	0.96	0.94	0.93	0.92	0.95	0.96	0.93	0.93	0.88	0.79	0.88
	4	0.90	0.88	0.93	0.97	0.93	0.88	0.96	0.92	0.98	0.95	0.95	0.95	0.94	0.95	0.95	0.95
vpm2	16	0.72	0.81	0.66	0.70	0.71	0.82	0.57	0.46	0.92	0.91	0.78	0.62	0.90	0.79	0.72	0.58
	5	0.94	0.82	0.92	0.86	0.87	0.84	0.78	0.79	0.93	0.88	0.93	0.94	0.85	0.89	0.77	0.90
arith.mean		0.72	0.74	0.69	0.72	0.76	0.79	0.65	0.72	0.86	0.86	0.87	0.86	0.87	0.87	0.85	0.85
quadr.mean		0.79	0.80	0.76	0.78	0.82	0.85	0.73	0.77	0.90	0.89	0.89	0.89	0.89	0.89	0.87	0.87

Table 8.7: Results for small inst. to arrowhead conc. μ_{blB}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
10teams	4	0.53	0.81			0.53	0.81			0.93	0.95	0.93	0.94	1.00	0.97	0.97	0.97
	16																
	5	0.32								0.84	0.84	0.83	0.84	0.85	0.86	0.74	0.85
a1c1s1	4	0.92	0.93	0.91	0.94	0.91	0.94	0.91	0.96	0.96	0.94	0.95	0.94	0.96	0.94	0.96	0.94
	16	0.77	0.84	0.74	0.82	0.74	0.86	0.72	0.70	0.89	0.86	0.91	0.86	0.88	0.85	0.89	0.84
	11	0.84	0.86	0.78	0.82	0.80	0.89	0.85	0.88	0.88	0.88	0.88	0.86	0.88	0.86	0.73	0.84
arki001	4	0.92	0.90	0.93	0.93	0.92	0.95	0.83	0.83	0.91	0.93	0.92	0.95	0.93	0.90	0.94	0.93
	16	0.68	0.69	0.65	0.73	0.70	0.75	0.59	0.66	0.71	0.70	0.72	0.67	0.71	0.76	0.86	0.86
	8	0.77	0.87	0.73	0.77	0.83	0.83	0.65	0.68	0.84	0.84	0.85	0.85	0.91	0.86	0.94	0.93
liu	4	0.95	0.97	0.94	0.94	0.99	0.98	0.96	0.97	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.93	0.95	0.88	0.86	0.96	0.93	0.84	0.89	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	8	0.95	0.95	0.92	0.91	0.98	0.97	0.93	0.92	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.99
manna81	4	0.98	0.95	0.96	0.91	0.99	0.99	0.99	0.70	0.96	0.99	0.96	0.96	1.00	1.00	1.00	1.00
	16	0.90	0.96	0.86	0.90	0.97	0.99	0.68	0.68	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	11	0.95	0.96	0.91	0.93	0.97	0.99	0.70	0.69	0.93	0.93	0.96	0.95	0.86	0.91	0.89	0.93
mkc	4	0.79	0.89	0.84	0.84	0.91	0.90	0.85	0.88	0.97	0.97	0.97	0.97	0.97	0.98	0.98	0.97
	16	0.77				0.75	0.75	0.53	0.61	0.91	0.91	0.91	0.89	0.92	0.91	0.91	0.91
	11	0.78	0.83	0.77	0.76	0.79	0.84	0.67	0.71	0.88	0.92	0.92	0.91	0.76	0.77	0.80	0.77
mod011	4	0.61	0.61	0.59	0.57	0.53	0.58	0.52	0.62	0.90	0.90	0.90	0.93	0.95	0.95	0.95	0.95
	16	0.47	0.45	0.42	0.39	0.45	0.45	0.44	0.34	0.70	0.67	0.67	0.60	0.61	0.61	0.62	0.62
	12	0.43	0.45	0.44	0.46	0.48	0.53	0.47	0.39	0.80	0.79	0.72	0.70	0.52	0.68	0.52	0.62
protfold	4	0.69	0.89	0.81	0.93	0.81	0.72	0.69	0.75	0.95	0.85	0.93	0.85	0.91	0.87	0.93	0.85
	16	0.36	0.51	0.43	0.47	0.34	0.52	0.36	0.44	0.50	0.61	0.51	0.42	0.51	0.48	0.52	0.47
	9	0.67	0.67	0.51	0.67	0.53	0.80	0.56	0.57	0.61	0.70	0.66	0.77	0.60	0.56	0.59	0.55
roll3000	4	0.90	0.89	0.89	0.76	0.88	0.81	0.78	0.81	0.87	0.87	0.89	0.86	0.82	0.83	0.85	0.80
	16	0.64	0.69	0.60	0.48	0.66	0.70	0.46	0.44	0.60	0.51	0.64	0.56	0.69	0.63	0.67	0.64
	8	0.78	0.80	0.74	0.72	0.76	0.71	0.62	0.69	0.70	0.75	0.69	0.70	0.71	0.71	0.75	0.68
arithm.mean		0.71	0.71	0.64	0.65	0.71	0.75	0.61	0.62	0.82	0.82	0.83	0.81	0.81	0.81	0.81	0.81
quadr.mean		0.75	0.77	0.71	0.72	0.76	0.79	0.67	0.68	0.85	0.85	0.85	0.84	0.84	0.84	0.84	0.84

Table 8.8: Results for medium inst. to arrowhead conc.μ_{blB}

Algorithm		BIPARTITE DECOMPOSING								HYPERCOLROW DECOMPOSING							
Metis method		PKW				RECURSIVE				PKW				RECURSIVE			
Dummy ratio		0%		20%		0%		20%		0%		20%		0%		20%	
Weighting scheme		un.	ad.	un.	ad.	un.	ad.	un.	ad.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl																
air04	4	0.60	0.68	0.54	0.51	0.58	0.86	0.41	0.44	0.93	0.85	0.92	0.91	0.87	0.98	0.90	0.95
	16		0.38							0.58	0.60	0.57	0.57	0.62	0.65	0.55	0.59
	7	0.46	0.50	0.35	0.58		0.51			0.77	0.83	0.81	0.83		0.73		0.80
air05	4	0.56	0.75	0.47	0.61	0.61	0.66	0.54	0.64	0.78	0.86	0.89	0.88	0.78	0.91	0.92	0.83
	16																
	6				0.40		0.52				0.79	0.73	0.72		0.74		
cap6000	4	0.08		0.07	0.12	0.93	0.92	0.66	0.83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.01	0.01			0.83	0.84	0.60	0.64	0.96	0.97	0.97	0.98	0.97	0.97	0.97	0.98
	10	0.02	0.02		0.03	0.88	0.83	0.62	0.74	0.97	0.98	0.98	0.99	0.87	0.92	0.86	0.93
dano3mip	4	0.87	0.87	0.75	0.91	0.72	0.93	0.65	0.85	0.89	0.92	0.93	0.92	0.97	0.99	0.96	0.95
	16	0.41	0.63	0.39	0.66	0.38	0.65	0.30	0.73	0.61	0.57	0.63	0.66	0.87	0.74	0.88	0.72
	11	0.36	0.72	0.65	0.62	0.31	0.77		0.54	0.72	0.63	0.74	0.62	0.85	0.73	0.78	0.69
disctom	4				0.65		0.73		0.48	0.96	0.96	1.00	0.96	0.96	0.96	1.00	0.96
	16									0.79	0.70	0.73	0.72	0.86	0.86	0.83	0.76
	6				0.41		0.51			0.93	0.98	0.83	0.93	0.63	0.93		0.90
msc98-ip	4	0.43	0.44	0.44	0.44	0.44	0.49	0.43	0.47	0.49	0.52	0.49	0.59	0.50	0.58	0.48	0.50
	16	0.39	0.41	0.36	0.41	0.38	0.42	0.38	0.39	0.41	0.42	0.39	0.41	0.39	0.39	0.40	0.39
	18	0.34	0.39	0.36	0.41	0.36	0.43	0.37	0.37	0.42	0.42	0.43	0.44	0.39	0.38	0.35	0.40
net12	4	0.90	0.87	0.88	0.93	0.92	0.95	0.95	0.93	0.94	0.91	0.96	0.97	1.00	0.92	0.99	1.00
	16	0.64	0.77	0.66	0.78	0.77	0.93	0.79	0.82	0.83	0.89	0.84	0.96	0.96	0.91	0.96	0.98
	17	0.66	0.79	0.65	0.79	0.73	0.92	0.74	0.80	0.89	0.99	0.85	0.93	0.93	0.98	0.88	0.99
seymour	4	0.67	0.67	0.65	0.66	0.50	0.53	0.74	0.59	0.65	0.61	0.64	0.63	0.71	0.64	0.70	0.64
	16	0.23	0.30	0.27	0.40	0.27		0.22	0.26	0.23	0.18	0.22	0.23	0.18	0.17	0.18	0.25
	8	0.44	0.35	0.39	0.35	0.40	0.39	0.33	0.29	0.29	0.30	0.29	0.30	0.28	0.30	0.28	0.27
swath	4	0.31	0.34	0.38	0.38	0.68	0.81	0.53	0.61	1.00	1.00	0.99	0.99	0.97	0.97	0.99	0.99
	16	0.07	0.07		0.06	0.55	0.57	0.34	0.48	0.95	0.94	0.96	0.97	0.89	0.87	0.92	0.94
	7	0.18	0.19	0.15	0.18	0.64	0.73	0.41	0.62	0.89	0.93	0.97	0.96	0.39	0.48	0.39	0.28
arithm.mean		0.32	0.38	0.31	0.42	0.44	0.59	0.37	0.46	0.70	0.73	0.73	0.74	0.66	0.73	0.64	0.69
quadr.mean		0.43	0.49	0.42	0.50	0.54	0.66	0.47	0.55	0.76	0.78	0.78	0.79	0.74	0.78	0.73	0.76

Table 8.9: Results for big inst. to arrowhead conc. μ_{blB}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		REC.		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
aflow30a	4	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97
	16	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.98	0.96
	6	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.97	0.98	0.98	0.98	0.98
aflow40b	4	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.99	0.99	0.98	0.99	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.98
	8	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
danoint	4	0.86	0.86	0.86	0.86	0.82	0.77	0.83	0.81	0.73	0.71	0.72	0.74
	16	0.80	0.81	0.82	0.83			0.70	0.72	0.66			
	6	0.85	0.84	0.85	0.85	0.78	0.74	0.81	0.79	0.77	0.74		
fiber	4	0.97	0.97	0.97	0.97	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.97
	16	0.97	0.96	0.96	0.96		0.92			0.92			
	5	0.97	0.97	0.96	0.97	0.95	0.95	0.95	0.95	0.95	0.95	0.96	0.95
fixnet6	4	0.99	0.99	0.99	0.99	0.96	0.96	0.96	0.97	0.95	0.96	0.95	0.96
	16	0.99	0.99	0.99	0.99	0.94	0.94	0.94	0.94	0.93	0.93	0.93	0.93
	6	0.99	0.99	0.99	0.99	0.96	0.96	0.95	0.96	0.95	0.95	0.94	0.95
gesa2	4	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.92	0.92	0.91	0.92	0.88	0.87	0.87	0.87	0.87	0.86	0.87	0.88
	8	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
gesa2-o	4	0.99	0.98	0.99	0.99	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.97	0.97	0.97	0.97	0.90	0.92	0.90	0.92	0.90	0.92	0.90	0.91
	8	0.98	0.98	0.98	0.98	0.96	0.95	0.96	0.94	0.96	0.94	0.96	0.94
glass4	4	0.69	0.69	0.66	0.69	0.64	0.64	0.64	0.64	0.63	0.65	0.65	0.65
	16	0.57	0.57			0.57	0.58	0.58	0.58	0.57	0.58	0.57	0.58
	5	0.64	0.66	0.64	0.66	0.62	0.63	0.63	0.63	0.62	0.62	0.62	0.62
harp2	4	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	16	0.98	0.99	0.99	0.99	0.99	0.99			0.99	0.99		
	4	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
modglob	4	0.98	0.98	0.98	0.98	0.96	0.95	0.97	0.95	0.96	0.95	0.97	0.96
	16	0.95	0.95	0.95	0.95	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
	5	0.97	0.97	0.97	0.97	0.97	0.95	0.97	0.95	0.97	0.95	0.96	0.96
noswot	4	0.92	0.94	0.92	0.94	0.92	0.92	0.93	0.94	0.92	0.91	0.95	0.95
	16									0.67			
	4	0.92	0.94	0.92	0.94	0.92	0.92	0.93	0.94	0.91	0.91	0.95	0.95
opt1217	4	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.96	0.97		0.98	0.98	0.98			0.98	0.98		
	3	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
p2756	4	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.99
	16	0.99	0.99	0.99	0.99	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.97
	7	1.00	1.00	1.00	1.00	0.99	0.99	0.98	0.98	0.99	0.99	0.98	0.97
pk1	4												
	16												
	3												
pp08a	4	0.98	0.98	0.98	0.98	0.90	0.94	0.92	0.94	0.88	0.87	0.92	0.93
	16	0.94	0.94	0.94	0.95	0.92	0.92			0.91	0.91		
	4	0.98	0.98	0.97	0.98	0.93	0.93	0.93	0.94	0.89	0.90	0.90	0.90
pp08aCUTS	4	0.98	0.98	0.98	0.98	0.96	0.94	0.94	0.94	0.92	0.94	0.94	0.93
	16	0.96	0.96	0.96	0.96	0.86	0.88	0.88	0.89	0.88	0.90	0.87	0.89
	5	0.98	0.98	0.98	0.98	0.93	0.94	0.96	0.94	0.94	0.94	0.93	0.93
qiu	4	0.91	0.90	0.94	0.94	0.92	0.88	0.93	0.92	0.92	0.87	0.91	0.84
	16	0.83	0.84	0.84	0.84								0.79
	7	0.89	0.88	0.88	0.89	0.75	0.79	0.77	0.80	0.82	0.80		0.80
timtab1	4	0.89	0.89	0.89	0.89	0.87	0.86	0.87	0.88	0.83	0.85	0.87	0.87
	16												
	4	0.89	0.89	0.89	0.89	0.87	0.87	0.87	0.88	0.83	0.85	0.87	0.88
timtab2	4	0.91	0.91	0.91	0.91	0.90	0.90	0.89	0.90	0.88	0.88	0.88	0.88
	16												
	5	0.90	0.90	0.90	0.91	0.87	0.89	0.87	0.87	0.88	0.89	0.88	0.88
tr12-30	4	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	16	0.98	0.98	0.97	0.98	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.92
	7	0.98	0.98	0.98	0.98	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
vpm2	4	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.94	0.94	0.94	0.93	0.91	0.90	0.91	0.91	0.91	0.90	0.92	0.92
	5	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
arithm.mean		0.85	0.85	0.82	0.84	0.79	0.80	0.76	0.76	0.82	0.78	0.72	0.74
quadr. mean		0.89	0.90	0.88	0.89	0.85	0.86	0.84	0.84	0.87	0.85	0.82	0.83

Table 8.10: Results for small inst. to bbd. conc. μ_{boN}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
10teams	4	0.95	0.95	0.95	0.95								
	16												
	5	0.94	0.95	0.94	0.95								
a1c1s1	4	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	0.91	0.91	0.91	0.91	0.88	0.90	0.88	0.90	0.86	0.90	0.86	0.90
	11	0.93	0.92	0.93	0.93	0.90	0.91	0.90	0.91	0.90	0.91	0.90	0.91
arki001	4	0.97	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.93	0.92	0.93	0.93
	16	0.95	0.96	0.96	0.96					0.84		0.86	0.85
	8	0.96	0.96	0.96	0.96	0.95	0.95	0.95	0.94	0.89	0.90	0.89	0.89
liu	4	0.54		0.56									
	16												
	8												
manna81	4	0.86	0.86	0.86	0.86	0.84	0.85	0.85	0.85	0.79	0.79	0.79	0.79
	16	0.80	0.80	0.80	0.80	0.72	0.72	0.72	0.72	0.75	0.75	0.72	0.72
	11	0.81	0.81	0.81	0.82	0.74	0.74	0.75	0.75	0.77	0.77	0.77	0.77
mkc	4	1.00	1.00	1.00	1.00	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	16	1.00	1.00	1.00	1.00	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
	11	1.00	1.00	1.00	1.00	0.97	0.96	0.96	0.96	0.96	0.96	0.97	
mod011	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	12	0.99	0.99	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
protfold	4	0.85	0.86	0.86	0.86	0.74	0.74	0.77	0.75		0.75	0.75	0.73
	16	0.71	0.71	0.72	0.72								
	9	0.80	0.80	0.79	0.80		0.70		0.67	0.72	0.67		0.67
roll3000	4	0.96	0.96	0.96	0.96	0.81	0.84	0.80	0.87	0.81	0.80	0.84	0.82
	16	0.88	0.88	0.88	0.89	0.83	0.82	0.85	0.82	0.81	0.78	0.80	0.82
	8	0.94	0.94	0.94	0.94	0.85	0.83	0.86	0.78	0.78	0.82	0.83	0.80
arithm.mean		0.80	0.78	0.80	0.79	0.59	0.62	0.60	0.62	0.62	0.61	0.62	0.61
quadr. mean		0.86	0.85	0.86	0.85	0.73	0.75	0.74	0.75	0.74	0.74	0.74	0.73

Table 8.11: Results for medium inst. to bbd. conc. μ_{boN}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
air04	4	0.93	0.93	0.93	0.93	0.94			0.94	0.94	0.94	0.94	
	16	0.93	0.93	0.93	0.93								
	7	0.93	0.93	0.93			0.93				0.93		
air05	4	0.95	0.95	0.95	0.95	0.96							
	16												
	6	0.95	0.95	0.95									
cap6000	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
dano3mip	4	0.92	0.93	0.93	0.93	0.90	0.91	0.90	0.91	0.90	0.89	0.89	0.89
	16	0.90	0.90	0.91	0.91	0.89	0.89	0.89	0.89	0.88	0.88	0.88	0.88
	11	0.91	0.91	0.92	0.92	0.88	0.89	0.89	0.89	0.88	0.89	0.89	0.89
disctom	4	0.97	0.97	0.97	0.97					0.98	0.98	0.98	0.98
	16	0.97	0.97	0.97	0.97			0.98	0.98				
	6	0.97	0.97	0.97	0.97								
msc98-ip	4			0.97	0.97	0.88	0.88	0.88	0.88	0.88	0.88	0.89	0.88
	16	0.91	0.91	0.92	0.92	0.80	0.80	0.80	0.80	0.80	0.80	0.81	0.80
	18	0.91	0.91	0.92	0.92	0.79	0.79	0.80	0.80		0.80		0.80
net12	4	0.97	0.98	0.98	0.98	0.63	0.63	0.63		0.62	0.63	0.62	0.62
	16	0.90	0.95	0.92	0.96								
	17	0.90	0.94	0.92	0.95	0.53	0.53	0.53	0.53				
seymour	4	0.78	0.78	0.79	0.81	0.69	0.71	0.70	0.75	0.67	0.68	0.67	0.73
	16	0.52	0.55	0.54	0.57			0.48		0.46		0.46	0.48
	8	0.62	0.67	0.66	0.72	0.57	0.57	0.58	0.59	0.52	0.57	0.55	0.55
swath	4	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.97
	16												
	7	0.95	0.95		0.95			0.95	0.95	0.96	0.96	0.96	0.96
arithm.mean		0.81	0.81	0.81	0.78	0.50	0.46	0.52	0.51	0.50	0.55	0.50	0.50
quadr. mean		0.86	0.87	0.87	0.85	0.66	0.63	0.67	0.68	0.66	0.70	0.66	0.66

Table 8.12: Results for big inst. to bbd. conc. μ_{boN}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
	4	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.93
aflow30a	16	0.90	0.89	0.92	0.91	0.92	0.91	0.93	0.91	0.92	0.91	0.93	0.90
	6	0.94	0.93	0.94	0.94	0.93	0.92	0.92	0.92	0.94	0.94	0.94	0.94
	4	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.96
aflow40b	16	0.96	0.96	0.96	0.96	0.96	0.95	0.96	0.95	0.96	0.96	0.96	0.95
	8	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.96	0.97	0.96	0.97	0.96
	4	0.75	0.75	0.75	0.75	0.68	0.59	0.69	0.67	0.51	0.49	0.51	0.53
danoint	16	0.65	0.66	0.68	0.69			0.46	0.50	0.39			
	6	0.73	0.71	0.74	0.73	0.61	0.54	0.66	0.62	0.59	0.54		
	4	0.87	0.87	0.87	0.87	0.76	0.76	0.78	0.77	0.79	0.78	0.77	0.84
fiber	16	0.85	0.82	0.80	0.82		0.61			0.63			
	5	0.87	0.87	0.84	0.85	0.77	0.77	0.77	0.77	0.77	0.77	0.80	0.79
	4	0.96	0.96	0.96	0.97	0.88	0.89	0.90	0.90	0.87	0.87	0.87	0.88
fixnet6	16	0.96	0.96	0.96	0.96	0.82	0.82	0.82	0.83	0.80	0.81	0.81	0.81
	6	0.96	0.96	0.96	0.96	0.88	0.88	0.87	0.88	0.85	0.85	0.83	0.85
	4	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
gesa2	16	0.84	0.84	0.84	0.84	0.77	0.75	0.76	0.76	0.76	0.74	0.76	0.77
	8	0.93	0.93	0.93	0.93	0.92	0.92	0.92	0.92	0.93	0.93	0.93	0.93
	4	0.97	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
gesa2-o	16	0.94	0.94	0.94	0.94	0.80	0.83	0.81	0.84	0.81	0.83	0.81	0.82
	8	0.96	0.96	0.96	0.96	0.91	0.90	0.92	0.89	0.92	0.89	0.92	0.89
	4	0.44	0.43	0.39	0.44	0.35	0.34	0.34	0.34	0.34	0.37	0.36	0.36
glass4	16	0.21	0.22			0.23	0.24	0.24	0.25	0.21	0.24	0.22	0.24
	5	0.35	0.38	0.34	0.38	0.31	0.33	0.33	0.33	0.31	0.31	0.31	0.31
	4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
harp2	16	0.55	0.57	0.57	0.57	0.65	0.65			0.65	0.65		
	4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
	4	0.95	0.95	0.95	0.95	0.90	0.88	0.93	0.88	0.90	0.88	0.92	0.90
modglob	16	0.89	0.89	0.89	0.88	0.82	0.83	0.83	0.83	0.83	0.82	0.82	0.82
	5	0.93	0.93	0.94	0.94	0.92	0.88	0.91	0.88	0.91	0.88	0.91	0.89
	4	0.86	0.89	0.86	0.89	0.86	0.86	0.89	0.90	0.86	0.85	0.91	0.91
noswot	16									0.43			
	4	0.86	0.89	0.87	0.89	0.86	0.86	0.89	0.90	0.85	0.84	0.91	0.91
	4	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
opt1217	16	0.45	0.64		0.75	0.75	0.75			0.75	0.75		
	3	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
	4	0.98	0.98	0.98	0.98	0.95	0.94	0.95	0.94	0.96	0.91	0.97	0.94
p2756	16	0.97	0.97	0.97	0.97	0.82	0.82	0.81	0.82	0.88	0.84	0.88	0.86
	7	0.98	0.98	0.98	0.98	0.94	0.93	0.91	0.91	0.97	0.96	0.90	0.87
	4												
pk1	16												
	3												
	4	0.97	0.97	0.97	0.97	0.80	0.88	0.85	0.87	0.76	0.75	0.85	0.85
pp08a	16	0.88	0.88	0.88	0.89	0.84	0.84			0.81	0.82		
	4	0.95	0.96	0.95	0.96	0.87	0.85	0.87	0.87	0.78	0.79	0.81	0.81
	4	0.94	0.94	0.94	0.94	0.89	0.82	0.85	0.82	0.78	0.82	0.82	0.82
pp08aCUTS	16	0.88	0.88	0.88	0.88	0.60	0.68	0.65	0.70	0.68	0.73	0.63	0.71
	5	0.94	0.94	0.94	0.94	0.80	0.82	0.88	0.82	0.82	0.82	0.80	0.80
	4	0.84	0.83	0.89	0.89	0.87	0.80	0.89	0.87	0.87	0.79	0.85	0.73
qiu	16	0.72	0.73	0.73	0.73								0.64
	7	0.81	0.80	0.80	0.81	0.57	0.64	0.61	0.65	0.69	0.65		0.65
	4	0.63	0.63	0.63	0.63	0.56	0.54	0.56	0.61	0.44	0.52	0.57	0.58
timtab1	16												
	4	0.63	0.63	0.63	0.63	0.57	0.57	0.57	0.59	0.44	0.51	0.56	0.59
	4	0.71	0.71	0.71	0.71	0.66	0.67	0.62	0.68	0.60	0.62	0.61	0.61
timtab2	16												
	5	0.66	0.67	0.67	0.69	0.57	0.64	0.59	0.59	0.60	0.62	0.61	0.60
	4	0.96	0.96	0.94	0.94	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
tr12-30	16	0.94	0.94	0.93	0.94	0.77	0.78	0.79	0.79	0.77	0.78	0.79	0.79
	7	0.95	0.95	0.94	0.95	0.89	0.89	0.90	0.90	0.90	0.90	0.90	0.90
	4	0.91	0.90	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91
vpm2	16	0.83	0.86	0.83	0.82	0.75	0.74	0.77	0.76	0.76	0.73	0.78	0.78
	5	0.90	0.91	0.89	0.89	0.89	0.89	0.89	0.89	0.90	0.90	0.91	0.89
arithm.mean		0.75	0.75	0.74	0.75	0.67	0.68	0.65	0.65	0.69	0.66	0.62	0.64
quadr. mean		0.80	0.81	0.80	0.81	0.74	0.74	0.73	0.73	0.74	0.73	0.72	0.72

Table 8.13: Results for small inst. to bbd. conc. μ_{boA}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
	4	0.53	0.53	0.53	0.53								
10teams	16												
	5	0.40	0.53	0.44	0.54								
	4	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
a1c1s1	16	0.81	0.81	0.81	0.82	0.74	0.79	0.75	0.79	0.71	0.80	0.71	0.80
	11	0.84	0.84	0.84	0.85	0.78	0.81	0.79	0.80	0.79	0.82	0.79	0.82
	4	0.92	0.94	0.95	0.95	0.95	0.94	0.93	0.94	0.83	0.81	0.83	0.84
arki001	16	0.89	0.90	0.90	0.90					0.63		0.66	0.66
	8	0.90	0.90	0.91	0.91	0.87	0.88	0.88	0.86	0.75	0.78	0.74	0.74
	4	0.30		0.33									
liu	16												
	8												
	4	0.78	0.78	0.78	0.78	0.76	0.77	0.77	0.77	0.68	0.68	0.68	0.68
manna81	16	0.70	0.70	0.70	0.70	0.58	0.58	0.58	0.58	0.62	0.62	0.57	0.57
	11	0.72	0.71	0.72	0.72	0.60	0.60	0.62	0.62	0.65	0.65	0.65	0.65
	4	0.99	0.99	0.99	0.99	0.92	0.93	0.92	0.92	0.92	0.91	0.92	0.91
mkc	16	0.99	0.99	0.99	0.99	0.91	0.91	0.90	0.91	0.90	0.90	0.90	0.90
	11	0.99	0.99	0.99	0.99	0.91	0.91	0.90	0.91	0.90	0.90	0.92	
	4	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
mod011	16	0.97	0.97	0.98	0.97	0.95	0.96	0.96	0.97	0.95	0.96	0.95	0.96
	12	0.97	0.98	0.98	0.98	0.96	0.97	0.97	0.97	0.97	0.97	0.96	0.97
	4	0.73	0.74	0.73	0.74	0.52	0.52	0.58	0.54		0.52	0.53	0.49
protfold	16	0.46	0.46	0.47	0.47								
	9	0.62	0.62	0.62	0.62		0.43		0.39	0.48	0.38		0.39
	4	0.93	0.93	0.93	0.93	0.72	0.75	0.69	0.80	0.72	0.70	0.76	0.73
roll3000	16	0.82	0.82	0.82	0.84	0.74	0.72	0.78	0.72	0.72	0.67	0.70	0.72
	8	0.91	0.91	0.91	0.91	0.77	0.75	0.79	0.67	0.67	0.73	0.74	0.71
arithm.mean		0.71	0.70	0.71	0.70	0.54	0.56	0.55	0.56	0.55	0.54	0.55	0.54
quadr. mean		0.77	0.77	0.78	0.78	0.67	0.68	0.68	0.68	0.66	0.66	0.67	0.65

Table 8.14: Results for medium inst. to bbd. conc. μ_{boA}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
	4	0.22	0.22	0.21	0.21	0.28			0.28	0.28	0.28	0.31	
air04	16	0.15	0.15	0.15	0.14								
	7	0.19	0.19	0.19			0.22				0.22		
	4	0.17	0.17	0.17	0.16	0.23							
air05	16												
	6	0.16	0.15	0.15									
	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
cap6000	16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	10	1.00	1.00	1.00	1.00	0.98	0.99	1.00	1.00	1.00	1.00	1.00	0.99
	4	0.58	0.61	0.63	0.64	0.47	0.49	0.48	0.50	0.45	0.42	0.42	0.40
dano3mip	16	0.47	0.48	0.52	0.53	0.40	0.40	0.41	0.41	0.36	0.36	0.36	0.37
	11	0.53	0.53	0.56	0.58	0.38	0.40	0.41	0.42	0.37	0.39	0.40	0.42
	4	0.27	0.26	0.25	0.25					0.45	0.45	0.49	0.49
disctom	16	0.25	0.25	0.25	0.22			0.50	0.50				
	6	0.25	0.26	0.25	0.25								
	4			0.92	0.93	0.71	0.73	0.71	0.72	0.72	0.72	0.73	0.72
msc98-ip	16	0.79	0.79	0.81	0.82	0.52	0.53	0.53	0.54	0.53	0.53	0.54	0.53
	18	0.78	0.79	0.81	0.81	0.52	0.52	0.53	0.53		0.52		0.53
	4	0.94	0.96	0.95	0.96	0.26	0.26	0.26		0.24	0.26	0.25	0.24
net12	16	0.80	0.90	0.83	0.91								
	17	0.80	0.88	0.84	0.90	0.06	0.06	0.06	0.06				
	4	0.72	0.72	0.73	0.75	0.60	0.63	0.61	0.68	0.57	0.59	0.58	0.65
seymour	16	0.38	0.42	0.41	0.45			0.34		0.31		0.31	0.33
	8	0.52	0.58	0.56	0.64	0.44	0.46	0.46	0.48	0.39	0.45	0.42	0.43
	4	0.61	0.62	0.61	0.61	0.67	0.67	0.68	0.67	0.68	0.67	0.73	0.73
swath	16												
	7	0.53	0.54		0.56			0.60	0.59	0.62	0.62	0.65	0.64
arithm.mean		0.49	0.50	0.51	0.53	0.32	0.31	0.35	0.35	0.33	0.35	0.34	0.35
quadr. mean		0.58	0.60	0.61	0.64	0.46	0.46	0.49	0.49	0.47	0.48	0.48	0.49

Table 8.15: Results for big inst. to bbd. conc. μ_{boA}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
aflow30a	4	0.99	0.99	0.99	0.99	0.95	0.95	0.91	0.92	0.93	0.93	0.93	0.67
	16	0.93	0.81	0.92	0.82	0.83	0.85	0.71	0.70	0.83	0.79	0.72	0.66
	6	0.94	0.94	0.90	0.85	0.96	0.97	0.70	0.71	0.92	0.92	0.67	0.65
aflow40b	4	0.95	0.94	0.96	0.97	0.97	0.96	0.96	0.93	0.96	0.96	0.95	0.65
	16	0.92	0.92	0.98	0.96	0.89	0.92	0.89	0.89	0.89	0.86	0.85	0.66
	8	0.96	0.91	0.96	0.97	0.94	0.94	0.94	0.83	0.93	0.92	0.91	0.64
danoint	4	0.91	0.91	0.91	0.87	0.90	0.72	0.88	0.83	0.75	0.53	0.72	0.63
	16	0.86	0.61	0.83	0.78			0.46	0.49	0.42			
	6	0.92	0.85	0.86	0.85	0.66	0.73	0.65	0.55	0.44	0.48		
fiber	4	0.88	0.91	0.91	0.93	0.70	0.71	0.84	0.72	0.81	0.84	0.82	0.50
	16	0.63	0.60	0.82	0.66		0.51			0.47			
	5	0.76	0.77	0.80	0.58	0.70	0.69	0.86	0.84	0.75	0.75	0.61	0.61
fixnet6	4	0.97	0.95	0.99	0.98	0.95	0.95	0.95	0.95	0.92	0.95	0.96	0.91
	16	0.91	0.87	0.93	0.85	0.75	0.76	0.81	0.72	0.75	0.76	0.83	0.75
	6	0.62	0.80	0.91	0.87	0.91	0.93	0.64	0.65	0.70	0.83	0.79	0.73
gesa2	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.87	0.94	0.97	0.94	0.82	0.89	0.79	0.85	0.85	0.87	0.80	0.58
	8	0.99	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
gesa2-o	4	0.96	0.96	0.92	0.92	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	16	0.93	0.88	0.92	0.91	0.83	0.88	0.84	0.82	0.83	0.83	0.83	0.81
	8	0.93	0.93	0.92	0.92	1.00	0.96	1.00	0.88	1.00	0.88	1.00	0.88
glass4	4	0.40	0.41	0.33	0.26	0.39	0.50	0.45	0.41	0.48	0.53	0.46	0.47
	16	0.19	0.18			0.16	0.17	0.14	0.13	0.29	0.17	0.20	0.12
	5	0.37	0.26	0.27	0.25	0.42	0.28	0.47	0.36	0.40	0.38	0.40	0.39
harp2	4	0.92	0.92	0.92	0.92	0.92	0.92	0.69	0.69	0.83	0.83	0.63	0.63
	16	0.91	0.94	0.96	0.95	0.83	0.83			0.58	0.58		
	4	0.92	0.92	0.92	0.92	0.92	0.92	0.69	0.69	0.83	0.83	0.63	0.63
modglob	4	0.96	0.97	0.95	0.97	0.88	0.93	0.95	0.89	0.87	0.88	0.89	0.83
	16	0.84	0.64	0.80	0.69	0.76	0.78	0.55	0.60	0.79	0.73	0.56	0.58
	5	0.90	0.91	0.91	0.85	0.93	0.88	0.93	0.79	0.92	0.87	0.62	0.77
noswot	4	0.88	0.67	0.87	0.67	0.81	0.75	0.52	0.49	0.87	0.68	0.45	0.44
	16									0.36			
	4	0.91	0.67	0.89	0.70	0.85	0.76	0.52	0.51	0.66	0.69	0.45	0.44
opt1217	4	1.00	1.00	1.00	1.00	1.00	1.00	0.64	0.64	0.85	0.85	0.64	0.64
	16	0.78	0.79		0.98	0.98	0.98			0.56	0.56		
	3	1.00	1.00	0.89	0.79	1.00	1.00	0.71	0.71	1.00	1.00	0.71	0.71
p2756	4	0.94	0.92	0.95	0.97	0.92	0.89	0.87	0.86	0.92	0.74	0.92	0.66
	16	0.94	0.96	0.93	0.94	0.47	0.48	0.34	0.38	0.55	0.47	0.45	0.42
	7	0.92	0.87	0.87	0.85	0.82	0.83	0.73	0.68	0.85	0.68	0.79	0.61
pk1	4												
	16												
	3												
pp08a	4	0.96	0.99	0.96	0.96	0.83	0.94	0.67	0.63	0.88	0.83	0.59	0.59
	16	0.79	0.79	0.79	0.66	0.75	0.69			0.65	0.69		
	4	0.90	0.65	0.89	0.63	0.84	0.86	0.73	0.79	0.71	0.77	0.69	0.70
pp08aCUTS	4	1.00	1.00	1.00	1.00	0.86	0.77	0.71	0.77	0.89	0.77	0.91	0.79
	16	0.88	0.69	0.88	0.88	0.46	0.60	0.58	0.51	0.62	0.74	0.46	0.46
	5	1.00	1.00	1.00	1.00	0.89	0.77	0.62	0.77	0.89	0.77	0.80	0.80
qiu	4	0.98	0.94	1.00	1.00	0.89	0.94	1.00	0.95	0.93	0.89	1.00	0.91
	16	0.90	0.60	0.85	0.69								0.58
	7	0.94	0.91	0.93	0.80	0.65	0.60	0.78	0.71	0.52	0.58		0.41
timtab1	4	0.76	0.75	0.79	0.77	0.48	0.59	0.64	0.43	0.58	0.57	0.55	0.40
	16												
	4	0.76	0.77	0.79	0.77	0.62	0.59	0.63	0.54	0.45	0.64	0.52	0.46
timtab2	4	0.72	0.71	0.75	0.73	0.58	0.65	0.58	0.72	0.63	0.58	0.50	0.75
	16												
	5	0.78	0.65	0.65	0.60	0.59	0.65	0.83	0.63	0.59	0.60	0.51	0.56
tr12-30	4	1.00	1.00	0.99	0.98	0.91	0.93	0.80	0.89	0.90	0.93	0.89	0.89
	16	0.94	0.82	0.97	0.90	0.63	0.72	0.63	0.59	0.63	0.72	0.59	0.60
	7	0.91	0.92	0.90	0.83	0.89	0.90	0.82	0.84	0.82	0.86	0.73	0.62
vpm2	4	0.93	0.90	0.93	0.93	0.91	0.91	0.91	0.90	0.90	0.91	0.91	0.91
	16	0.92	0.77	0.88	0.59	0.69	0.72	0.54	0.53	0.58	0.91	0.55	0.54
	5	0.87	0.93	0.89	0.87	0.84	0.85	0.75	0.73	0.81	0.80	0.71	0.68
arithm.mean		0.79	0.75	0.77	0.74	0.68	0.70	0.61	0.59	0.68	0.65	0.56	0.53
quadr. mean		0.84	0.81	0.84	0.80	0.76	0.76	0.69	0.67	0.73	0.72	0.65	0.61

Table 8.16: Results for small inst. to bbd. conc. μ_{blB}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
10teams	4	0.97	0.93	0.95	0.95								
	16												
	5	0.85	0.73	0.82	0.74								
a1c1s1	4	0.94	0.96	0.94	0.95	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
	16	0.87	0.82	0.88	0.88	0.83	0.80	0.81	0.76	0.79	0.79	0.81	0.79
	11	0.87	0.81	0.86	0.62	0.82	0.79	0.80	0.85	0.77	0.72	0.67	0.64
arki001	4	0.92	0.87	0.96	0.93	0.87	0.86	0.91	0.91	0.83	0.64	0.79	0.61
	16	0.91	0.75	0.93	0.85					0.44		0.40	0.32
	8	0.92	0.85	0.97	0.93	0.65	0.67	0.65	0.65	0.50	0.56	0.54	0.49
liu	4	0.42		0.34									
	16												
	8												
manna81	4	0.99	0.99	0.96	0.98	0.96	0.97	0.98	0.94	0.98	0.96	0.96	0.98
	16	0.92	0.92	0.92	0.89	0.94	0.91	0.92	0.93	0.94	0.92	0.94	0.95
	11	0.93	0.93	0.89	0.88	0.94	0.90	0.91	0.90	0.87	0.91	0.88	0.88
mkc	4	0.98	0.99	0.98	0.99	0.87	0.90	0.77	0.75	0.90	0.83	0.69	0.64
	16	0.92	0.91	0.94	0.93	0.74	0.76	0.71	0.72	0.70	0.69	0.57	0.46
	11	0.84	0.79	0.83	0.80	0.74	0.77	0.58	0.60	0.65	0.59	0.52	
mod011	4	0.89	0.89	0.96	0.96	0.80	0.83	0.85	0.81	0.71	0.70	0.59	0.73
	16	0.67	0.68	0.59	0.72	0.38	0.40	0.39	0.42	0.29	0.40	0.38	0.34
	12	0.70	0.68	0.55	0.54	0.48	0.42	0.46	0.51	0.37	0.38	0.35	0.42
protfold	4	0.91	0.79	0.97	0.96	0.61	0.66	0.46	0.50		0.56	0.47	0.67
	16	0.84	0.71	0.82	0.67								
	9	0.92	0.76	0.88	0.77		0.50		0.64	0.40	0.46		0.53
roll3000	4	0.90	0.89	0.91	0.89	0.86	0.66	0.55	0.59	0.81	0.77	0.85	0.73
	16	0.74	0.54	0.75	0.64	0.53	0.51	0.51	0.50	0.55	0.50	0.53	0.48
	8	0.82	0.78	0.86	0.80	0.72	0.60	0.70	0.57	0.70	0.67	0.67	0.63
arithm.mean		0.76	0.70	0.76	0.71	0.50	0.51	0.47	0.50	0.48	0.48	0.46	0.45
quadr. mean		0.82	0.77	0.82	0.78	0.63	0.62	0.60	0.61	0.60	0.59	0.57	0.56

Table 8.17: Results for medium inst. to bbd. conc. μ_{blB}

Algorithm		HYPERROW DECOMP.				HYPERCOL DECOMPOSING							
Metis method		PKW		RECURSIVE		PKW				RECURSIVE			
Dummy ratio		0%	20%	0%	20%	0%		20%		0%		20%	
Weighting scheme		un.	un.	un.	un.	un.	ps.	un.	ps.	un.	ps.	un.	ps.
Instance	nBl												
air04	4	0.95	0.89	0.91	0.93	0.55			0.60	0.50	0.63	0.41	
	16	0.68	0.58	0.65	0.61								
	7	0.80	0.80	0.66			0.52				0.47		
air05	4	0.91	0.80	0.91	0.80	0.46							
	16												
	6	0.86	0.68	0.46									
cap6000	4	1.00	1.00	1.00	1.00	0.88	0.89	0.80	0.81	0.79	0.79	0.55	0.55
	16	0.97	0.97	1.00	0.99	0.80	0.82	0.74	0.73	0.68	0.68	0.47	0.47
	10	0.98	0.84	0.91	0.83	0.75	0.75	0.70	0.69	0.65	0.66	0.45	0.45
dano3mip	4	0.41	0.36	0.34	0.35	0.39	0.56	0.40	0.54	0.52	0.48	0.15	0.25
	16	0.10	0.10	0.10	0.09	0.30	0.32	0.30	0.34	0.23	0.30	0.19	0.26
	11	0.15	0.15	0.14	0.10	0.24	0.30	0.28	0.36	0.24	0.30	0.21	0.21
disctom	4	0.96	0.96	0.96	1.00					0.14	0.15	0.12	0.12
	16	0.80	0.73	0.68	0.72			0.43	0.39				
	6	0.84	0.82	0.50	0.63								
msc98-ip	4			0.29	0.25	0.63	0.62	0.62	0.61	0.59	0.60	0.58	0.63
	16	0.26	0.25	0.20	0.14	0.43	0.39	0.37	0.37	0.36	0.38	0.33	0.37
	18	0.24	0.21	0.18	0.15	0.41	0.38	0.38	0.35		0.34		0.37
net12	4	0.95	0.92	0.91	0.92	0.43	0.41	0.42		0.56	0.50	0.58	0.53
	16	0.84	0.60	0.76	0.58								
	17	0.87	0.80	0.72	0.52	0.25	0.21	0.25	0.21				
seymour	4	0.62	0.66	0.64	0.53	0.63	0.63	0.58	0.67	0.55	0.51	0.53	0.68
	16	0.36	0.26	0.26	0.22			0.26		0.27		0.27	0.25
	8	0.53	0.39	0.37	0.28	0.41	0.37	0.33	0.33	0.37	0.35	0.37	0.48
swath	4	0.50	0.60	0.38	0.73	0.36	0.36	0.37	0.37	0.35	0.29	0.24	0.24
	16												
	7	0.64	0.77		0.26			0.23	0.23	0.22	0.22	0.24	0.19
arithm.mean		0.60	0.56	0.52	0.47	0.29	0.28	0.28	0.28	0.26	0.28	0.21	0.22
quadr. mean		0.69	0.65	0.61	0.58	0.41	0.40	0.37	0.39	0.37	0.38	0.30	0.32

Table 8.18: Results for big inst. to bbd. conc. μ_{blB}

number of blocks requested				2			4			8			16			32			64			128			256		
instance	#rows	#cols	#nonz		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.		F&H	Dec.
25fv47	821	1571	10400		0.97	0.91		0.88	0.855		0.65	0.762		0.73	0.611		0.69	0.524		0.58	0.473		0.45	0.369		0.27	0.285
80bau3b	2262	9799	21002		0.78	0.937		0.62	0.91		0.6	0.88		0.59	0.851		0.56	0.816		0.52	0.77		0.43	0.719		0.41	0.642
adlittle	56	97	383		0.95	0.83		0.59	0.74		0.5	0.633		0.47	0.573		0.35	0.421		0.18	0.273		0.1	0.11		0.03	0.033
afiro	27	32	83		1	0.824		0.77	0.754		0.56	0.616		0.51	0.43		0.33	0.362		0.16	0.166		0	0		0	0
agg	488	163	2410		0.8	0.886		0.74	0.734		0.65	0.474		0.44	0.321		0.27	0.255		0.19	0.202		0.19	0.197		0.18	0.178
agg2	516	302	4284		1	0.852		0.89	0.69		0.71	0.54		0.76	0.48		0.59	0.38		0.53	0.302		0.48	0.25		0.45	0.18
agg3	516	302	4300		1	0.865		1	0.694		0.83	0.534		0.78	0.471		0.66	0.38		0.57	0.318		0.53	0.233		0.5	0.179
bandm	305	472	2494		0.91	0.938		0.77	0.831		0.71	0.782		0.66	0.712		0.58	0.566		0.43	0.516		0.34	0.443		0.34	0.379
beaconfd	173	262	3375		0.73	0.796		0.49	0.742		0.48	0.717		0.46	0.689		0.44	0.636		0.44	0.471		0.38	0.341		0.28	0.278
blend	74	83	491		0.75	0.831		0.75	0.706		0.62	0.622		0.46	0.411		0.29	0.343		0.18	0.251		0	0.13		0	0.075
bnl1	643	1175	5121		0.84	0.906		0.75	0.847		0.69	0.802		0.62	0.746		0.58	0.668		0.52	0.63		0.48	0.527		0.37	0.453
bnl2	2324	3489	13999		0.87	0.928		0.83	0.866		0.78	0.83		0.71	0.787		0.66	0.75		0.6	0.713		0.58	0.673		0.52	0.614
boeing1	351	384	3485		1	0.897		0.73	0.866		0.69	0.788		0.66	0.705		0.62	0.59		0.57	0.469		0.44	0.312		0.34	0.186
boeing2	166	143	1196		0.65	0.768		0.52	0.635		0.39	0.538		0.34	0.401		0.32	0.341		0.25	0.236		0.17	0.079		0.16	0.027
bore3d	233	315	1429		0.85	0.916		0.74	0.8		0.65	0.733		0.62	0.682		0.52	0.632		0.49	0.577		0.41	0.492		0.38	0.408
brandy	220	249	2148		0.8	0.632		0.66	0.582		0.57	0.523		0.49	0.454		0.42	0.386		0.4	0.349		0.34	0.31		0.31	0.24
capri	271	353	1767		0.8	0.889		0.62	0.811		0.59	0.734		0.5	0.616		0.42	0.554		0.34	0.487		0.27	0.409		0.17	0.306
cycle	1903	2857	20720		0.91	0.961		0.8	0.942		0.77	0.853		0.7	0.718		0.66	0.636		0.58	0.589		0.55	0.534		0.41	0.481
czprob	929	3523	10669		0.91	0.96		0.61	0.963		0.57	0.963		0.48	0.957		0.41	0.954		0.43	0.941		0.44	0.925		0.43	0.903
d2q06c	2171	5167	32417		0.94	0.941		0.89	0.887		0.8	0.845		0.79	0.769		0.75	0.697		0.66	0.594		0.52	0.501		0.41	0.435
d6cube	415	6184	37704		0.32	0.518		0.06	0.332		0.04	0.195		0.04	0.132		0.02	0.087		0	0.061		0	0.025		0	0.018
degen2	444	534	3978		0.71	0.837		0.61	0.778		0.5	0.725		0.45	0.64		0.37	0.549		0.29	0.467		0.21	0.398		0.13	0.298
degen3	1503	1818	24646		0.78	0.882		0.6	0.854		0.54	0.793		0.45	0.744		0.34	0.692		0.3	0.632		0.25	0.542		0.2	0.445
dfl001	6071	12230	41873		1	0.91		0.83	0.853		0.76	0.828		0.67	0.786		0.6	0.717		0.57	0.666		0.49	0.616		0.44	0.562
e226	223	282	2578		0.81	0.883		0.71	0.708		0.63	0.634		0.59	0.601		0.46	0.532		0.39	0.448		0.35	0.354		0.34	0.221
etamacro	400	688	2409		0.92	0.858		0.69	0.753		0.62	0.68		0.56	0.58		0.45	0.509		0.4	0.452		0.29	0.379		0.18	0.294
fffff800	524	854	6227		0.83	0.864		0.54	0.789		0.45	0.749		0.34	0.712		0.33	0.653		0.28	0.587		0.28	0.526		0.27	0.474
finnis	497	614	2310		1	0.923		0.87	0.898		0.8	0.869		0.69	0.802		0.66	0.758		0.61	0.702		0.56	0.614		0.45	0.523
fit1p	627	1677	9868		0.6	0.986		0.24	0.985		0.12	0.983		0.12	0.983		0	0.977		0	0.974		0	0.945		0	0.736
fit2p	3000	13525	50284		0.14	0.998		0.24	0.998		0.02	0.998		0.02	0.997		0.02	0.995		0	0.993		0	0.987		0	0.98
forplan	161	421	4563		0.96	0.24		0.88	0.177		0.84	0.151		0.68	0.134		0.62	0.123		0.54	0.116		0.42	0.104		0.35	0.089
ganges	1309	1681	6912		1	0.978		0.92	0.953		0.88	0.907		0.78	0.848		0.74	0.786		0.66	0.675		0.57	0.636		0.48	0.593
gfrd-pnc	616	1092	2377		0.97	0.99		0.87	0.973		0.84	0.95		0.82	0.911		0.79	0.857		0.72	0.794		0.65	0.709		0.52	0.537
greenbea	2392	5405	30877		0.96	0.942		0.83	0.891		0.73	0.855		0.61	0.8		0.57	0.719		0.47	0.689		0.38	0.65		0.26	0.61
greenbeb	2392	5405	30877		1	0.941		0.9	0.893		0.79	0.855		0.66	0.807		0.62	0.729		0.51	0.687		0.41	0.648		0.28	0.609
grow15	300	645	5620		0.97	0.936		0.72	0.822		0.68	0.627		0.6	0.309		0.55	0.053		0	0.042		0	0.031		0	0.021
grow22	440	946	8252		0.66	0.977		0.64	0.886		0.61	0.733		0.52	0.516		0.39	0.062		0.18	0.051		0	0.031		0	0.021
grow7	140	301	2612		0.76	0.872		0.53	0.619		0.49	0.239		0.3	0.066		0	0.041		0	0.047		0	0.024		0	0.025
israel	174	142	2269		0.78	0.731		0.64	0.711		0.53	0.659		0.57	0.605		0.54	0.49		0.33	0.385		0.25	0.322		0.21	0.206
kb2	43	41	286		0.55	0.677		0.52	0.501		0.34	0.404		0.24	0.331		0.18	0.282		0.19	0.101		0	0.068		0	0.031
lotfi	153	308	1078		0.92	0.896		0.65	0.832		0.61	0.719		0.57	0.651		0.49	0.593		0.52	0.525		0.47	0.428		0.4	0.345
maros	846	1443	9614		1	0.94		0.95	0.826		0.87	0.75		0.82	0.622		0.79	0.533		0.71	0.468		0.51	0.412		0.36	0.363
nesm	662	2923	13288		0.87	0.975		0.71	0.89		0.64	0.732		0.57	0.558		0.5	0.441		0.46	0.414		0.44	0.396		0.37	0.386
perold	625	1376	6018		1	0.897		0.83	0.824		0.73	0.716		0.63	0.678		0.55	0.544		0.36	0.481		0.28	0.414		0.21	0.315

Table 8.19: comparison with results of Ferris and Horn (part1)

number of blocks requested				2		4		8		16		32		64		128		256	
instance	#rows	#cols	#nonz	F&H	Dec.	F&H	Dec.	F&H	Dec.	F&H	Dec.	F&H	Dec.	F&H	Dec.	F&H	Dec.	F&H	Dec.
pilot	1441	3652	43167	0.75	0.833	0.62	0.715	0.51	0.637	0.37	0.583	0.23	0.528	0.21	0.438	0.15	0.371	0.14	0.306
pilot.ja	940	1988	14698	0.97	0.831	0.74	0.771	0.62	0.742	0.55	0.677	0.46	0.625	0.39	0.564	0.35	0.502	0.29	0.432
pilot.we	722	2789	9126	1	0.93	0.8	0.865	0.72	0.803	0.6	0.719	0.56	0.625	0.4	0.497	0.29	0.402	0.23	0.326
pilot4	410	1000	5141	1	0.885	0.99	0.819	0.77	0.624	0.6	0.57	0.49	0.553	0.44	0.477	0.39	0.453	0.21	0.374
pilot87	2030	4883	73152	0.87	0.745	0.68	0.629	0.51	0.588	0.39	0.55	0.37	0.504	0.23	0.446	0.19	0.408	0.16	0.366
pilotnov	975	2172	13057	0.86	0.872	0.71	0.831	0.65	0.786	0.59	0.689	0.52	0.618	0.44	0.552	0.38	0.501	0.29	0.428
recipe	91	180	663	1	0.849	0.93	0.732	0.9	0.602	0.76	0.559	0.46	0.503	0.26	0.356	0.01	0.253	0	0.076
sc105	105	103	280	0.95	0.947	0.84	0.842	0.76	0.748	0.61	0.636	0.45	0.456	0.32	0.296	0.24	0.131	0.17	0
sc205	205	203	551	1	0.974	0.94	0.921	0.85	0.845	0.72	0.748	0.62	0.637	0.47	0.436	0.37	0.321	0.24	0
sc50a	50	48	130	1	0.878	0.88	0.792	0.78	0.646	0.58	0.457	0.41	0.293	0.25	0.125	0	0	0	0
sc50b	50	48	118	0.92	0.859	0.79	0.792	0.66	0.686	0.56	0.559	0.44	0.369	0.33	0	0	0	0	0
scagr25	471	500	1554	0.97	0.987	0.91	0.961	0.86	0.912	0.85	0.816	0.75	0.746	0.69	0.688	0.62	0.638	0.55	0.495
scagr7	129	140	420	0.87	0.954	0.79	0.849	0.71	0.773	0.63	0.698	0.56	0.649	0.52	0.527	0.46	0.398	0.33	0.198
scfxm1	330	457	2589	1	0.952	0.87	0.851	0.75	0.721	0.65	0.651	0.59	0.577	0.48	0.514	0.38	0.43	0.32	0.326
scfxm2	660	914	5183	1	0.993	1	0.945	0.99	0.846	0.83	0.721	0.72	0.651	0.64	0.575	0.56	0.506	0.45	0.429
scfxm3	990	1371	7777	1	0.979	0.95	0.937	0.89	0.898	0.77	0.8	0.74	0.692	0.62	0.615	0.53	0.55	0.47	0.472
scorpion	388	358	1426	1	0.971	1	0.938	0.97	0.861	0.94	0.818	0.92	0.787	0.87	0.689	0.66	0.459	0.45	0.406
scrs8	490	1169	3182	0.91	0.938	0.8	0.886	0.78	0.835	0.69	0.784	0.62	0.719	0.5	0.683	0.46	0.633	0.43	0.554
scsd1	77	760	2388	0	0.844	0	0.452	0	0.106	0	0	0	0	0	0	0	0	0	0
scsd6	147	1350	4316	0.25	0.863	0.64	0.592	0.17	0.477	0.04	0.159	0.02	0	0.04	0	0	0	0	0
scsd8	397	2750	8584	0.75	0.974	0.92	0.91	0.74	0.78	0.36	0.531	0.1	0.183	0.04	0.03	0.04	0.012	0	0
sctap1	300	480	1692	0.9	0.94	0.81	0.91	0.76	0.85	0.62	0.77	0.55	0.682	0.46	0.587	0.38	0.412	0.15	0.288
sctap2	1090	1880	6714	1	0.964	0.97	0.928	0.91	0.899	0.86	0.871	0.8	0.811	0.72	0.783	0.59	0.706	0.44	0.611
sctap3	1480	2480	8874	1	0.974	0.94	0.943	0.89	0.91	0.83	0.879	0.77	0.848	0.71	0.783	0.61	0.717	0.53	0.634
seba	515	1028	4352	0.26	0.986	0.22	0.971	0.21	0.96	0.2	0.928	0.2	0.9	0.19	0.883	0.17	0.822	0.14	0.749
share1b	117	225	1151	0.78	0.945	0.71	0.773	0.65	0.532	0.53	0.486	0.36	0.349	0.12	0.251	0.13	0.215	0.09	0.167
share2b	96	79	694	0.89	0.898	0.75	0.678	0.72	0.488	0.32	0.288	0.21	0.214	0.16	0.115	0.19	0.046	0.19	0.013
shell	536	1775	3556	0.65	0.956	0.62	0.908	0.6	0.881	0.53	0.83	0.47	0.799	0.44	0.728	0.41	0.666	0.36	0.564
ship04l	402	2118	6332	1	0.768	0.46	0.751	0.19	0.747	0.21	0.733	0.16	0.722	0.11	0.702	0.11	0.677	0.08	0.619
ship04s	402	1458	4352	0.56	0.825	0.47	0.774	0.42	0.749	0.41	0.733	0.38	0.725	0.36	0.696	0.35	0.634	0.34	0.48
ship08l	778	4283	12802	1	0.886	1	0.858	1	0.768	0.73	0.76	0.53	0.751	0.47	0.738	0.4	0.723	0.33	0.706
ship08s	778	2387	7114	0.88	0.91	0.89	0.901	0.82	0.784	0.76	0.753	0.73	0.741	0.7	0.718	0.66	0.68	0.64	0.606
ship12l	1151	5427	16170	1	0.909	0.7	0.901	0.68	0.792	0.59	0.747	0.5	0.746	0.48	0.738	0.44	0.731	0.42	0.692
ship12s	1151	2763	8178	1	0.909	0.98	0.907	0.91	0.849	0.9	0.75	0.87	0.73	0.85	0.7	0.79	0.667	0.79	0.631
sierra	1227	2036	7302	1	0.946	1	0.921	0.88	0.894	0.79	0.843	0.75	0.795	0.77	0.73	0.69	0.668	0.52	0.615
stair	356	467	3856	0.97	0.928	0.85	0.808	0.71	0.667	0.48	0.529	0.34	0.47	0.23	0.418	0.18	0.356	0.15	0.273
standata	359	1075	3031	0.9	0.938	0.77	0.918	0.75	0.841	0.68	0.775	0.66	0.706	0.59	0.674	0.57	0.651	0.54	0.629
standgub	361	1184	3139	0.87	0.929	0.69	0.88	0.67	0.831	0.63	0.77	0.62	0.705	0.58	0.672	0.53	0.66	0.52	0.623
standmps	467	1075	3679	0.86	0.954	0.76	0.882	0.63	0.818	0.56	0.777	0.52	0.749	0.43	0.74	0.46	0.697	0.43	0.485
stocfor1	117	111	447	1	0.919	0.75	0.784	0.65	0.69	0.56	0.574	0.49	0.507	0.43	0.327	0.3	0.158	0.13	0.077
stocfor2	2157	2031	8343	0.9	0.994	0.8	0.98	0.73	0.962	0.73	0.921	0.69	0.836	0.63	0.729	0.55	0.65	0.44	0.602
tuff	333	587	4520	0.78	0.783	0.43	0.726	0.39	0.697	0.32	0.655	0.3	0.603	0.31	0.558	0.27	0.519	0.2	0.322
vtp.base	198	203	908	0.95	0.924	0.8	0.821	0.67	0.748	0.62	0.665	0.53	0.612	0.49	0.546	0.49	0.451	0.36	0.236
wood1p	244	2594	70215	0.05	0.593	0.03	0.512	0.02	0.468	0.02	0.434	0.01	0.336	0.01	0.249	0.01	0.168	0.01	0.117
woodw	1098	8405	37474	1	0.954	0.67	0.95	0.19	0.852	0.14	0.712	0.1	0.61	0.1	0.475	0.05	0.376	0.05	0.282
quadratic mean				0.87	0.89	0.76	0.82	0.67	0.75	0.59	0.67	0.52	0.61	0.45	0.55	0.39	0.49	0.33	0.42

Table 8.20: comparison with results of Ferris and Horn (part2)

number of blocks				2	4	8	16	32	64	128	256
instance	#rows	#cols	#nonz								
"50v-10"	233	2013	2745	0.93	0.90	0.88	0.83	0.77	0.73	0.57	
"a1c1s1"	3312	3648	10178	0.98	0.94	0.92	0.91	0.88	0.81	0.77	0.72
"acc-tight4"	3285	1620	17073	0.86	0.79	0.76	0.72	0.71	0.68	0.59	
"acc-tight5"	3052	1339	16134	0.84	0.75	0.68	0.61	0.56	0.51		
"acc-tight6"	3047	1335	16108	0.85	0.76	0.68	0.61	0.55	0.51		
"aflow40b"	1442	2728	6783	0.98	0.97	0.97	0.95	0.95	0.95	0.93	0.93
"air04"	823	8904	72965	0.36	0.29	0.24	0.18				
"ash608gpia-3col"	24748	3651	74244								
"atm20-100"	4380	6480	58878	0.97	0.96	0.95	0.92	0.88	0.85	0.81	0.75
"b2c1s1"	3904	3872	11408	0.99	0.98	0.97	0.97	0.96	0.94	0.89	0.86
"beasleyC3"	1750	2500	5000	0.98	0.92	0.88	0.84	0.80	0.75	0.72	
"berlin_5_8_0"	1532	1083	4507	0.98	0.93	0.85	0.78	0.71	0.65		
"bg512142"	1307	792	3953	0.85	0.63	0.48	0.32				
"biella1"	1203	7328	71489	0.91	0.90	0.89	0.85	0.81	0.74		
"bienst2"	576	505	2184	0.99	0.99	0.96	0.93	0.90	0.87	0.83	0.73
"binkar10_1"	1026	2298	4496	0.96	0.95	0.95	0.94	0.93	0.91	0.88	0.83
"bnatt350"	4923	3150	19061	0.96	0.94	0.94	0.93	0.92	0.91	0.88	0.84
"bnatt400"	5614	3600	21698								
"cov1075"	637	120	14280	0.82	0.63	0.43	0.33				
"csched007"	351	1758	6379	0.86	0.63	0.47	0.36				
"csched008"	351	1536	5687	0.85	0.64	0.52	0.36				
"csched010"	351	1758	6376	0.29	0.26	0.22	0.21	0.22	0.20		
"dano3mip"	3202	13873	79655	0.76	0.71	0.68	0.59	0.53	0.49	0.49	
"danoint"	664	521	3232	0.86	0.83	0.83	0.75	0.73	0.69	0.61	
"dfn-gwin-UUM"	158	938	2632	0.73	0.64	0.62					
"dg012142"	6310	2080	14795	0.96	0.90	0.85	0.82	0.78	0.72	0.67	
"eil33-2"	32	4516	44243								
"eilB101"	100	2818	24120								
"enlight13"	169	338	962	0.91	0.79	0.67	0.45				
"enlight14"	196	392	1120	0.91	0.82	0.71	0.50				
"enlight15"	225	450	1290	0.91	0.84	0.70	0.51				
"enlight16"	256	512	1472	0.93	0.85	0.71	0.59				
"enlight9"	81	162	450	0.85	0.73	0.46					
"f2000"	10500	4000	29500	0.58	0.50	0.47	0.46	0.44	0.43	0.34	0.32
"g200x740i"	940	1480	2960	0.99	0.97	0.96	0.93	0.89	0.85	0.79	0.77
"germany50-DBM"	2526	8189	24479	0.93	0.93	0.93	0.93	0.90	0.84	0.75	0.63
"glass4"	396	322	1815	0.95	0.94	0.93	0.91	0.89	0.82	0.78	
"gmu-35-40"	424	1205	4843	0.70	0.67	0.61	0.53				
"gmu-35-50"	435	1919	8643	0.62	0.59	0.50	0.42				
"go19"	441	441	1885	0.92	0.86	0.75	0.62	0.49			
"hanoi5"	16399	3862	39718	0.98	0.94	0.80	0.78	0.73	0.67	0.58	0.44
"harp2"	112	2993	5840	0.68	0.68	0.67	0.55				
"ic97_potential"	1046	728	3138	0.98	0.97	0.94	0.92	0.86	0.80	0.75	0.70
"iis-100-0-cov"	3831	100	22986	0.17							
"iis-bupa-cov"	4803	345	38392	0.50	0.34	0.28	0.24				
"iis-pima-cov"	7201	768	71941	0.71	0.45	0.36	0.29	0.25			
"janos-us-DDM"	760	2184	6384	0.90	0.89	0.87	0.85	0.79	0.69	0.58	
"k16x240"	256	480	960	0.95	0.94	0.94	0.93	0.91	0.93	0.93	
"lectsched-4-obj"	14163	7901	82428	0.96	0.96	0.96	0.96	0.96	0.95	0.95	0.94
"liu"	2178	1156	10626	0.92	0.92	0.92	0.92	0.92	0.91	0.88	0.86
"lotsize"	1920	2985	6565	1.00	0.99	0.98	0.96	0.91	0.86	0.82	0.77
"lrsa120"	14521	3839	39956	0.97	0.97	0.97	0.96	0.96	0.96	0.95	0.95
"m100n500k4r1"	100	500	2000	0.21	0.20						
"macrophage"	3164	2260	9492	0.99	0.99	0.98	0.97	0.95	0.92	0.89	0.85
"markshare_5_0"	5	45	203								
"maxgasflow"	7160	7437	19717	1.00	1.00	0.99	0.99	0.99	0.98	0.96	0.93
"mc11"	1920	3040	6080	0.99	0.98	0.97	0.95	0.93	0.89	0.84	0.79
"mcsched"	2107	1747	8088	0.97	0.95	0.94	0.94	0.93	0.92	0.89	0.88
"methanosarcina"	14604	7930	43812	0.99	0.99	0.99	0.99	0.98	0.98	0.97	0.96
"mik-250-1-100-1"	151	251	5351			0.30	0.31				
"mine-166-5"	8429	830	19412	0.84	0.53	0.42	0.32	0.28			
"mine-90-10"	6270	900	15407	0.97	0.83	0.55	0.43	0.34	0.24		
"mkc"	3411	5325	17038	0.99	0.99	0.98	0.98	0.97	0.95	0.92	0.89
"msc98-ip"	15850	21143	92918	0.96	0.94	0.92	0.90	0.85	0.83	0.80	0.78
"n3-3"	2425	9028	35380	0.90	0.89	0.87	0.85	0.82	0.80	0.66	0.56
"n3700"	5150	10000	20000	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.97
"n3705"	5150	10000	20000	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.97
"n370a"	5150	10000	20000	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.97

Table 8.21: results for miplib2010 to arrowhead (part 1)

number of blocks				2	4	8	16	32	64	128	256
instance											
"n4-3"	1236	3596	14036	0.91	0.90	0.88	0.86	0.83	0.80	0.68	0.61
"n9-3"	2364	7644	30072	0.93	0.93	0.91	0.88	0.88	0.85	0.71	0.60
"nag"	5840	2884	26499	0.95	0.93	0.91	0.90	0.89	0.88	0.85	0.82
"neos-1109824"	28979	1520	89528	0.10	0.16						
"neos-1112782"	2115	4140	8145	0.98	0.98	0.98	0.98	0.98	0.98	0.96	0.96
"neos-1112787"	1680	3280	6440	0.98	0.98	0.98	0.98	0.98	0.98	0.96	0.94
"neos-1171692"	4239	1638	42945	0.98	0.92	0.78	0.79	0.50			
"neos-1171737"	4179	2340	58620	0.99	0.96	0.90	0.82	0.73	0.52		
"neos-1224597"	3276	3395	25090	0.94	0.94	0.86	0.84	0.80	0.71	0.56	0.45
"neos-1225589"	675	1300	2525	0.97	0.97	0.96	0.97	0.96	0.95	0.94	0.94
"neos-1311124"	1643	1092	7140	0.98	0.96	0.96	0.91	0.80	0.59		
"neos-1337307"	5687	2840	30799	0.98	0.95	0.93	0.92	0.89	0.88	0.84	0.76
"neos-1396125"	1494	1161	5511	0.90	0.87	0.83	0.82	0.77	0.72	0.70	
"neos-1426635"	796	520	3400	0.96	0.96	0.90	0.80	0.59			
"neos-1426662"	1914	832	8048	0.97	0.94	0.98	0.89				
"neos-1436709"	1417	676	6214	0.96	0.93	0.94	0.90	0.64			
"neos-1440225"	330	1285	14168	0.67	0.51	0.36					
"neos-1440460"	989	468	4302	0.93	0.94	0.87					
"neos-1442119"	1524	728	6692	0.96	0.97	0.92	0.90	0.63			
"neos-1442657"	1310	624	5736	0.96	0.92	0.84	0.70	0.61			
"neos15"	552	792	1766	0.99	0.96	0.89	0.82	0.75	0.67	0.59	
"neos-1601936"	3131	4446	72500	0.71	0.64	0.60	0.59	0.49	0.46	0.37	
"neos-1605061"	3474	4111	93483	0.82	0.54	0.44	0.38	0.34	0.28	0.25	
"neos-1605075"	3467	4173	91377	0.81	0.52	0.45	0.38	0.34	0.28		
"neos-1616732"	1999	200	3998	0.38	0.33	0.30	0.22				
"neos-1620770"	9296	792	19292	1.00	0.92	0.80	0.70	0.20			
"neos16"	1018	377	2801	0.95	0.93	0.91	0.90	0.86	0.78		
"neos18"	11402	3312	24614	0.98	0.98	0.97	0.97	0.96	0.94	0.94	0.91
"neos-506422"	6811	2527	31815	0.96	0.94	0.92	0.91	0.91	0.90	0.90	0.89
"neos-555424"	2676	3815	15667	0.99	0.98	0.91	0.89	0.87	0.84	0.79	0.71
"neos-686190"	3664	3660	18085	0.95	0.94	0.94	0.93	0.94	0.94	0.93	0.93
"neos-777800"	479	6400	32000	0.48	0.42	0.39	0.37	0.32			
"neos-785912"	1714	1380	16610	0.90	0.87	0.86	0.84	0.78	0.74	0.70	0.63
"neos788725"	433	352	4912	0.82	0.69						
"neos-807456"	840	1635	4905	0.67	0.55	0.50	0.46	0.39	0.36		
"neos-820146"	830	600	3225	0.81	0.79	0.76	0.76	0.72	0.64		
"neos-820157"	1015	1200	4875	0.91	0.88	0.87	0.83	0.78	0.74	0.72	
"neos-824695"	9576	23970	72590	1.00	0.99	0.99	0.98	0.97	0.96	0.92	0.86
"neos-826650"	2414	5912	20440	0.93	0.93	0.93	0.93	0.92	0.79	0.63	0.51
"neos-826694"	6904	16410	59268	0.98	0.97	0.97	0.96	0.95	0.92	0.86	0.81
"neos-826812"	6844	15864	53808	0.98	0.98	0.98	0.97	0.96	0.94	0.89	0.84
"neos-826841"	2354	5516	18460	0.95	0.95	0.95	0.95	0.95	0.84	0.71	0.56
"neos-847302"	609	737	9566	0.71	0.60	0.48	0.41	0.32			
"neos-849702"	1041	1737	19308	0.70	0.65	0.59	0.55	0.53	0.31		
"neos858960"	132	160	2770	0.65	0.65	0.64	0.61	0.55			
"neos-911880"	83	888	2568	0.62	0.62						
"neos-935627"	7859	10301	40476	0.79	0.78	0.72	0.71	0.71	0.69	0.62	0.59
"neos-935769"	6741	9799	36447	0.77	0.77	0.70	0.69	0.68	0.66	0.60	0.56
"neos-937511"	8158	11332	44237	0.81	0.81	0.75	0.73	0.72	0.71	0.65	0.60
"neos-937815"	9251	11646	48013	0.81	0.81	0.76	0.75	0.73	0.72	0.66	0.62
"neos-941262"	6703	9480	35659	0.77	0.76	0.70	0.69	0.67	0.65	0.61	0.56
"neos-942830"	803	882	13290	0.84	0.55	0.39	0.25	0.19			
"neos-948126"	7271	9551	38219	0.79	0.78	0.73	0.70	0.69	0.67	0.62	0.58
"neos-952987"	354	31329	90384	0.44	0.55	0.52					
"neos-984165"	6962	8883	36742	0.81	0.79	0.73	0.71	0.68	0.66	0.62	0.58
"net12"	14021	14115	80384	0.94	0.93	0.90	0.89	0.87	0.86	0.86	0.86
"newdano"	576	505	2184	0.91	0.90	0.89	0.86	0.80	0.74	0.67	
"nobel-eu-DBE"	879	3771	11313	0.91	0.91	0.90	0.90	0.89			
"noswot"	182	128	735	0.89	0.87	0.65	0.58	0.38			
"ns1208400"	4289	2883	81746	0.33	0.26	0.24	0.23	0.13	0.07	0.05	
"ns1606230"	3503	4173	92133	0.80	0.51	0.45	0.40	0.33	0.29	0.25	
"ns1686196"	4055	2738	68529	0.85	0.80	0.76	0.70	0.65	0.62	0.61	0.59
"ns1688347"	4191	2685	66908	0.87	0.80	0.75	0.70	0.65	0.61	0.59	0.56
"ns1702808"	1474	804	5856	0.99	0.95	0.85	0.81	0.79	0.77	0.76	0.75
"ns1745726"	4687	3208	90278	0.85	0.77	0.71	0.67	0.63	0.59	0.58	
"ns1766074"	182	100	666	0.85	0.83	0.80	0.77	0.70			
"ns1778858"	10666	4720	32673	0.99	0.99	0.99	0.99	0.97	0.93	0.90	0.83
"ns1905800"	8289	3228	38100	0.97	0.91	0.91	0.89	0.87	0.87	0.86	0.85

Table 8.22: results for miplib2010 to arrowhead (part 2)

number of blocks				2	4	8	16	32	64	128	256
instance											
"ns4-pr3"	2210	8601	25986	0.95	0.95	0.94	0.92	0.89	0.86	0.68	0.59
"ns4-pr9"	2220	7350	22176	0.96	0.96	0.96	0.93	0.90	0.86	0.72	0.63
"ns894236"	8218	9666	41067	0.98	0.98	0.97	0.96	0.93	0.86	0.79	0.69
"ns894244"	12129	21856	90864	0.98	0.98	0.97	0.96	0.94	0.90	0.80	0.73
"ns894788"	2279	3463	14381	0.96	0.94	0.91	0.85	0.77	0.69	0.63	
"ns903616"	18052	21582	91641	0.99	0.98	0.98	0.97	0.97	0.93	0.89	0.80
"nu120-pr3"	2210	8601	25986	0.95	0.95	0.94	0.92	0.89	0.86	0.68	0.59
"nu60-pr9"	2220	7350	22176	0.96	0.96	0.96	0.93	0.90	0.86	0.72	0.63
"opm2-z7-s2"	31798	2023	79762	0.69	0.54	0.41	0.37	0.34	0.31	0.28	
"p100x588b"	688	1176	2352	1.00	0.99	0.98	0.96	0.94	0.90	0.85	0.83
"p2m2p1m1p0n100"	1	100	100								
"p6b"	5852	462	11704	0.72	0.53	0.36	0.26				
"p80x400b"	480	800	1600	0.99	0.99	0.98	0.95	0.91	0.88	0.83	
"pg5_34"	225	2600	7700	0.90	0.90	0.89	0.86	0.58			
"pg"	125	2700	5200	0.82	0.82	0.81	0.79				
"pigeon-10"	931	490	8150	0.65	0.62	0.60	0.57	0.46			
"pigeon-11"	1123	572	9889	0.67	0.64	0.62	0.57	0.52			
"pigeon-12"	1333	660	11796	0.68	0.66	0.64	0.61	0.59	0.57		
"pigeon-13"	1561	754	13871	0.70	0.68	0.66	0.66	0.56	0.53		
"pigeon-19"	3307	1444	29849	0.75	0.75	0.74	0.73	0.69	0.63	0.55	
"probportfolio"	302	320	6620	0.94	0.94	0.94	0.94	0.92	0.84		
"protfold"	2112	1835	23491	0.79	0.71	0.61	0.52	0.40	0.31		
"pw-myciel4"	8164	1059	17779	0.98	0.95	0.90	0.75	0.56	0.50	0.39	
"qiu"	1192	840	3432	0.95	0.95	0.94	0.94	0.93	0.92	0.82	0.81
"queens-30"	960	900	93440								
"r80x800"	880	1600	3200	0.97	0.96	0.95	0.94	0.93	0.91	0.90	0.87
"ramos3"	2187	2187	32805	0.45	0.22	0.14	0.08				
"ran14x18-disj-8"	447	504	10277	0.79	0.71	0.70	0.61	0.61	0.57		
"ran14x18"	284	504	1008	0.96	0.94	0.90	0.88	0.88	0.86	0.90	
"ran16x16"	288	512	1024	0.93	0.92	0.90	0.89	0.88	0.86	0.84	
"reblock166"	17024	1660	39442	0.98	0.79	0.53	0.40	0.30	0.25		
"reblock354"	19906	3540	52901	0.97	0.79	0.68	0.58	0.50	0.42	0.37	
"reblock67"	2523	670	7495	0.94	0.82	0.65	0.56	0.52	0.42		
"rmatr100-p10"	7260	7359	21877	0.99	0.99	0.99	0.99	0.98	0.98	0.97	0.95
"rmatr100-p5"	8685	8784	26152	0.99	0.99	0.99	0.99	0.99	0.98	0.97	0.96
"rmatr200-p20"	29406	29605	88415	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
"rmine6"	7078	1096	18084	0.91	0.75	0.69	0.59	0.49	0.45		
"rococoB10-011000"	1667	4456	16517	0.93	0.92	0.94	0.91	0.86	0.73	0.67	
"rococoC10-001000"	1293	3117	11751	0.93	0.92	0.93	0.92	0.92	0.75	0.69	
"rococoC11-011100"	2367	6491	30472	0.94	0.93	0.91	0.90	0.86	0.75	0.67	
"rococoC12-111000"	10776	8619	48920	0.98	0.98	0.98	0.97	0.96	0.98	0.92	0.90
"roll3000"	2295	1166	29386	0.82	0.74	0.70	0.64	0.54	0.48		
"satellites1-25"	5996	9013	59023	0.94	0.91	0.88	0.75	0.70	0.66	0.64	0.61
"set3-10"	3747	4019	13747	0.98	0.93	0.91	0.89	0.87	0.81	0.77	0.72
"set3-15"	3747	4019	13747	0.98	0.93	0.91	0.89	0.87	0.81	0.77	0.71
"set3-20"	3747	4019	13747	0.98	0.93	0.91	0.89	0.87	0.81	0.77	0.71
"seymour"	4944	1372	33549	0.84	0.75	0.62	0.55	0.49			
"seymour-disj-10"	5108	1209	64704	0.55	0.48	0.43	0.40	0.35			
"sp98ir"	1531	1680	71704	0.73	0.61	0.60	0.60	0.57	0.56	0.53	
"sts405"	27270	405	81810								
"swath"	884	6805	34965	0.88	0.89	0.88	0.88	0.81	0.79	0.57	0.44
"tanglegram2"	8980	4714	26940	1.00	0.99	0.99	0.99	0.99	0.98	0.98	0.97
"timtab1"	171	397	829	0.96	0.94	0.92	0.88	0.84	0.80		
"toll-like"	4408	2883	13224	0.98	0.98	0.97	0.96	0.95	0.93	0.91	0.87
"transportmoment"	9616	9685	29541	1.00	1.00	0.99	0.99	0.98	0.97	0.95	0.91
"tw-myciel4"	8146	760	27961	0.47	0.20	0.15	0.12				
"uct-subprob"	1973	2256	10147	0.96	0.93	0.90	0.88	0.85	0.83	0.80	0.78
"umts"	4465	2947	23016	0.98	0.97	0.97	0.96	0.95	0.92	0.90	0.88
"usAbbrv-8-25_70"	3291	2312	9628	0.98	0.94	0.91	0.88	0.84	0.80	0.76	0.72
"wachplan"	1553	3361	89361	0.50	0.46	0.41	0.39	0.34			
"zib54-UUE"	1809	5150	15288	0.96	0.96	0.96	0.96	0.96	0.88	0.81	0.74
quadratic mean				0.86	0.82	0.78	0.75	0.70	0.65	0.59	0.51

Table 8.23: results for miplib2010 to arrowhead (part 3)

number of blocks				2	4	8	16	32	64	128	256
instance	#rows	#cols	#nonz								
"50v-10"	233	2013	2745	0.93	0.90	0.89	0.86	0.83	0.64		
"a1c1s1"	3312	3648	10178	0.98	0.93	0.86	0.81	0.77	0.71	0.65	
"acc-tight4"	3285	1620	17073	0.79	0.71	0.67	0.60	0.54	0.48	0.42	
"acc-tight5"	3052	1339	16134	0.79	0.71	0.66	0.55	0.48	0.42		
"acc-tight6"	3047	1335	16108	0.79	0.71	0.65	0.55	0.48	0.43	0.36	
"aflow40b"	1442	2728	6783	0.97	0.97	0.97	0.96	0.94	0.94	0.93	0.93
"air04"	823	8904	72965	0.31	0.27	0.24	0.17				
"ash608gpia-3col"	24748	3651	74244								
"atm20-100"	4380	6480	58878	0.98	0.94	0.88	0.84	0.79	0.73	0.67	
"b2c1s1"	3904	3872	11408	0.99	0.98	0.98	0.97	0.96	0.92	0.88	0.84
"beasleyC3"	1750	2500	5000	0.97	0.80	0.71	0.60				
"berlin_5_8_0"	1532	1083	4507	0.97	0.94	0.89	0.82	0.77			
"bg512142"	1307	792	3953	0.84	0.60	0.45	0.36				
"biella1"	1203	7328	71489	0.87	0.82	0.78	0.77	0.76			
"bienst2"	576	505	2184	0.99	0.98	0.96	0.93	0.90	0.86	0.81	0.72
"binkar10_1"	1026	2298	4496	0.84	0.70	0.64	0.61	0.56	0.52		
"bnatt350"	4923	3150	19061	0.84	0.72	0.66	0.62	0.58	0.54		
"bnatt400"	5614	3600	21698								
"cov1075"	637	120	14280	0.80	0.56	0.42	0.36	0.28			
"csched007"	351	1758	6379	0.78	0.54	0.40	0.34				
"csched008"	351	1536	5687	0.85	0.61	0.50	0.40				
"csched010"	351	1758	6376	0.31	0.30	0.28	0.28	0.28	0.18		
"dano3mip"	3202	13873	79655	0.67	0.60	0.55	0.47				
"danoint"	664	521	3232	0.81	0.76	0.72	0.67	0.69			
"dfn-gwin-UUM"	158	938	2632	0.73	0.65	0.62					
"dg012142"	6310	2080	14795	0.96	0.94	0.90	0.85	0.81	0.79	0.77	
"eil33-2"	32	4516	44243								
"eilB101"	100	2818	24120								
"enlight13"	169	338	962	0.86	0.74	0.55					
"enlight14"	196	392	1120	0.87	0.76	0.59					
"enlight15"	225	450	1290	0.88	0.78	0.63					
"enlight16"	256	512	1472	0.89	0.80	0.63					
"enlight9"	81	162	450	0.80	0.63						
"f2000"	10500	4000	29500	0.62	0.45	0.38	0.35	0.34	0.31	0.28	0.26
"g200x740i"	940	1480	2960	0.99	0.97	0.95	0.93	0.89	0.84	0.77	0.75
"germany50-DBM"	2526	8189	24479	0.93	0.93	0.93	0.93	0.90	0.84	0.77	0.66
"glass4"	396	322	1815	0.66	0.41	0.28	0.22				
"gmu-35-40"	424	1205	4843	0.98	0.95	0.86	0.73				
"gmu-35-50"	435	1919	8643	0.96	0.92	0.78	0.70	0.61			
"go19"	441	441	1885	0.92	0.86	0.73	0.62	0.50			
"hanoi5"	16399	3862	39718	0.99	0.97	0.93	0.85	0.76	0.65	0.55	
"harp2"	112	2993	5840	0.68	0.68	0.68	0.61	0.35			
"ic97_potential"	1046	728	3138	0.98	0.93	0.87	0.81				
"iis-100-0-cov"	3831	100	22986								
"iis-bupa-cov"	4803	345	38392								
"iis-pima-cov"	7201	768	71941								
"janos-us-DDM"	760	2184	6384	0.90	0.89	0.88	0.85	0.83	0.72		
"k16x240"	256	480	960	0.95	0.94	0.94	0.93	0.93	0.90	0.88	
"lectsched-4-obj"	14163	7901	82428	0.72	0.56	0.43	0.36	0.31			
"liu"	2178	1156	10626	0.58							
"lotsize"	1920	2985	6565	0.99	0.99	0.98	0.96	0.91	0.85	0.82	0.77
"lrsa120"	14521	3839	39956	0.76	0.63	0.55	0.52	0.50			
"m100n500k4r1"	100	500	2000	0.21	0.17	0.16					
"macrophage"	3164	2260	9492	0.99	0.96	0.94	0.88	0.79			
"markshare_5_0"	5	45	203								
"maxgasflow"	7160	7437	19717	1.00	1.00	0.99	0.99	0.98	0.98	0.95	0.89
"mc11"	1920	3040	6080	0.99	0.98	0.97	0.95	0.93	0.89	0.82	0.78
"mcsched"	2107	1747	8088	0.96	0.94	0.85	0.79	0.75	0.69	0.63	
"methanosarcina"	14604	7930	43812	0.82	0.57	0.40	0.28				
"mik-250-1-100-1"	151	251	5351	0.40	0.38	0.36	0.36				
"mine-166-5"	8429	830	19412	0.95	0.78	0.69	0.55				
"mine-90-10"	6270	900	15407	0.99	0.93	0.77	0.66	0.53			
"mkc"	3411	5325	17038	0.99	0.99	0.99	0.98	0.97	0.95	0.91	0.89
"msc98-ip"	15850	21143	92918	0.92	0.86	0.79	0.75	0.70	0.66	0.59	
"n3-3"	2425	9028	35380	0.91	0.91	0.89	0.88	0.86	0.84	0.70	0.60
"n3700"	5150	10000	20000	0.99	0.99	0.98	0.98	0.97	0.97	0.97	0.95
"n3705"	5150	10000	20000	0.99	0.99	0.98	0.98	0.97	0.97	0.97	0.96
"n370a"	5150	10000	20000	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.96

Table 8.24: results for miplib2010 to bordered block diagonal (part1)

number of blocks				2	4	8	16	32	64	128	256
instance											
"n4-3"	1236	3596	14036	0.92	0.91	0.91	0.90	0.89	0.88	0.74	0.67
"n9-3"	2364	7644	30072	0.93	0.93	0.93	0.92	0.91	0.86	0.78	0.65
"nag"	5840	2884	26499	0.72	0.55	0.44	0.39	0.33	0.32	0.29	0.27
"neos-1109824"	28979	1520	89528	1.00	1.00	0.57	0.37	0.24	0.16	0.11	0.09
"neos-1112782"	2115	4140	8145	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.96
"neos-1112787"	1680	3280	6440	0.98	0.98	0.98	0.98	0.97	0.97	0.96	0.96
"neos-1171692"	4239	1638	42945	0.98	0.97	0.97	0.87	0.75			
"neos-1171737"	4179	2340	58620	0.99	0.98	0.96	0.94	0.94	0.43		
"neos-1224597"	3276	3395	25090	0.96	0.94	0.87	0.87	0.84			
"neos-1225589"	675	1300	2525	0.97	0.97	0.97	0.96	0.95	0.95	0.94	0.94
"neos-1311124"	1643	1092	7140	0.99	0.98	0.98	0.96	0.96	0.73		
"neos-1337307"	5687	2840	30799	0.97	0.95	0.94	0.94	0.92	0.87	0.80	
"neos-1396125"	1494	1161	5511	0.85	0.80	0.77	0.78	0.73	0.58		
"neos-1426635"	796	520	3400	0.98	0.98	0.93	0.94	0.70			
"neos-1426662"	1914	832	8048	0.99	0.98	0.98	0.95	0.94			
"neos-1436709"	1417	676	6214	0.99	0.98	0.95	0.92	0.84	0.61		
"neos-1440225"	330	1285	14168	0.55	0.51	0.38					
"neos-1440460"	989	468	4302	0.98	0.97	0.94	0.93	0.69			
"neos-1442119"	1524	728	6692	0.99	0.97	0.98	0.91	0.86			
"neos-1442657"	1310	624	5736	0.98	0.97	0.94	0.95	0.78			
"neos15"	552	792	1766	0.98	0.95	0.87	0.81	0.75	0.67	0.54	
"neos-1601936"	3131	4446	72500	0.74	0.67	0.64	0.52	0.37			
"neos-1605061"	3474	4111	93483	0.68	0.47	0.43	0.40	0.28			
"neos-1605075"	3467	4173	91377	0.68	0.52	0.47	0.40	0.32			
"neos-1616732"	1999	200	3998	0.67	0.49						
"neos-1620770"	9296	792	19292	1.00	0.98	0.98	0.88	0.75			
"neos16"	1018	377	2801	0.84	0.76	0.69	0.59	0.50	0.43		
"neos18"	11402	3312	24614	0.93	0.86	0.81	0.79	0.76	0.70	0.61	0.52
"neos-506422"	6811	2527	31815	0.71	0.55	0.45	0.39	0.35			
"neos-555424"	2676	3815	15667	0.99	0.98	0.94	0.91	0.91	0.89	0.80	0.68
"neos-686190"	3664	3660	18085	0.61	0.40						
"neos-777800"	479	6400	32000	0.51	0.42	0.41	0.37	0.34			
"neos-785912"	1714	1380	16610	0.90	0.88	0.87	0.84	0.80	0.77		
"neos788725"	433	352	4912	0.83	0.74	0.71					
"neos-807456"	840	1635	4905	0.64	0.54	0.49	0.45	0.41	0.36		
"neos-820146"	830	600	3225	0.89	0.85						
"neos-820157"	1015	1200	4875	0.89	0.86						
"neos-824695"	9576	23970	72590	1.00	0.99	0.99	0.99	0.97	0.95	0.91	0.77
"neos-826650"	2414	5912	20440	0.93	0.93	0.93	0.93	0.93	0.76	0.68	0.65
"neos-826694"	6904	16410	59268	0.98	0.97	0.97	0.96	0.95	0.91	0.83	0.71
"neos-826812"	6844	15864	53808	0.98	0.98	0.97	0.96	0.96	0.91	0.83	0.71
"neos-826841"	2354	5516	18460	0.95	0.95	0.95	0.95	0.95	0.78	0.69	0.66
"neos-847302"	609	737	9566	0.64	0.57	0.52	0.52				
"neos-849702"	1041	1737	19308	0.70	0.65	0.61	0.58	0.56			
"neos858960"	132	160	2770	0.69	0.66	0.64	0.63	0.61	0.59		
"neos-911880"	83	888	2568	0.62	0.62	0.34					
"neos-935627"	7859	10301	40476	0.96	0.95	0.88	0.86	0.85	0.83	0.77	0.71
"neos-935769"	6741	9799	36447	0.95	0.94	0.86	0.85	0.84	0.82	0.75	
"neos-937511"	8158	11332	44237	0.95	0.94	0.87	0.87	0.85	0.84	0.77	0.72
"neos-937815"	9251	11646	48013	0.96	0.95	0.88	0.88	0.86	0.84	0.77	0.75
"neos-941262"	6703	9480	35659	0.95	0.93	0.86	0.85	0.83	0.80	0.74	
"neos-942830"	803	882	13290	0.81	0.65	0.38	0.21				
"neos-948126"	7271	9551	38219	0.96	0.94	0.87	0.84	0.83	0.80	0.75	0.70
"neos-952987"	354	31329	90384								
"neos-984165"	6962	8883	36742	0.96	0.94	0.87	0.85	0.83	0.80	0.75	0.71
"net12"	14021	14115	80384	0.96	0.96	0.96	0.87	0.74			
"newdano"	576	505	2184	0.86	0.81	0.78	0.77	0.77			
"nobel-eu-DBE"	879	3771	11313	0.91	0.91	0.89	0.90	0.86			
"noswot"	182	128	735	0.94	0.82	0.70					
"ns1208400"	4289	2883	81746	0.77	0.69	0.64	0.60	0.58			
"ns1606230"	3503	4173	92133	0.68	0.51	0.47	0.42	0.33	0.27		
"ns1686196"	4055	2738	68529	0.89	0.72	0.62	0.51				
"ns1688347"	4191	2685	66908	0.92	0.84	0.76	0.68	0.59	0.48		
"ns1702808"	1474	804	5856	0.99	0.92	0.83					
"ns1745726"	4687	3208	90278	0.89	0.72	0.59	0.49				
"ns1766074"	182	100	666	0.59	0.35						
"ns1778858"	10666	4720	32673	0.99	0.99	0.99	0.99	0.97	0.90	0.79	
"ns1905800"	8289	3228	38100	0.94							

Table 8.25: results for miplib2010 to bordered block diagonal (part2)

number of blocks				2	4	8	16	32	64	128	256
instance											
"ns2081729"	1190	661	5680	0.84	0.68	0.39					
"ns2122603"	24754	19300	77044	1.00	0.99	0.97	0.94	0.90	0.88	0.86	0.83
"ns4-pr3"	2210	8601	25986	0.95	0.94	0.94	0.92	0.89	0.85	0.71	0.59
"ns4-pr9"	2220	7350	22176	0.96	0.96	0.96	0.93	0.90	0.85	0.75	0.64
"ns894236"	8218	9666	41067	0.98	0.98	0.96	0.94	0.88	0.80	0.70	0.62
"ns894244"	12129	21856	90864	0.98	0.97	0.96	0.95	0.91	0.83	0.71	0.62
"ns894788"	2279	3463	14381	0.95	0.93	0.86	0.78	0.70	0.64		
"ns903616"	18052	21582	91641	0.99	0.98	0.97	0.95	0.94	0.89	0.80	0.69
"nu120-pr3"	2210	8601	25986	0.95	0.94	0.94	0.92	0.89	0.85	0.71	0.60
"nu60-pr9"	2220	7350	22176	0.96	0.96	0.96	0.93	0.90	0.85	0.75	0.64
"opm2-z7-s2"	31798	2023	79762	0.81	0.69	0.55	0.45				
"p100x588b"	688	1176	2352	1.00	0.99	0.97	0.96	0.90	0.89	0.83	0.83
"p2m2p1m1p0n100"	1	100	100								
"p6b"	5852	462	11704	0.84	0.69	0.56					
"p80x400b"	480	800	1600	0.99	0.98	0.98	0.94	0.90	0.87	0.82	
"pg5_34"	225	2600	7700	0.90	0.90	0.89	0.86	0.46			
"pg"	125	2700	5200	0.82	0.82	0.81	0.79	0.79			
"pigeon-10"	931	490	8150	0.61							
"pigeon-11"	1123	572	9889	0.61							
"pigeon-12"	1333	660	11796	0.62	0.40						
"pigeon-13"	1561	754	13871	0.63							
"pigeon-19"	3307	1444	29849	0.63							
"probportfolio"	302	320	6620								
"protfold"	2112	1835	23491	0.80	0.76	0.65	0.49	0.36			
"pw-myciel4"	8164	1059	17779	0.99	0.98	0.97	0.87	0.79			
"qiu"	1192	840	3432	0.89	0.84	0.80	0.71	0.64	0.60		
"queens-30"	960	900	93440								
"r80x800"	880	1600	3200	0.96	0.96	0.95	0.94	0.93	0.90	0.89	0.87
"ramos3"	2187	2187	32805	0.45	0.23	0.16	0.12				
"ran14x18-disj-8"	447	504	10277	0.79	0.75	0.72	0.62	0.61	0.60	0.56	
"ran14x18"	284	504	1008	0.96	0.95	0.89	0.88	0.88	0.86	0.84	
"ran16x16"	288	512	1024	0.95	0.93	0.92	0.89	0.86	0.86	0.84	
"reblock166"	17024	1660	39442	0.99	0.94	0.77	0.67	0.55			
"reblock354"	19906	3540	52901	0.96	0.93	0.83	0.75	0.65			
"reblock67"	2523	670	7495	0.94	0.89	0.75	0.65	0.55			
"rmatr100-p10"	7260	7359	21877	0.72	0.61	0.55	0.51	0.48	0.44		
"rmatr100-p5"	8685	8784	26152	0.72	0.59	0.54	0.50	0.47	0.44		
"rmatr200-p20"	29406	29605	88415	0.71	0.59	0.53	0.50	0.49	0.48	0.42	
"rmine6"	7078	1096	18084	0.95	0.88	0.78	0.71	0.60			
"rococoB10-011000"	1667	4456	16517	0.94	0.92	0.92	0.92	0.87	0.76		
"rococoC10-001000"	1293	3117	11751	0.92	0.93	0.92	0.89	0.87	0.78		
"rococoC11-011100"	2367	6491	30472	0.95	0.93	0.92	0.91	0.87	0.81	0.70	
"rococoC12-111000"	10776	8619	48920	0.95	0.94	0.90	0.89	0.88	0.88	0.55	0.37
"roll3000"	2295	1166	29386	0.93	0.93	0.90	0.82	0.61	0.53		
"satellites1-25"	5996	9013	59023	0.82	0.73	0.65	0.63	0.57	0.52	0.48	
"set3-10"	3747	4019	13747	0.98	0.94	0.89	0.86	0.81	0.71	0.63	
"set3-15"	3747	4019	13747	0.98	0.94	0.89	0.85	0.81	0.71	0.63	
"set3-20"	3747	4019	13747	0.98	0.94	0.89	0.86	0.80	0.71	0.63	
"seymour"	4944	1372	33549	0.90	0.71	0.60	0.42	0.32	0.21		
"seymour-disj-10"	5108	1209	64704	0.89	0.68	0.51	0.37				
"sp98ir"	1531	1680	71704	0.87	0.85	0.85	0.84	0.83	0.82	0.80	0.78
"sts405"	27270	405	81810	0.32	0.29						
"swath"	884	6805	34965	0.74	0.62	0.54	0.50				
"tanglegram2"	8980	4714	26940	0.77	0.64	0.47	0.36				
"timtab1"	171	397	829	0.82	0.65	0.55					
"toll-like"	4408	2883	13224	0.96	0.93	0.86	0.77	0.65			
"transportmoment"	9616	9685	29541	1.00	1.00	1.00	0.99	0.98	0.97	0.94	0.86
"tw-myciel4"	8146	760	27961	0.85	0.54	0.32	0.21				
"uct-subprob"	1973	2256	10147	0.84	0.75	0.65	0.53	0.43			
"umts"	4465	2947	23016	0.95	0.94	0.92	0.88	0.71	0.53	0.42	
"usAbbrv-8-25_70"	3291	2312	9628	0.96	0.85	0.76	0.66	0.58			
"wachplan"	1553	3361	89361	0.67	0.50	0.38					
"zib54-UUE"	1809	5150	15288	0.96	0.96	0.96	0.96	0.96	0.89	0.79	0.74
				0.85	0.79	0.73	0.68	0.63	0.52	0.45	0.36

Table 8.26: results for miplib2010 to bordered block diagonal (part3)

		IP_A			IP_{AR}			IP_{AC}		
Instance	lc.	gap	nNodes	time	gap	nNodes	time	gap	nNodes	time
bell3a	LO	0%	2	2.52	0%	1	0.44	0%	1	0.49
	ME	0%	19	8.18	0%	15	8.96	0%	9	4.27
	TI	0%	73	18.88	0%	46	21.76	0%	58	19.85
bell5	LO	0%	2	1.83	0%	1	0.42	0%	1	1.59
	ME	0%	3	2.41	0%	12	4.55	0%	21	5.49
	TI	0%	6	2.72	0%	21	6.71	0%	23	7.27
bm23	LO	0%	462	15.64	0%	345	17.19	0%	109	15.18
	ME	infeas.	1	0.06	infeas.	1	0.08	infeas.	1	0.07
	TI	infeas.	1	0.07	infeas.	1	0.04	infeas.	1	0.06
egout	LO	0%	13	4.66	0%	7	2.7	0%	12	5.57
	ME	0%	14	4.8	0%	9	4.45	0%	11	5.32
	TI	0%	10	3.93	0%	13	5.13	0%	12	8.24
enigma	LO	0%	2913	20.86	0%	54	8.26	0%	804	13.14
	ME	0%	37	6.05	0%	27	3.96	0%	26	5.65
	TI	0%	17	5.89	0%	21	6.73	0%	25	9.89
fixnet3	LO	0%	1	42.41	0%	1	27.67	0%	1	48.87
	ME	0%	1939	649.17	0%	1638	885.26	0%	5816	1733.59
	TI	0%	2564	1151.11	0%	1119	498.19	0%	2278	734.75
flugpl	LO	0%	1	0.04	0%	5	0.1	0%	1	0.03
	ME	0%	1	0.03	0%	4	0.14	0%	1	0.04
	TI	0%	1	0.04	0%	4	0.09	0%	1	0.05
gt2	LO	0%	144	13.68	0%	72	13.24	0%	60	13.52
	ME	0%	283	15.9	0%	132	15.26	0%	228	17.3
	TI	0%	361	39.26	0%	716	45.97	0%	374	33.43
khb05250	LO	0%	1	5.45	0%	1	5.65	0%	1	47.47
	ME	394%	2204	1800.68	374%	2717	1800.02	372%	2633	1800.02
	TI	1e+20	350	1800.06	1e+20	564	1800.05	1e+20	592	1800.08
lseu	LO	0%	3	1.23	0%	1	0.87	0%	5	1.63
	ME	0%	24	6.07	0%	18	4.56	0%	22	5.64
	TI	0%	3	3.64	0%	9	5.93	0%	3	5.3
markshare1	LO	0%	1	0.09	0%	1	0.1	0%	1	0.12
	ME	infeas.	1	0.04	infeas.	1	0.04	infeas.	1	0.04
	TI	infeas.	1	0.05	infeas.	1	0.04	infeas.	1	0.04
markshare2	LO	0%	1	0.11	0%	1	0.16	0%	1	0.14
	ME	infeas.	1	0.06	infeas.	1	0.06	infeas.	1	0.05
	TI	infeas.	1	0.04	infeas.	1	0.06	infeas.	1	0.06
misc01	LO	0%	271	22.89	0%	13	11.16	0%	19	13.97
	ME	0%	24769	568.64	0%	18387	426.66	0%	15040	411.63
	TI	0%	5	14.5	0%	5	13.15	0%	5	14.87
mod008	LO	0%	1576	44.63	0%	708	45.38	0%	144	53.3
	ME	0%	276	62.78	0%	85	41.51	0%	1281	99.93
	TI	infeas.	1	0.14	infeas.	1	0.14	infeas.	1	0.17
neos858960	LO	0%	2	17.69	0%	1	2.7	0%	1	2.75
	ME	0%	6488	1262.6	0%	4426	1003.13	0%	4321	864.61
	TI	0%	1187	1328.04	0%	853	1156.81	0%	965	1227.43
noswot	LO	0%	852	44.83	0%	148	30.14	0%	249	35.55
	ME	0%	370	36.71	0%	362	42.91	0%	212	40.71
	TI	48%	57281	1800	53%	48395	1800	45%	60318	1800
p0033	LO	0%	1	0.06	0%	1	0.1	0%	1	0.07
	ME	0%	3	0.24	0%	1	0.16	0%	1	0.19
	TI	0%	2	0.21	0%	2	0.18	0%	2	0.24
p0040	LO	0%	13	0.87	0%	22	0.84	0%	1	0.32
	ME	0%	3	0.34	0%	5	0.35	0%	5	0.34
	TI	0%	5	0.34	0%	5	0.34	0%	1	0.28
pp08a	LO	0%	2	5	0%	2	3.11	0%	1	0.73
	ME	0%	2894	90.9	0%	978	31.67	0%	595	38.49
	TI	0%	2608	89.5	0%	357	28.42	0%	166	22.76
pipex	LO	0%	47	4.69	0%	41	2.97	0%	39	3.17
	ME	0%	29	4.67	0%	23	3.5	0%	14	2.54
	TI	0%	3	0.85	0%	3	0.99	0%	3	1
rgn	LO	0%	395	21.74	0%	246	15.37	0%	246	19.23
	ME	0%	15	7.06	0%	9	4.64	0%	5	4.6
	TI	0%	363	20.2	0%	485	20.17	0%	413	28.82
pk1	LO	0%	12	7.25	0%	9	8.99	0%	1	1.2
	ME	0%	7	18.6	0%	12	18.42	0%	39	28.07
	TI	0%	1	0.18	0%	1	0.17	0%	1	0.19
sample2	LO	0%	1	0.11	0%	1	0.52	0%	7	1.62
	ME	0%	38	3.31	0%	36	3.02	0%	30	3.38
	TI	0%	318	9.34	0%	629	9.08	0%	374	8.93
stein9	LO	0%	28	0.28	0%	30	0.29	0%	20	0.2
	ME	0%	536	0.66	0%	416	0.69	0%	475	0.6
	TI	0%	3	0.08	0%	1	0.09	0%	3	0.07
stein15	LO	0%	1245	3.71	0%	186	3.05	0%	26	1.06
	ME	0%	14787	20.61	0%	7382	13.3	0%	11873	16.97
	TI	0%	7	0.78	0%	5	0.71	0%	5	0.57
stein27	LO	0%	261913	1300.67	0%	3708	34.71	0%	1103	16.57
	ME	85%	368826	1800	82%	283033	1800	74%	369689	1800
	TI	0%	15	11.48	0%	13	9.47	0%	7	4.44
stein45	LO	289%	17238	1800.03	0%	14401	664.49	0%	9581	518.12
	ME	488%	45474	1800	495%	28656	1800	403%	46567	1800.02
	TI	0%	409	192.33	0%	817	455.33	0%	372	168.12
timtab1	LO	0%	2	7.85	0%	8	20.34	0%	1	8.34
	ME	0%	883	82.51	0%	706	61.01	0%	176	35.77
	TI	0%	1343	112.63	0%	1020	84.02	0%	1454	132.37
vpm1	LO	0%	17	30.53	0%	2	11.4	0%	13	20.96
	ME	0%	1430	101.76	0%	460	64.24	0%	3733	258.5
	TI	0%	190	45.36	0%	232	47.63	0%	126	46.91

Table 8.27: Results for exact solving (2Blocks)

		IP_A			IP_{AR}			IP_{AC}		
Instance	lc.	gap	nNodes	time	gap	nNodes	time	gap	nNodes	time
bell3a	LO	0%	2084	505.67	0%	494	172.07	0%	596	211.79
	ME	0%	173	74.95	0%	265	108.41	0%	489	143.56
	TI	0%	1024	245.32	0%	1148	322.06	0%	1561	393.51
bell5	LO	0%	1140	167.33	0%	441	81.36	0%	474	90.52
	ME	0%	1707	282.77	0%	1294	207.54	0%	1293	163.05
	TI	0%	6418	832.77	0%	9241	1018.86	0%	4129	518.99
bm23	LO	0%	1	29.93	0%	3	44.69	0%	5	36.62
	ME	0%	1	0.37	0%	1	3.77	0%	1	0.41
	TI	0%	1	0.21	0%	1	0.27	0%	1	0.21
egout	LO	0%	201	36.28	0%	205	46.31	0%	221	64.39
	ME	0%	147	60.34	0%	266	79.04	0%	82	56.24
	TI	0%	652	130.37	0%	515	103.78	0%	482	108.72
enigma	LO	0%	14021	563.49	0%	4243	240.05	0%	3933	346.3
	ME	0%	1522	104.23	0%	650	69.38	0%	432	68.83
	TI	0%	1212	277.08	0%	1507	292.19	0%	1304	346.74
fixnet3	LO	20300%	181	1800.03	2072%	260	1802.3	1333%	84	1800.12
	ME	1e+20	11	1803.27	1900%	249	1800.09	1e+20	6	1800.17
	TI	3904%	34	1800.02	2244%	153	1800.06	1967%	122	1800.1
flugpl	LO	0%	630	6.96	0%	31	4.31	0%	44	3.68
	ME	0%	53	3.49	0%	23	2.87	0%	21	1.76
	TI	0%	509	9.37	0%	649	10.61	0%	585	9.55
gt2	LO	139%	11506	1800	0%	9315	1339.48	150%	3955	1800.04
	ME	95%	8295	1800	27%	12493	1800.16	31%	5824	1800
	TI	1e+20	2910	1800.04	1e+20	2567	1800.05	1e+20	2509	1800.02
khb05250	LO	9900%	93	1800	0%	93	1416.34	2275%	36	1800.07
	ME	1e+20	6	1800.2	1e+20	6	1800.11	1e+20	6	1800.22
	TI	1e+20	5	1803.91	1e+20	6	1800.08	1e+20	6	1800.13
lseu	LO	0%	912	66.62	0%	275	30.19	0%	291	49.55
	ME	0%	22	43.15	0%	20	40.31	0%	169	75.38
	TI	0%	83	68.25	0%	23	56.28	0%	82	76.71
markshare1	LO	0%	1	0.25	0%	1	0.31	0%	1	4.4
	ME	0%	7	21.97	0%	1	4.82	0%	3	23.77
	TI	infeas.	1	0.09	infeas.	1	0.07	infeas.	1	0.09
markshare2	LO	0%	1	0.4	0%	1	0.47	0%	1	9.95
	ME	0%	1	32	0%	1	10.34	0%	3	55.62
	TI	infeas.	1	0.13	infeas.	1	0.1	infeas.	1	0.13
misc01	LO	408%	1751	1800	207%	1662	1800.29	228%	1374	1800.06
	ME	552%	3012	1800.02	743%	2415	1800	570%	2164	1800.03
	TI	0%	83	268.31	0%	105	376.24	0%	145	495.98
mod008	LO	0%	766	323.19	0%	2115	298.46	0%	3096	824.97
	ME	0%	93	295.39	0%	167	173.13	0%	411	338.24
	TI	0%	82	270.78	0%	257	329.86	0%	1	2.63
neos858960	LO	220%	292	1800.05	60%	447	1800.01	0%	253	1237.29
	ME	1833%	86	1800.01	1366%	749	1800.09	988%	430	1800.08
	TI	1e+20	43	1800.1	1e+20	36	1802.62	1e+20	41	1800.08
noswot	LO	849%	921	1800.01	495%	552	1800	242%	743	1800.04
	ME	86%	2915	1800.24	0%	1543	1220.61	90%	2495	1800.68
	TI	335%	1206	1800.5	212%	1892	1800.04	235%	2525	1800.03
p0033	LO	0%	11	2.9	0%	10	2.53	0%	27	4.42
	ME	0%	88	10.43	0%	28	8.15	0%	22	6.04
	TI	0%	30	10.73	0%	177	12.75	0%	31	12.28
p0040	LO	0%	1152	12.41	0%	10	5.64	0%	3	4.76
	ME	0%	7	3.68	0%	1	2.58	0%	1	1.24
	TI	0%	5883	92.18	0%	16982	289.32	0%	17878	152.44
pp08a	LO	0%	3526	839.68	0%	463	193.31	0%	710	442.78
	ME	33%	3311	1800.53	0%	2648	1195.75	0%	2140	1362.17
	TI	0%	3583	1793.87	0%	3211	1751.14	52%	2616	1800.09
pipex	LO	0%	2319	64.34	0%	1150	30.2	0%	1087	43.33
	ME	0%	554	31.58	0%	27	23.49	0%	15	22.3
	TI	0%	63	34.31	0%	162	38.42	0%	73	36.23
rgn	LO	87%	19939	1800.01	0%	17278	1711.57	77%	8829	1800
	ME	0%	6593	1409.36	0%	5505	1046.92	0%	5245	1686.3
	TI	0%	1793	1269.28	1e+20	2993	1800.02	0%	1936	1610.41
pk1	LO	0%	125	150.25	0%	102	66.58	0%	5	51.95
	ME	0%	2145	358.96	0%	3362	601.15	0%	3005	802.52
	TI	0%	468	471.06	0%	470	525.12	0%	441	435.99
sample2	LO	0%	252	21.62	0%	129	18.2	0%	89	21.33
	ME	0%	4489	185.12	0%	2360	105.36	0%	1953	86.57
	TI	1e+20	29209	1800	1e+20	26881	1800.06	1e+20	27761	1800
stein9	LO	0%	6885	12.58	0%	2173	7.4	0%	2966	7.36
	ME	0%	5991	12.13	0%	2655	8.45	0%	2849	7.55
	TI	0%	1	0.7	0%	1	0.65	0%	5	0.77
stein15	LO	0%	163730	833.71	0%	52086	327.15	0%	41241	218.03
	ME	0%	43811	275.5	0%	31045	225.35	0%	26953	131.61
	TI	0%	3	4.82	0%	3	7.39	0%	3	5.11
stein27	LO	349%	8864	1800	271%	8160	1800	228%	9044	1800.02
	ME	265%	22806	1800.01	261%	17563	1800.02	190%	31085	1800
	TI	0%	236	251.88	0%	193	226.87	0%	258	235.98
stein45	LO	1908%	358	1800.01	2250%	203	1800.04	1289%	493	1800.46
	ME	2804%	1023	1800.07	2861%	669	1800	2002%	443	1800.03
	TI	1e+20	152	1800.05	1e+20	133	1800.05	1e+20	138	1800.1
timtab1	LO	0%	71	115.02	0%	100	211.99	0%	11	136.95
	ME	292%	1312	1800.04	253%	1422	1800.05	213%	1579	1800.06
	TI	297%	1614	1800.9	214%	2103	1800.02	311%	1869	1800.01
vpm1	LO	710%	2259	1800.52	0%	1156	1174.87	0%	845	1482.29
	ME	434%	1055	1800.06	334%	752	1800.02	279%	926	1800
	TI	634%	1215	1800	428%	1612	1800.24	646%	1157	1800.56

Table 8.28: Results for exact solving (4 blocks)

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] IBM ILOG CPLEX 12.10. reference manual. 2009.

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

[3] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):361–372, 2006. doi: 10.1016/j.orl.2005.07.009.

[4] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: Exponential speed-up for planar graph problems. In *in Electronic Colloquium on Computational Complexity (ECCC*, pages 261–272. Springer, 2001.

[5] N. Alon, P. Seymour, and R. Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990.

[6] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 222–231, New York, NY, USA, 2004. ACM. ISBN 1-58113-852-0.

[7] C. Aykanat, A. Pinar, and Ümit V. Catalyurek. Permuting sparse rectangular matrices into block-diagonal form. *SIAM Journal on Scientific Computing*, 25:1860–1879, 2002.

[8] M. Bergner, A. Caprara, F. Furini, M. E. Lübbecke, E. Malaguti, and E. Traversi. Partial convexification of general mips by dantzig-wolfe reformulation. In O. Günlük and G. J. Woeginger, editors, *IPCO*, volume 6655 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2011. ISBN 978-3-642-20806-5.

[9] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

[10] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1997. ISBN 1886529191.

[11] A. Björck. *Numerical methods for least squares problems*. Society for Industrial Mathematics, 1st edition, 1996.

[12] R. Borndörfer, C. E. Ferreira, and A. Martin. Decomposing matrices into blocks. *SIAM Journal on Optimization*, 9(1):236–269, 1998. ZIB Report 97-15.

[13] T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters 42*, pages 153 – 159, 1992.

[14] V. Chvátal. *Linear programming.* W. H. Freeman, 1983.

[15] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162:2005, 2004.

[16] U. Feige and M. Mahdian. Finding small balanced separators. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 375–384, New York, NY, USA, 2006. ACM. ISBN 1-59593-134-1.

[17] M. C. Ferris and J. D. Horn. Partitioning mathematical programs for parallel solution. *Mathematical Programming 80*, 1998.

[18] G. Gamrath. Generic branch-cut-and-price. Diploma thesis, Technische Universität Berlin, 2010.

[19] M. R. Garey and D. Johnson. *Computers and intractability.* W.H.Freeman and company, 1st edition, 1979.

[20] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984.

[21] D. S. Hochbaum. The pseudoflow algorithm and the pseudoflow-based simplex for the maximum flow problem. In *Proceedings of the 6th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 325–337, London, UK, 1998. Springer-Verlag. ISBN 3-540-64590-X.

[22] E. Ihler, D. Wagner, and F. Wagner. Modeling hypergraphs by graphs with the same mincut properties. *Inf. Process. Lett.*, 45(4):171–175, 1993.

[23] G. Karakostas. A better approximation ratio for the vertex cover problem. *ACM Trans. Algorithms*, 5:41:1–41:8, November 2009. ISSN 1549-6325.

[24] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[25] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *In Proceedings of the Design and Automation Conference*, pages 343–348, 1998.

[26] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, 48(1):96–129, 1998.

[27] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. 20(1):359–392, 1999.

[28] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekar. Multilevel hypergraph partitioning: Application in vlsi design. *Proceedings of DAC*, pages 526–529, 1997.

[29] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 7(1):69–79, 1999. ISSN 1063-8210. doi: 10.1109/92.748202.

[30] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. Miplib 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011. doi: 10.1007/s12532-011-0025-9.

[31] D. P. Koester. Parallel block-diagonal-bordered sparse linear solvers for electrical power system applications, 1995.

[32] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math., 36(2)*, pages 177 – 189, 1979.

[33] J. W. H. Liu. A graph partitioning algorithm by node separators. *ACM Trans. Math. Softw.*, 15:198–219, September 1989. ISSN 0098-3500.

[34] F. Margot. Symmetry in integer linear programming. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer Berlin Heidelberg, 2010. ISBN 978-3-540-68279-0.

[35] B. Mobasher, H. Jain, E. H. Han, and J. Srivastava. Web mining: Pattern discovery from world wide web. Technical Report TR-96-050, Department of Computer Science, University of Minnesota, Minneapolis, 1996.

[36] D. A. Papa and I. L. Markov. Hypergraph partitioning and clustering. In *In Approximation Algorithms and Metaheuristics*, 2007.

[37] K. Schloegel, G. Karypis, V. Kumar, J. Dongarra, I. Foster, G. Fox, K. Kennedy, A. White, and M. Kaufmann. Graph partitioning for high performance scientific simulations, 2000.

[38] G. Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, 2003. ISBN 9780961408893.

[39] L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1 edition, Nov. 1999. ISBN 0471359432.

[40] Ümit V. Çatalyürek and C. Aykanat. Decomposing irregularly sparse matrices for parallel matrix-vector multiplication. volume 1117 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 1996. ISBN 3-540-61549-0.

[41] Ümit V. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.

[42] Ümit V. Çatalyürek, C. Aykanat, and B. Uçar. On two-dimensional sparse matrix partitioning: Models, methods, and a recipe. *SIAM J. Scientific Computing*, 32(2): 656–683, 2010.