Combinatorial Aspects of Low-Rank Matrix Factorization and Two Applications in Bioinformatics

Epameinondas Fritzilas

Februar 2009

Dissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
(Doctor rerum naturalium)

an der Technischen Fakultät
der Universität Bielefeld

Betreuer:
Prof. Dr. rer. nat. Sven Rahmann
Prof. Dr. rer. nat. Jens Stoye

1. Referee: Prof. Dr. Sven Rahmann
2. Referee: Prof. Dr. Jens Stoye

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706

Στην οικογϵνεια μου

To my family

Contents

Chapter 1

Introduction

In many signal processing and data mining applications we need to approximate a given matrix Y with a low-rank product $Y \approx AX$. Both matrices A and X have to be determined and we assume that from the specifics of the application we can derive some constraints for A and X. In general, there are different factorizations that approximate a given Y equally well and, therefore, the problem is inherently ill-defined. On the other hand, we intuitively expect that the constraints that we impose on the factors must somehow offer some control over the space of possible solutions.

In this work, we focus on an especially strong class of constraints. They arise in applications that involve a bipartite network of sources that are emitting some signals over discrete time and sensors that are monitoring these signals. In this context, Y contains sensor measurements over several time points, X contains source signals over time points and A contains the source-sensor mixing coefficients. We assume that we know a-priori the connectiviy of the network, which implies that, in the factorization $Y \approx AX$, A (the matrix of the mixing coefficients) must have zeros at certain positions. For this class of constraints a fundamental question arises: Does the known zero pattern of A contribute anything to the uniqueness of the factorization?

An observation that follows from the linearity of the model naturally leads us to a characterization of uniqueness up to diagonal scaling. It is important to note that this characterization is combinatorial, in the sense that it is based solely on the structure of the source-sensor network and not on the numerical values of a particular (A, X) solution. In fact, it only assumes that the matrices A and X of a solution are numerically generic. This discussion is formalized in Chapter 3 with the definition of *identifiable bipartite graphs*. Thereby, the concept of *structural rank* is the crucial link between linear algebra and graph theory.

Identifiable graphs are defined in terms of bipartite matchings, which are very well-studied objects both in graph theory and in computer science. Below we mention some existing results from both communities that we use in our investigations. We can only start with *Hall's marriage theorem*, which gives a concise theoretical characterization for the existence of perfect matchings in bipartite graphs. From the algorithmic point of view, a *maximum matching* can be efficiently computed due to *Berge's theorem* and the concept of *augmenting paths*. On the other hand, *König's theorem* states that in bipartite graphs a maximum matching and a minimum vertex cover have equal sizes, i.e., bipartite graphs are easy input instances of the (generally hard) minimum vertex cover problem. An elegant connection of bipartite matchings to *linear programming* via *totally unimodular matrices* and integer polyhedra builds a bridge between continuous and combinatorial optimization. In the case of non-identifiable graphs, we draw some conclusions about the identifiability of the model, using *Dulmage-Mendelsohn (DM) decomposition*, a canonical decomposition of bipartite graphs with especially strong properties. Finally, the concepts of *surplus* and, in general, of *submodular set functions* appear again and again at different points of the discussion.

After having established identifiable bipartite graphs as the basic object of our study, we focus on two optimization problems that arise in the context of source-sensor networks. For these problems we coin the names `MINSENSOR` and `MINSOURCE`; we define and study them in Chapters 4 and 5, respectively. Roughly speaking, both problems deal with the selection of good subgraphs: Given a bipartite graph G the goal is to find a subgraph of G that is identifiable and also satisfies some additional restrictions. Both problems turn out to be NP-hard, as we show with reductions from `SET COVER`. This is a prototypical NP-hard problem with many generalizations and variants, for many of which the approximation (and inapproximability) properties have been well-studied. One powerful generalization is `SUBMODULAR SET COVER`, for which a greedy approach achieves a logarithmic approximation guarantee. We derive an approximation algorithm for `MINSENSOR` rather painlessly, by showing that it is, in fact, a special case of `SUBMODULAR SET COVER`.

In Chapter 6 we ask another natural question that arises from our need to model uncertainty in the network structure. Given an identifiable graph G, where the edges have been predicted with some uncertainty, how many edge modifications (deletions and additions) does it take, so that G loses the property of identifiability? This robustness question is reduced to the computation of surplus in bipartite graphs and we show how this can be done in polynomial time.

In Chapters 7 and 8 we present two applications from bioinformatics that can be abstracted in the context of a source-sensor network. The first one is dealing with the processing of microarray data under the presence of non-specific probes and the second one is dealing with the quantification of transcription factor activities in simple regulatory networks.

In summary, our discussion will cross two bridges that are built on solid ground. Bipartite matchings and related concepts, like surplus and DM decomposition, will bring us to the side of polynomial solvability and `SET COVER` will bring us to the side of NP-hardness. The connection to these prototypical problems of computer science offers a large palette of existing theoretical results. I certainly gained much from my attempts to digest this classical material and combine it in new ways for my own investigations.

Contributions. To the best of our knowledge we are the first to define identifiable bipartite graphs in order to study the uniqueness of solutions in low-rank matrix factorization. We are also the first to investigate the properties of these graphs and related combinatorial optimization problems that arise in the context of source-sensor networks. For our investigations we use several existing theoretical results, which we combine in new ways.

Publications. Some parts of Chapter 3 (basic problem of non-identifiability, definition of identifiable graphs) and Chapter 4 (definition of `MINSENSOR`, NP-hardness and MILP solution) were presented in COCOON'08 [FRSR08]. An improved version of the conference paper, extended with the approximation algorithm for `MINSENSOR`, has been accepted under revisions in Algorithmica. For the near future, we are planning a conference sumbission with (parts of) the remaining material (`MINSOURCE`, robustness, partially identifiable graphs).

Acknowledgments. This work was carried out in the Faculty of Technology of Bielefeld University under a three-year long scholarship from the NRW International Graduate School in Bioinformatics and Genome Research. I am grateful for this scholarship that made it possible to concentrate on my Ph.D. project, without any distractions and additional responsibilities. I thank all my present and former colleagues from the Genome Informatics research group for the relaxed and, at the same time, productive working atmosphere. In our weekly group seminars I gained much, not only in domain-specific knowledge, but also in scientific methodology and presentation skills. I thank my supervisors Sven Rahmann and Jens Stoye for their trust and encouragement at some critical points of the project and the financial support they offered towards the end. Finally, special acknowledgments go to Yasmin Rios-Solis and Martin Milanič for their friendship and scientific advice that decisively contributed to the completion of this thesis.

Chapter 2

Low-Rank Matrix Factorization

Summary of the chapter. This chapter attempts to provide a unifying survey of existing results. We define the basic problems that motivate the work in this thesis: Approximate Matrix Factorization (`AMF`) and Constrained Approximate Matrix Factorization (`CAMF`). We sketch some applications of these problems in signal processing and data mining and also present algorithms for their solution.

At the basis of our discussion lies the following mathematical problem: Given an $n \times k$ real-valued matrix Y, we want to write it as a product $Y = AX$, where the matrices A and X have dimensions $n \times m$ and $m \times k$, respectively, and $m < \min(n, k)$. If Y is not degenerate, then it will be $\operatorname{rank}(Y) = \min(n, k)$, while for the product it always holds $\operatorname{rank}(AX) \leq \min(\operatorname{rank}(A), \operatorname{rank}(X)) \leq m < \min(n, k)$. Therefore, if Y is not degenerate, its exact factorization as a low-rank product is not possible. Instead, we seek an approximate solution, i.e., we want to compute the matrices A and X such that the quantity $\|Y - AX\|$ is minimized, for some appropriate proximity measure $\| \cdot \|$. Below, we use Frobenius norm as a straightforward measure of elementwise proximity. Let us recall that the Frobenius norm of an $n \times m$ matrix $A = (a_{ij})$ is defined as $\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2}$. We will refer to this problem as Approximate Matrix Factorization (`AMF`).

Problem 1. Approximate Matrix Factorization (`AMF`)
Given is an $n \times k$ real-valued matrix Y. We want to compute an $n \times m$ matrix A and an $m \times k$ matrix X, with $m < \min(n, k)$, such that $\|Y - AX\|_F$ is minimized.

Let us point out from the beginning that `AMF` is an inherently ill-defined problem. To see that, let us assume that we have already computed a global or local minimum (A, X) for `AMF`. Clearly, many other pairs $(\widehat{A}, \widehat{X})$ may exist that satisfy $\widehat{A}\widehat{X} = AX$ and are, therefore, indistinguishable from (A, X) in terms of how well they approximate Y. For example, let us consider $\widehat{A} = AR$ and $\widehat{X} = R^{-1}X$ for an arbitrary, invertible, $m \times m$ matrix R. This inherent non-uniqueness of solutions and how it can be alleviated is the topic that motivates much of the material in this thesis. Below we show that, due to its generality, `AMF` appears in many different areas of science and engineering.

2.1 Signal processing: Blind signal separation

Let us consider the following generic signal processing application. There is a set of n sensors that monitor m signal sources over k discrete time points. We assume that, in general, each sensor measures a mixture of signals from more than one source and that the sensor measurements depend linearly on the source signals. Both the values of the source signals and the values of the mixing coefficients are unknown, and this is exactly our task: Given the sensor measurements over the k time points, we want to infer the source signals over the k time points and the mixing coefficients. This problem is also known as *blind signal separation*, since we have no or very little information about the source signals and the mixing process.

In formal terms, we are given an $n \times k$ matrix $Y = (y_{it})$ where y_{it} is the signal measured at sensor i at time point t. We want to express Y as a product $Y = AX$ of an $n \times m$ matrix $A = (a_{ij})$, where a_{ij} is the mixing coefficient between sensor i and signal source j, and an $m \times k$ matrix $X = (x_{jt})$ that contains the source-signal intensities at different time points.

In Chapters 7 and 8, we will see two applications of blind signal separation in bioinformatics. Firstly, in the context of transcriptomics, it can be used to analyze measurements from DNA microarrays that contain non-specific probes. In this case, the source signals are the mRNA target concentrations and the sensor readouts are the probe intensities. In the context of gene regulatory networks, it has been a successful approach for determining the activities of transcription factors from transcriptome data. In this case, the source signals are the transcription factor activities and the sensors are the regulated genes. For exact descriptions of these applications we refer to the corresponding chapters.

2.2 Data mining: Dimension reduction

In applications of data mining and pattern recognition, a typical dataset consists of a set of n objects that are represented as vectors in a k-dimensional feature space. Such a dataset can be represented as a $n \times k$ matrix $Y = (y_{ij})$, where y_{ij} is the value of the j-th feature for the i-th object. Data mining practitioners often want to represent the objects of their studies as vectors in an m-dimensional space, where $m < k$. Such a low-dimensional representation can be, for example, beneficial for the performance of learning and data exploration algorithms, as well as for the interpretation and visualization of the results.

A plethora of papers published over the years propose different methods to actually perform dimension reduction (for a review we refer to [Fod02]). Here, we only mention that these methods can be roughly classified into two broad categories: feature selection and feature extraction methods. The first aim at choosing a small subset of the initial features that capture the crucial properties of the objects, while the latter try to discover a set of "summarizing" new features that can be computed from the initial features.

Let us motivate our discussion with a classical example from text mining [BDJ99, SBPP06]. In this application, Y is the so-called document-term matrix, where the rows correspond to documents of a text corpus, the columns correspond to the terms of a finite vocabulary and the entry y_{ij} is the (normalized) appearance frequency of the j-th term in the i-th document. We intuitively expect that our text corpus is built on top of an underlying set of a few topics that are, however, hidden and corrupted by the wide variety of used words. If we could somehow discover the hidden topics, then it would be beneficial to represent the document vectors in the m-dimensional topic space instead of representing them in the initial k-dimensional term space. In this way, the negative effects of polysemy and synonymy[1] on query-based text retrieval would be made less severe.

Below we assume that we have n documents, k terms and $m < \min(n, k)$ hidden topics. With the goal of discovering the hidden topics we can try a feature extraction approach that is based on a simple linear model of text generation. We assume that the i-th document discusses the ℓ-th topic with strength $a_{i\ell}$ and that the ℓ-th topic involves the j-th term of the vocabulary with strength $x_{\ell j}$. According to the linear model, the observed frequency y_{ij} would then be explained as $y_{ij} = \sum_{\ell=1}^{m} a_{i\ell} x_{\ell j}$. This imediately gives rise to the matrix equation $Y = AX$, where both matrices A and X have to be estimated from the observed frequencies Y. This is exactly the `AMF` problem.

2.3 Introducing a-priori knowledge about the factors

In many applications that give rise to the `AMF` problem, we have an additional intuitive understanding of the underlying physical model. For instance, in the text mining application of Section 2.2, for the text generation model to make sense it is reasonable to require that the matrices A and X should have nonnegative entries.

Let us now consider a case of blind signal separation, with the signal sources being sound emitters (e.g. human speakers) and the sensors being microphones. Nonnegativity of the factors is a reasonable requirement also in this application, but in this case we may even be able to derive more constraints from our understanding of the physical system. For example, if we know in advance the distances between the microphones and the sound sources, we can draw some conclusions about the magnitudes of

[1]Polysemy: the same word can have multiple meanings. Synonymy: many different words can have the same meaning.

the mixing coefficients; below we denote with d_{ij} the distance between the i-th microphone and the j-th source. For example, if $d_{ij} \gg d_{ik}$, it is reasonable to expect that $a_{ij} \gg a_{ik}$. On the other hand, if $d_{ij} \approx d_{ik}$ we can expect that $|a_{ij} - a_{ik}| < \epsilon$, where ϵ is an appropriately chosen threshold. Going one step further, if we have available a physical model that roughly predicts the mixing coefficient a_{ij} as a function of d_{ij}, we could even make a statement about the absolute value of the mixing coefficient. Finally, if the distance d_{ij} is larger than an appropriate threshold we can expect $a_{ij} = 0$. We will later see that such an a-priori knowledge of some entries of A being fixed to zero has some strong implications for the identifiability of the model.

Taking into account the physically motivated constraints complicates our basic problem: Given a matrix Y, we want to write it as an approximate low-rank product $Y \approx AX$ and, now, we additionally want to enforce some constraints on the factors A and X. Below we will use the notation $A \lhd \mathcal{C}$ to denote the generic statement that a matrix A satisfies a set of constraints $\mathcal{C}$. We will restrict ourselves to the case in which $\mathcal{C}$ consists only of *linear equalities and inequalities* that involve the elements of A in arbitrary combinations. In fact, several useful classes of constraints that arise in practice can be expressed in that form.

Problem 2. Constrained Approximate Matrix Factorization (`CAMF`)
Given is an $n \times k$ real-valued matrix Y and two sets of linear constraints $\mathcal{C}_A$ and $\mathcal{C}_X$. We want to compute an $n \times m$ matrix A and an $m \times k$ matrix X, with $m < \min(n, k)$, such that $A \lhd \mathcal{C}_A$, $X \lhd \mathcal{C}_X$ and $\|Y - AX\|_F$ is minimized.

As it was the case with the unconstrained problem `AMF`, also for `CAMF` there are, in general, many different feasible (A, X) pairs that approximate a given Y equally well. On the other hand, we intuitively expect that the enforcement of the constraints $\mathcal{C}_A$ and $\mathcal{C}_X$ must somehow offer some control over the space of possible solutions. In Chapter 3 we investigate an especially strong class of constraints, where $\mathcal{C}_A$ specifies that certain entries in A are fixed to zero.

2.4 Algorithmic issues

2.4.1 Unconstrained problem `AMF`

In this section we will focus on the algorithmic aspects of `AMF` and `CAMF`: Given a matrix Y, how can we actually compute a pair (A, X) that minimizes $\|Y - AX\|_F$ (under the constraints $\mathcal{C}_A$ and $\mathcal{C}_X$)? At first we discuss the unconstrained `AMF` problem. It turns out that it can be efficiently solved to optimality from the singular value decomposition of Y; for its constrained counterpart `CAMF` the situation is more complicated. The singular value decomposition (SVD), as summarized in Theorem 2.1, is a fundamental theoretical result in linear algebra that also has tremendous practical impact. For the following two standard results and for efficient and stable algorithms to numerically compute the SVD we refer to the classical book of Golub and Van Loan [GL96].

Theorem 2.1. *Any $n \times k$ matrix Y (w.l.o.g. we assume that $n \geq k$) can be factorized as $Y = U\Sigma V^T$, where the matrices U, Σ and V have the following properties:*

- *U has dimensions $n \times k$ and satisfies $U^T U = I$.*
- *Σ has dimensions $k \times k$ and is diagonal, with all its entries being nonnegative.*
- *V has dimensions $k \times k$ and satisfies $V^T V = I$.*

The columns of U and V^T are called left and right singular vectors of Y, respectively. The entries of Σ are called singular values of Y; usually we assume that the columns of U and the rows of V^T are permuted such that the singular values σ_i are sorted in decreasing order on the diagonal of Σ. The following theorem states that the best m-rank approximation of a matrix Y, with $m < \text{rank}(Y)$, can be computed from the SVD of Y. In particular, it suffices to truncate the SVD product by keeping only the m largest singular values and the corresponding singular vectors.

Theorem 2.2. *We consider the $n \times k$ matrix Y, with rank$(Y) > m$. The matrix approximation problem*

$$\min_Z \|Y - Z\|_F \quad s.t. \quad rank(Z) = m$$

has the solution $Z^* = U_m \Sigma_m V_m^T$, where the matrices U_m, Σ_m and V_m^T result from the SVD matrices by keeping only the m largest singular values and the corresponding singular vectors. Moreover, the minimum value of the objective function is

$$\|Y - Z^*\|_F = \sqrt{\sum_{i=m+1}^{\min(n,k)} \sigma_i^2}.$$

Therefore, we can solve the unconstrained problem `AMF` by just taking $A = U_m \Sigma_m$ and $X = V_m^T$. Of course, this is only one of infinitely many different and equally good solutions.

2.4.2 Constrained problem `CAMF`

The enforcement of the constraints $\mathcal{C}_A$ and $\mathcal{C}_X$ makes the problem significantly harder. The optimal solution of the unconstrained problem obtained with SVD will not be, in general, feasible for the constrained problem. Moreover, the objective function is not convex with respect to the variables (A, X), so standard convex optimization methods cannot be applied for the efficient computation of a global minimum. If it is critical to find the global minimum, methods from global optimization can be employed, keeping, however, in mind that their complexity can be prohibitively large. Alternatively, if a local minimum is sufficient for our purposes, we can approach `CAMF` with Algorithm 1 that is described below. In fact, in Algorithm 1 we have replaced `CAMF` with the equivalent problem that results from squaring the objective function. The algorithm first starts with a guess for matrix A that satisfies the constraints $\mathcal{C}_A$ and then alternates between two minimization steps until convergence. In each one of these steps, one matrix is considered fixed and minimization is performed with respect to the other matrix. Under some mild assumptions on the constraints, the algorithm is guaranteed to converge to a critical point of the objective function, which, not surprisingly, depends on the initialization of A. Therefore, in order to increase the chance to find a good minimum, a common practical approach is to run the algorithm several times with different initializations.

Algorithm 1 Alternating minimization algorithm for the solution of `CAMF`.

```
1: $n \leftarrow 0$
2: Initialize $A_n$ with values that satisfy $\mathcal{C}_A$
3: $X_n \leftarrow \text{argmin}_X \|Y - A_0 X\|_F^2 \quad \text{s.t. } X \lhd \mathcal{C}_X$
4: **repeat**
5:     $n \leftarrow n + 1$
6:     $A_n \leftarrow \text{argmin}_A \|Y - A X_{n-1}\|_F^2 \quad \text{s.t. } A \lhd \mathcal{C}_A$
7:     $X_n \leftarrow \text{argmin}_X \|Y - A_n X\|_F^2 \quad \text{s.t. } X \lhd \mathcal{C}_X$
8: **until** $\|A_n X_n - A_{n-1} X_{n-1}\|_F^2 < \epsilon$
9: **return** $(A_n, X_n)$
```

Although for the original problem `CAMF` it is hard to compute a global minimum, it is important to note that in Algorithm 1 the minimization subproblems of lines 6 and 7 can be efficiently solved to optimality. Let us consider, for example, the minimization of line 7 (the minimization of line 6 is symmetric):

$$\min_X \|Y - AX\|_F^2 \quad \text{s.t.} \quad X \lhd \mathcal{C}_X \tag{2.1}$$

Let $\widehat{A}$ be the block diagonal matrix built from k copies of A and let $\widehat{y}$ and $\widehat{x}$ be the vectors that result from stacking on top of each other the columns of Y and X, respectively:

$$\widehat{A} = \begin{pmatrix} A & 0 & \cdots & 0 \\ 0 & A & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A \end{pmatrix} \quad , \quad \widehat{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} \quad , \quad \widehat{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$$

Then, the matrix minimization problem (2.1) becomes:

$$\min_{\widehat{x}} \|\widehat{y} - \widehat{A}\widehat{x}\|_2^2 \quad \text{s.t.} \quad \widehat{x} \lhd \mathcal{C}_{\widehat{x}} \tag{2.2}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. Let us recall that the Euclidean norm of an n-dimensional vector $x = (x_i)$ is defined as $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$. Since $\mathcal{C}_{\widehat{x}}$ consists only of linear constraints, (2.2) is a Euclidean norm approximation (also known as least-squares) problem with linear constraints. This can be efficiently solved to global optimality. In fact, such problems have been an active area of research for a long time and highly optimized algorithms have been developed for their solution [HH82]. We note that such a problem is, in fact, a special case of a convex quadratic program, i.e., minimization of a convex quadratic function under linear equality and inequality constraints. Therefore, it can be in principle solved with a generic interior point solver [BV04].

2.4.3 Convergence of alternating minimization

Algorithm 1 is, in fact, a special case of a more general minimization scheme that is called block-nonlinear Gauss-Seidel (block-GS) method and works as follows. Given a multivariate objective function that has to be minimized under some constraints, we first partition the variables into disjoint blocks. We start with an initial feasible point and at each iteration we minimize with respect to the variables of one block, while keeping the variables of all other blocks fixed. We alternate this partial minimization step cyclically through all variable blocks with the hope of convergence.

It is known that, in general, the block GS minimization will not converge, in the sense that it may produce a sequence with limit points that are *not critical points* of the objective function. On the other hand, a number of studies have given convergence results for the unconstrained and constrained case under suitable convexity assumptions. Moreover, in the special case of having only two variable blocks, Grippo and Sciandrone [GS00] proved that every limit point is a critical point, even *without any convexity assumption* on the objective function. Below, we summarize their result for this important special case; the objective function is $f : \mathbb{R}^{n_1+n_2} \mapsto \mathbb{R}$ and x_1, x_2 stand for the two blocks of variables.

Problem 3.

$$\begin{aligned} \text{minimize:} \quad & f(x_1, x_2) \\ \text{subject to:} \quad & x_1 \in X_1 \subseteq \mathbb{R}^{n_1} \\ & x_2 \in X_2 \subseteq \mathbb{R}^{n_2} \end{aligned}$$

For Problem 3, the k-th iteration of the block-GS method is:

$$x_1^{k+1} = \operatorname{argmin}_{y_1 \in X_1} f(y_1, x_2^k) \tag{2.3}$$

$$x_2^{k+1} = \operatorname{argmin}_{y_2 \in X_2} f(x_1^{k+1}, y_2) \tag{2.4}$$

Theorem 2.3 ([GS00]). *We assume that (i) f is continuously differentiable, (ii) the sets X_1, X_2 are closed, nonempty and convex, and (iii) the minimization subproblems (2.3) and (2.4) have an optimal solution. Then, every limit point of the sequence $\{(x_1^k, x_2^k)\}$ generated by the block-GS method is a critical point of Problem 3.*

It turns out that Algorithm 1 is a special case of the two-block GS method, where the matrices A and X are the two blocks of variables. Furthermore, the properties of Theorem 2.3 hold. Firstly, the objective function is continuously differentiable, i.e., the first-order partial derivatives exist and are continuous:

$$\frac{\partial \|Y - AX\|_F^2}{\partial A} = 2(AX - Y)X^T \quad , \quad \frac{\partial \|Y - AX\|_F^2}{\partial X} = 2A^T(AX - Y)$$

Secondly, the feasible sets A and X are convex, since we only impose linear constraints on the matrix entries a_{ij} and x_{ij}. In order to guarantee that they are also closed, we can in practice add some very loose lower and upper bounds on a_{ij} and x_{ij}. Thirdly, each one of the minimization subproblems has a global optimum, which we can efficiently compute.

2.5 A special case: Nonnegative matrix factorization

The special case of `CAMF` where the only constraint applied on the two factors is nonnegativity, deserves a more detailed discussion. In fact, this is the so-called *nonnegative matrix factorization (NMF)*, a method that has been introduced in [LS99] and since then succesfully used in many different applications. The popularity of NMF lies in the fact that in many applications nonnegativity is an essential (and the only) assumption for the validity of the model.

In Section 2.2 we sketched with an example from text mining how low-rank matrix factorization can be used for dimension reduction, and in Section 2.3 we pointed out that nonnegativity of the factors is a reasonable requirement. In fact, the same scheme of explaining a set of observations as linear combinations of some hidden factors has been applied to many other scientific disciplines. These include image recognition [LS99], acoustics [VBB08, Beh03], video summarization [CF02] and Internet research [LXY03].

Being to a large extent a data-driven discipline, bioinformatics has also witnessed the use of matrix factorization methods for dimension reduction. A typical application comes from the field of microarray studies, where the rows of matrix Y correspond to samples, the columns to genes and the matrix entries are expression values. The result of the factorization can be interpreted as projecting the sample-vectors onto a low-dimensional space of "metagenes"; this representation is then used to assign the samples to classes of different disease types [BTGM04, KP07]. Other applications of NMF in bioinformatics include biclustering of microarray data [CSPMT$^+$06], discovery of gene relationships from the biomedical literature [CCSS$^+$06] and analysis of regulatory motifs [HMSG08].

Due to the wide popularity of NMF, its algorithmic aspects have been intensively studied. Namely, in the place of Algorithm 1, many methods from the gradient descent family have been proposed [BBL$^+$07, Lin07]. In practice, these methods run faster than the alternating minimization described above, because their update rules (taking a step into the direction of the gradient) can be implemented with a few smart matrix operations. However, as opposed to alternating minimization, their convergence behavior is a black spot in the literature.

Furthermore, several variants of NMF have been developed that explicitly enforce a minimum sparsity, i.e., that the factors A, X contain at least a specified number of zeros [Hoy04, HS06]. Sparsity of the factors is a desired property in dimension reduction, because it contributes to an easier interpretation of the results. However, in a typical application the exact sparsity pattern of the factors is *not known a-priori*, because the purpose of empirical data mining is exactly to explore the structure that is hidden in the data, when there are no strong a-priori assumptions. In contrast, in this work we will investigate what happens, if the exact sparsity pattern of one of the factors is fixed a-priori.

Chapter 3

Identifiability in Low-Rank Matrix Factorization

Summary of the chapter. We focus on a special case of `CAMF`, where in the factorization $Y \approx AX$ one of the factors must have zeros at certain positions. This class of constraints arises in applications that involve a bipartite network of sources that are emitting some signals over discrete time and sensors that are monitoring these signals. The question about the uniqueness of the solutions leads to the definition of identifiable bipartite graphs, which is crucially based on bipartite matchings. Since identifiable graphs are the basic objects of our study, we present some of their properties that may provide some insight into their combinatorial structure. Finally, we turn our attention to non-identifiable graphs and, using the Dulmage-Mendelsohn decomposition, we show how we can draw some useful structural conclusions even in these cases.

Let us consider the following generic setting: There is a bipartite network of n observable sensors that monitor m hidden signal sources over k discrete time points. In general, each sensor measures a mixture of signals from more than one source. We assume that the connectivity between the sensors and the sources is known and, moreover, that the sensor measurements depend linearly on the source signals. However, the exact values of the mixing coefficients are unknown, except for those that are constrained to zero as a consequence of the connectivity of the source-sensor network. Given the sensor measurements over the k time points, our task is to infer the source signals over the k time points and the non-zero mixing coefficients.

In formal terms, we are given an $n \times k$ matrix $Y = (y_{it})$ where y_{it} is the signal measured at sensor i at time point t. We want to express Y as a product $Y = AX$ of an $n \times m$ matrix $A = (a_{ij})$, where a_{ij} is the mixing coefficient between sensor i and signal source j, and an $m \times k$ matrix $X = (x_{jt})$ that contains the source-signal intensities at different time points. We assume that $m < \min(n, k)$ (low-rank factorization). The sparsity structure of A that is known a priori from the connectivity of the network is modeled with a 0/1 matrix $\mathcal{Z}$, called *zero pattern*. We say that a real-valued matrix A *satisfies* a zero pattern $\mathcal{Z}$ (denoted by $A \lhd \mathcal{Z}$), if A and $\mathcal{Z}$ have the same dimensions and $\mathcal{Z}_{ij} = 0$

implies $A_{ij} = 0$. In fact, we are dealing with a special case of the CAMF problem of Section 2.3. Here, we seek a matrix pair (A, X) such that $A \lhd \mathcal{Z}$ and $\|Y - AX\|_F$ is minimized; other constraints $\mathcal{C}_A$ and $\mathcal{C}_X$ may be also available, but in our discussion below we will only use the zero pattern $\mathcal{Z}$. We will refer to this optimization problem as *Zero Constrained Approximate Matrix Factorization* (ZCAMF).

Let us see how the a-priori knowledge of the zero pattern $\mathcal{Z}$ appears in the two applications from bioinformatics that we have already sketched in Section 2.1. In the context of microarrays, the connectivity between signal sources (mRNA targets) and sensors (probes) can be derived from the sequence similarity between the probes and the targets. In the context of regulatory networks, the sources are the transcription factor activities and the sensors are the regulated genes; the connectivity of the network is known either from lab experiments or from sequence-based in-silico prediction of binding sites.

Here, we do not focus on how we can actually compute a global or local minimum for ZCAMF; in Section 2.4.2 we saw that alternating minimization (see Algorithm 1) is one possible way to go. Instead, we focus on the inherent (non-)identifiability of the model. Assume that we have obtained a solution (A, X) for AMF. Many other pairs $(\widehat{A}, \widehat{X})$ may exist that satisfy $\widehat{A}\widehat{X} = AX$ and $\widehat{A} \lhd \mathcal{Z}$, and are therefore indistinguishable from (A, X) in terms of how well they approximate Y.

The zero pattern $\mathcal{Z}$ (equivalently the connectivity of the bipartite source-sensor graph) restricts the $(\widehat{A}, \widehat{X})$-space. In Section 3.1, we characterize the bipartite graphs that, under mild non-degeneracy assumptions on A and X, make the AMF model identifiable up to diagonal scaling. This characterization is based on the notion of structural rank from combinatorial matrix theory and relies on maximum bipartite matchings.

3.1 Definition of identifiable graphs

Given a matrix pair (A, X) and a zero pattern $\mathcal{Z}$ with $A \lhd \mathcal{Z}$, we first characterize the class of matrix pairs $(\widehat{A}, \widehat{X})$ such that $\widehat{A}\widehat{X} = AX$ and $\widehat{A} \lhd \mathcal{Z}$.

Proposition 3.1. *Let A and $\widehat{A}$ be $n \times m$ matrices, and X and $\widehat{X}$ be $m \times k$ matrices with $AX = \widehat{A}\widehat{X}$ and such that rank(A) = rank$(X) = m$. Then there exists an invertible $m \times m$ matrix R such that $\widehat{A} = AR$ and $\widehat{X} = R^{-1}X$.*

Proof. From $AX = \widehat{A}\widehat{X}$, we get $A^TAX = A^T\widehat{A}\widehat{X}$. Since $\text{rank}(A) = m$, A^TA is non-singular and we can write $X = R\widehat{X}$, where $R = (A^TA)^{-1}A^T\widehat{A}$. From $X = R\widehat{X}$ we get $m = \text{rank}(X) \leq \min\{\text{rank}(R), \text{rank}(\widehat{X})\}$. On the other hand, from the dimensions of R and $\widehat{X}$, we have that $\text{rank}(R) \leq m$ and $\text{rank}(\widehat{X}) \leq \min\{m, k\}$ and we conclude that $\text{rank}(R) = \text{rank}(\widehat{X}) = m$. From the equality $X = R\widehat{X}$ and the non-singularity of R

$$\overbrace{\begin{pmatrix} \mathbf{0} & a_{12} & a_{13} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{24} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{34} \\ a_{41} & \mathbf{0} & a_{43} & \mathbf{0} \\ a_{51} & a_{52} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & a_{62} & \mathbf{0} & a_{64} \end{pmatrix}}^{A} \overbrace{\begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix}}^{R} \lhd \overbrace{\begin{pmatrix} \mathbf{0} & 1 & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \\ 1 & \mathbf{0} & 1 & \mathbf{0} \\ 1 & 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & 1 \end{pmatrix}}^{\mathcal{Z}} \quad \text{leads to}$$

$$\overbrace{\begin{pmatrix} a_{12} & a_{13} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & a_{24} \\ \mathbf{0} & \mathbf{0} & a_{34} \\ a_{62} & \mathbf{0} & a_{64} \end{pmatrix}}^{\widetilde{A}_1} \overbrace{\begin{pmatrix} r_{21} \\ r_{31} \\ r_{41} \end{pmatrix}}^{\widetilde{r}_1} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad \overbrace{\begin{pmatrix} \mathbf{0} & \mathbf{0} & a_{24} \\ \mathbf{0} & \mathbf{0} & a_{34} \\ a_{41} & a_{43} & \mathbf{0} \end{pmatrix}}^{\widetilde{A}_2} \overbrace{\begin{pmatrix} r_{12} \\ r_{32} \\ r_{42} \end{pmatrix}}^{\widetilde{r}_2} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix},$$

$$\overbrace{\begin{pmatrix} \mathbf{0} & \mathbf{0} & a_{24} \\ \mathbf{0} & \mathbf{0} & a_{34} \\ a_{51} & a_{52} & \mathbf{0} \\ \mathbf{0} & a_{62} & a_{64} \end{pmatrix}}^{\widetilde{A}_3} \overbrace{\begin{pmatrix} r_{13} \\ r_{23} \\ r_{43} \end{pmatrix}}^{\widetilde{r}_3} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad \overbrace{\begin{pmatrix} \mathbf{0} & a_{12} & a_{13} \\ a_{41} & \mathbf{0} & a_{43} \\ a_{51} & a_{52} & \mathbf{0} \end{pmatrix}}^{\widetilde{A}_4} \overbrace{\begin{pmatrix} r_{14} \\ r_{24} \\ r_{34} \end{pmatrix}}^{\widetilde{r}_4} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}.$$

Figure 3.1: An example of a zero pattern $\mathcal{Z}$, a general matrix $A \lhd \mathcal{Z}$, and the four linear systems derived from A, $\mathcal{Z}$, and the condition $AR \lhd \mathcal{Z}$.

we get $\widehat{X} = R^{-1}X$. The same equality, combined with $AX = \widehat{A}\widehat{X}$, gives $AR\widehat{X} = \widehat{A}\widehat{X}$. Since $\operatorname{rank}(\widehat{X}) = m$, we finally get $\widehat{A} = AR$. □

Under the full-rank assumption of Proposition 3.1 the entries of R are restricted by the condition $\widehat{A} = AR \lhd \mathcal{Z}$. Only zeros in $\mathcal{Z}$ induce constraints: $\mathcal{Z}_{ij} = 0$ implies that $\sum_\ell a_{i\ell} r_{\ell j} = 0$. This leads to m linear systems, one for each column of R. In particular, the j-th column of R (denoted by r_j) is constrained by the linear system $A_j r_j = 0$, where A_j stands for the submatrix of A that results from taking only the rows where the j-th column of $\mathcal{Z}$ has zeros. Since $A \lhd \mathcal{Z}$, the j-th column of A_j is all zero. Therefore, the system $A_j r_j = 0$ can be simplified to $\widetilde{A}_j \widetilde{r}_j = 0$, where $\widetilde{A}_j$ results from A_j by dropping the j-th column and $\widetilde{r}_j$ results from r_j by dropping the j-th element. An example is shown in Figure 3.1.

The column rank of the matrix $\widetilde{A}_j$ determines the degrees of freedom of the j-th column of R. If $\operatorname{rank}(\widetilde{A}_j) = m - 1$, then the corresponding linear system only allows the zero solution and the j-th column of R has all its entries, except for the j-th, constrained to zero. If the condition $\operatorname{rank}(\widetilde{A}_j) = m - 1$ holds for all columns $j \in \{1, \ldots, m\}$, then R is a diagonal matrix. Let us recall from Proposition 3.1 that $\widehat{A} = AR$ and $\widehat{X} = R^{-1}X$; therefore, if R and consequently R^{-1} are diagonal, then $\widehat{A}$ differs from A only by column scaling and $\widehat{X}$ differs from X only by row scaling. In many applications, diagonal scaling does not affect the interpretation of the results, since it can be eliminated with normalization. In this sense, the solution (A, X) of `ZCAMF` is essentially unique.

The ranks of the submatrices $\widetilde{A}_j$ depend on both the structure of $\mathcal{Z}$ and on the values of the free a_{ij} entries. In the example of Figure 3.1, we note that the rank of $\widetilde{A}_2$ can be at most 2, for any choice of values of a_{ij}. On the other hand, the ranks of $\widetilde{A}_1$, $\widetilde{A}_3$ and $\widetilde{A}_4$ can be 3 or smaller, depending on the numerical values of a_{ij}. While the rank degeneracy of $\widetilde{A}_2$ is due to the "bad structure" of $\mathcal{Z}$, only adversarily chosen values of A would make the other $\widetilde{A}_j$ rank-degenerate. Below we formalize the notion of a "badly structured" zero pattern. To do that, we need to introduce the concept of structural rank, which will be the link between linear algebra and graph theory. Then, Proposition 3.4 relates the structural rank of a zero pattern with the numerical rank of the matrices that satisfy this pattern. Both Definition 3.3 and Proposition 3.4 can be found, e.g., in the book of Murota [Mur00].

Definition 3.2. A matching M in a graph G is a subset of edges such that no two edges in M share a vertex. The size of a maximum matching in G is called *matching number* and it is denoted by $\nu(G)$.

In fact, the structural rank of a zero pattern is defined as the matching number of an appropriate graph. More specifically, we represent a zero pattern $\mathcal{Z}$ as a bipartite graph, where the two vertex partitions correspond to the columns and rows of $\mathcal{Z}$ and the edges correspond to the ones in $\mathcal{Z}$.

Definition 3.3. The *structural rank* of a zero pattern $\mathcal{Z}$, denoted by $\operatorname{srank}(\mathcal{Z})$, is the matching number of the corresponding bipartite graph.

Proposition 3.4. *If a matrix B satisfies a zero pattern $\mathcal{Z}$, then $\operatorname{rank}(B) \leq \operatorname{srank}(\mathcal{Z})$.*

Let us now assume that B and $\mathcal{Z}$ both have dimensions $n \times m$, with $n \geq m$, and $\operatorname{srank}(\mathcal{Z}) = m$. We intuitively expect that if the non-zero entries of B are "generic", then $\operatorname{rank}(B) = m$. Below, we give a semiformal argument for this statement. We know that $\operatorname{rank}(B)$ equals the order of the largest non-vanishing minor in B. Therefore, $\operatorname{rank}(B) < m$ if and only if for each subset of rows I, with $|I| = m$, it holds $\det(B_I) = 0$; here B_I denotes the submatrix that results from B by taking only the rows in I. Now, taking all possible choices of I into account, the set of conditions of the form $\det(B_I) = 0$ gives rise to a system of $\binom{n}{m}$ multivariate polynomial equations with respect to the variables b_{ij}. The zero set of any non-zero polynomial is very "thin"; by Sard's theorem it has Lebesgue measure zero [Sar42]. This implies that the zero set of the polynomial system, i.e., the set of assignments of b_{ij} that make $\operatorname{rank}(B) < m$, also has measure zero. Therefore, a rank-degenerate matrix B appears with zero probability, if the entries b_{ij} are drawn from any reasonable continuous distribution.

In our example, the structure of $\mathcal{Z}$ immediately gives a certificate for the non-identifiability of `ZCAMF`. More specifically, $\operatorname{srank}(\widetilde{\mathcal{Z}}_2) = 2$ and this is a structural reason that makes $\operatorname{rank}(\widetilde{A}_2) < 3$ and the model non-identifiable. This is illustrated in Figure 3.2.

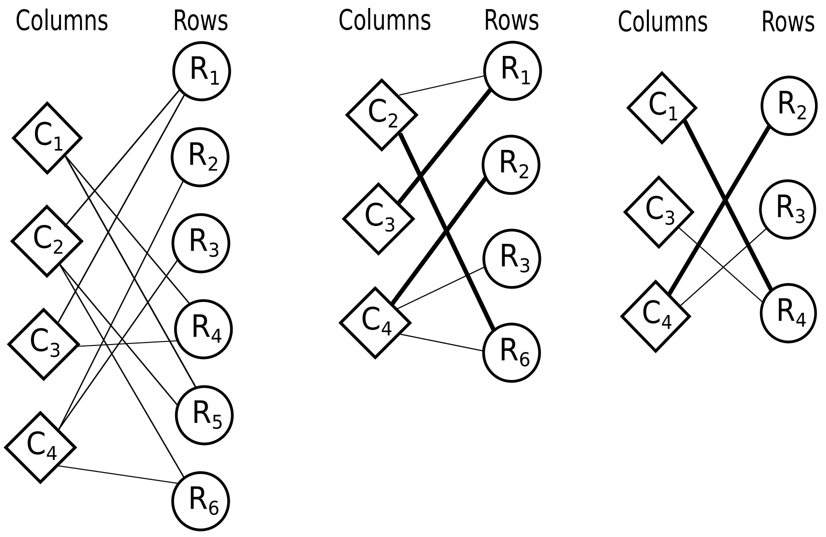


Figure 3.2: Example of Figure 3.1 continued. Left: graph of $\mathcal{Z}$. Center: graph of $\widetilde{\mathcal{Z}}_1$; srank$(\widetilde{\mathcal{Z}}_1) = 3$. Right: graph of $\widetilde{\mathcal{Z}}_2$; srank$(\widetilde{\mathcal{Z}}_2) = 2$. The graph of $\widetilde{\mathcal{Z}}_j$, for $j = 1, \ldots, 4$, results from the graph of $\mathcal{Z}$ by discarding the column-vertex C_j and its neighbors. The maximum matchings are drawn with thick lines.

Let us recall that in the generic signal processing application described at the beginning of this chapter the zero pattern $\mathcal{Z}$ arises from the known connectivity of the source-sensor network. In applications, we would like to avoid source-sensor networks that give rise to such inherently problematic zero patterns as the one of Figure 3.1. This motivates the following definition, where, from the application's point of view, C represents the set of sources and R represents the set of sensors. With $N(x)$ we denote the set of neighbors of a vertex x.

Definition 3.5. Let $G = (C, R; E)$ be a bipartite graph with at least one edge. A vertex $x \in C$ is called *identifiable*, if the subgraph induced by $(C \setminus \{x\}) \cup (R \setminus N(x))$ has a matching of cardinality $|C| - 1$. The whole graph G is called identifiable, if *all* vertices of C are identifiable.

We should emphasize that an identifiable source-sensor graph G *does not guarantee* the uniqueness up to scaling of `AMF`'s solution (A, X), but rather avoids a possible reason for its non-uniqueness. Even if G is identifiable, it is possible that the non-zero entries of A have such values that at least one of the submatrices $\widetilde{A}_j$ is rank-degenerate. However, according to our previous discussion, such assignments of the a_{ij} entries belong to the union of m measure-zero sets, and altogether they have measure zero. Intuitively, if G is identifiable and in the solution (A, X) the non-zero entries of A have generic values, then (A, X) is unique up to scaling. In fact, this genericity assumption is reasonable for numbers that correspond to physical quantities.

For a bipartite graph $G = (C, R; E)$ and vertex sets $X \subseteq C$, $Y \subseteq R$, we will denote with $G[X, Y]$ the subgraph of G induced by $X \cup Y$. For brevity, we will use $\widetilde{G}_x$ to denote the subgraph $G[C \setminus \{x\}, R \setminus N(x)]$.

Given a bipartite graph $G = (C, R; E)$, we can efficiently test if it is identifiable, by computing the matching numbers of the subgraphs $\widetilde{G}_x$, for all $x \in C$. Finding a maximum matching in a (not necessarily) bipartite graph can be done in polynomial time; more details about matching algorithms are given in the following section.

3.2 A digression concerning bipartite matchings

For most of the discussions in this thesis, Definition 3.5 will play the central role. In order to gain a working knowledge of this definition, we have to briefly review some classic graph-theoretical results concerning bipartite matchings. Given a bipartite graph $G = (C, R; E)$, we call a matching *C-perfect* if it has cardinality $|C|$, i.e., covers all vertices of partition C. The most fundamental result about bipartite matchings is Hall's marriage[1] theorem that completely characterizes the bipartite graphs that have a perfect matching. Below, we denote with $N(X)$ the neighborhood of a vertex set X, i.e., the set of all vertices that are adjacent to at least one vertex from X: $N(X) = \cup_{x \in X} N(x)$.

Theorem 3.6 ([Hal35]). *Let $G = (C, R; E)$ be a bipartite graph. There is a C-perfect matching in G if and only if $|N(X)| \geq |X|$, for all subsets $X \subseteq C$.*

Although theoretically appealing, Hall's theorem does not readily provide an *efficient* algorithm for checking the existence of a C-perfect matching, since a naive application of it would require checking all $2^{|C|} - 1$ non-empty subsets of C. Moreover, if a perfect matching does not exist, we usually want to compute the matching number of the graph, i.e., find a matching that is as large as possible. For the algorithmic treatment of maximum matchings, Theorem 3.8 comes as a rescue. But for its understanding we first need to introduce the notion of augmenting path.

Given is a (not necessarily bipartite) graph G and a (not necessarily maximum) matching M in G. The edges of M are called *matched* and the other edges are called *free.* Similarly, a vertex is called matched if it is endpoint of a matched edge, otherwise it is called free. An augmenting path with respect to M (briefly *M-augmenting path*) is a path in G such that its edges are free and matched in alternation and both endpoints are free. It is easy to show Proposition 3.7: If we can find an M-augmenting path in G, then we can construct a matching that is larger than M.

[1]The term "marriage" shows that in some (rare) cases mathematics can have a positive impact on people's personal lives. We consider the following scenario: C is a set of girls, R is a set of boys and E are the pairwise attraction relationships. A C-perfect matching is an assignment of boys to girls such that each girl gets married to a boy she finds attractive.

Proposition 3.7 ([Ber57])**.** *Given a matching M and an M-augmenting path P, the symmetric difference $M' = (M \cup P) \setminus (M \cap P)$ is a matching with $|M'| > |M|$.*

Interestingly, Berge showed that the converse is also true and this observation is a key for matching algorithms.

Theorem 3.8 ([Ber57])**.** *A matching M in a graph G is maximum if and only if G has no M-augmenting path.*

This theorem suggests the following straightforward algorithm for finding a maximum matching in a graph G. We start with any (heuristically chosen) matching M in G and we look for an M-augmenting path. If we find such a path P, then we set $M \leftarrow (M \cup P) \setminus (M \cap P)$ and we repeat the search for augmenting paths. If no M-augmenting path exists, then M is a maximum matching and we stop. Therefore, finding a maximum matching in G reduces to finding M-augmenting paths. If G has n vertices, this step can be repeated up to $\lfloor \frac{n}{2} \rfloor$ times, because at each step the cardinality of the matching is increased at least by one and a matching in G can be of size at most $\lfloor \frac{n}{2} \rfloor$. For general graphs the task of finding an M-augmenting path requires quite elaborate techniques, but for bipartite graphs it is rather easy, as we describe below.

Given a *bipartite* graph $G = (C, R; E)$ and a matching M in it, we can "grow" all M-augmenting paths simultaneously with breadth-first search. First, we observe that an augmenting path always has odd length, with the two endpoints being free vertices in the two partitions. Therefore, without loss of generality, we can start growing paths from the free C-vertices, trying to reach a free R-vertex via a sequence of alternating edges. In order to do that, we construct a directed version of G, denoted by G_d, (in fact, we only need to simulate the construction of G_d) as follows: All free edges are directed from C to R and all matched edges are directed from R to C. Then, the M-augmenting paths in G are exactly the directed paths from a free C-vertex to a free R-vertex in G_d. Therefore, an augmenting path can be found with breadth-first search in $\mathcal{O}(n^2)$ time, where n is the number of vertices in G. As already said, the augmentation step can be repeated up to $\lfloor \frac{n}{2} \rfloor$ times and, therefore, a maximum bipartite matching can be computed in $\mathcal{O}(n^3)$ time. Algorithm 2 and Theorem 3.11, which are adapted from [Wol98], formalize the above discussion. It is important to note that, along with a maximum matching, Algorithm 2 automatically computes a minimum vertex cover. We will use this fact in Section 4.3, in order to develop a cutting-plane approach for our integer linear program.

Definition 3.9. A vertex cover VC in a graph is a subset of vertices such that every edge of the graph is incident to at least one vertex in VC.

We note that in general graphs, for any vertex cover VC and for any matching M, it always holds that $|VC| \geq |M|$. Moreover, computing a minimum vertex cover is, in general, an NP-hard problem, while a maximum matching can be always computed in polynomial time. Theorem 3.11 states that in the case of bipartite graphs the two problems are equivalent and also gives an algorithmic proof of a strong min-max relation that originally goes back to a result obtained by König in 1931:

Theorem 3.10 ([Die05]). *In a bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.*

Hopcroft and Karp discovered an algorithm for maximum bipartite matching that runs in $\mathcal{O}\left(\sqrt{|V|}|E|\right)$ time, reducing the problem to a max-flow problem and then applying a max-flow algorithm [HK73]. This is also the asymptotically fastest solution known.

For the sake of completeness let us mention that finding a maximum matching in general graphs can be still done in polynomial time ($\mathcal{O}\left(|V|^4\right)$), although the lack of bipartite structure makes the task of finding augmenting paths far more difficult. The intricacy of the problem and its elegant solution were first pointed out by Edmonds in a seminal paper [Edm65].

Theorem 3.11. *Upon termination of Algorithm 2, M is a maximum matching and $C^- \cup R^+$ is a minimum vertex cover in G. Moreover, $|M| = |C^- \cup R^+|$.*

3.3 A non-algebraic view of identifiable graphs

Starting from Definition 3.5, we can ask if there are practical situations, other than `ZCAMF`, where identifiable bipartite graphs would be relevant. For example, let us consider the following scenario: C is a set of professors, R is a set of students and E models compatible research interests between professors and students. At the beginning of each semester, each professor must get at least one “compatible” student as teaching assistant and a student can be assigned to at most one professor at a time. However, the professors happen to be famous and influential and this results in a complication: If a professor leaves the university for a better position, he has the means to lure and take with him *all* students with whom he shares common interests. In such an unpleasant scenario, is it still possible to assign teaching assistants to the remaining professors?

If the graph $G = (C, R; E)$ of research compatibilities is identifiable, then this is possible, no matter which professor happens to leave, and all remaining professors will be happy with their assistants. But even then, we must not forget that research interests are dynamic in time. Therefore, the edge set of G may change at some point, such that the desirable property of identifiability is destroyed. How many changes in E can G tolerate before it loses the property? We study this *robustness* question in Chapter 6.

Algorithm 2 Computation of a maximum bipartite matching

```
1: **Input:** A bipartite graph $G = (C, R; E)$ and a matching $M \subseteq E$ in $G$.
2: **Output:** A maximum matching and a minimum vertex cover in $G$.
3:
4: All vertices of $G$ are not labeled and not scanned.
5: All $C$-vertices that are free w.r.t. $M$ get a dummy label \$.
6:
7: repeat
8:
9:    for all $c \in C$ that are labeled and not scanned do
10:      for all $r \in N(c)$ such that $(c, r)$ is free and $r$ is not labeled do
11:         label$[r] \leftarrow c$
12:      end for
13:      Mark $c$ as scanned.
14:   end for
15:
16:   for all $r \in R$ that are labeled and not scanned do
17:      if $r$ is free then
18:         An $M$-augmenting path $P$ was found.
19:         $P \leftarrow$ use the labels to backtrack from $r$ to \$.
20:         $M \leftarrow (M \cup P) \setminus (M \cap P)$
21:         goto line 4.
22:      else
23:         Find the edge $(c, r) \in M$ and do label$[c] \leftarrow r$
24:         Mark $r$ as scanned.
25:      end if
26:   end for
27:
28: until there is no vertex in $G$ that is labeled and not scanned
29:
30: $C^+$, $R^+ \leftarrow$ vertices of $C$, $R$ that are labeled.
31: $C^-$, $R^- \leftarrow$ vertices of $C$, $R$ that are not labeled.
32: return maximum matching $M$, minimum vertex cover $C^- \cup R^+$.
```

At the end of each academic year, some students graduate and leave the university and this can destroy the identifiability of G. On the other hand, a certain number of new students will be admitted to join in. After having interviewed and ranked the applicants, the selection committee faces the following problem: Select a subset of the applicants that are ranked as high as possible and additionally restore the identifiability of G. This is, in fact, a generalization of the `MINSENSOR` problem that we define and study in Chapter 4.

Clearly, this example from university life is rather artificial. However, it would be interesting to investigate whether identifiable bipartite graphs have, as combinatorial objects, more realistic applications, e.g., in scheduling. They could be used to model the following constraint: If any C-vertex disappears together with its neighborhood, all remaining C-vertices must be matchable. This also motivates us to ask: Does it make sense to define this property for general graphs?

3.4 Some combinatorial properties

In this section we list some properties of identifiable graphs; we gained them as a by-product of our attempts to understand the combinatorial structure of these graphs. In some of our arguments throughout the section, we will use the concepts of surplus and tight subsets, which are generally important for the study of bipartite matchings. Both concepts can be found, e.g., in the book of Lovász and Plummer [LP86].

Definition 3.12. Let $G = (C, R; E)$ be a bipartite graph. For a set $X \subseteq C$, the *surplus* of X is defined as $\sigma(X) = |N(X)| - |X|$. The surplus of the whole graph G is $\sigma(G) = \min_{\emptyset \neq X \subseteq C}\{\sigma(X)\}$. A set $X \subseteq C$ is called *tight*, if $X \neq \emptyset$ and $\sigma(X) = \sigma(G)$.

Intuitively, the surplus of a set $X \subseteq C$ quantifies how much X satisfies Hall's condition $|N(X)| \geq |X|$ and tight sets are the ones that minimally satisfy (or maximally violate) this condition. Rephrasing Hall's theorem (Theorem 3.6), G has a C-perfect matching if and only if $\sigma(G) \geq 0$. As we will see in Chapter 6, the computation of the surplus will be the key for determining the robustness of identifiability. Moreover, in Section 4.3, we will see that finding a tight set in a graph that violates Hall's condition will be the basis for an efficient cutting-plane approach for the ILP of `MINSENSOR`. It turns out that the surplus and a tight set of a graph can be efficiently computed; see Algorithms 5 and 6.

Below we summarize some properties of identifiable graphs. With $d(x)$ we denote the degree of a vertex x.

Proposition 3.13. *Let $G = (C, R; E)$ be a bipartite graph with at least one edge.*

(i) G is identifiable if and only if its connected components are identifiable.

(ii) If there exists a vertex $x \in C$ that is connected and identifiable, then G has a C-perfect matching. In particular, this is the case if G has two identifiable vertices $x, y \in C$.

(iii) Suppose that G is identifiable. If $X \subseteq C$ is a tight subset, then $d(x) = \sigma(G) + 1$, for all $x \in X$ (i.e., for all $x \in X$, $\{x\}$ is also a tight subset of G). Moreover, $\sigma(G) = \min_{x \in C} d(x) - 1$.

(iv) If G is identifiable, then its connected components are either single edges or have positive surplus.

(v) If there is a subset $R' \subseteq R$ such that the graph induced by $C \cup R'$ is identifiable, then G is identifiable too.

(vi) We call a vertex $z \in R$ that is adjacent to all vertices of C a dominating R-vertex. The addition or removal of such a vertex preserves identifiability: G is identifiable if and only if $G + z$ is identifiable.

Proof.

(i) This easily follows from the fact that all neighbors of a vertex $x \in C$ are in the same connected component as x.

(ii) Vertex x has at least one neighbor $z \in R$. Because x is identifiable, there exists a matching M_x that does not touch z and that covers all vertices of $C \setminus \{x\}$. Therefore, $M_x \cup \{(x, z)\}$ is a matching that covers all vertices of C. If y is identifiable, then x must be matchable in $\widetilde{G}_y$ and, therefore, it cannot be isolated.

(iii) From the definition of $\sigma(G)$, $d(x) \geq \sigma(G) + 1$, for all $x \in C$. Let $X \subseteq C$ be a tight subset, i.e., $|N(X)| = |X| + \sigma(G)$. If $|X| = 1$ the first part is immediate, so assume $|X| \geq 2$. For all $x \in X$, it must be $d(x) \leq \sigma(G) + 1$, because otherwise it would be $|N(X) \setminus N(x)| < |X| - 1$ and not all vertices of $X \setminus \{x\}$ would be matchable in $\widetilde{G}_x$. Therefore, $d(x) = \sigma(G) + 1$, for all $x \in X$. Now, we have $d(x) - 1 = \sigma(\{x\}) = \sigma(G) \leq \sigma(\{y\}) = d(y) - 1$ for any $y \in C$ and the equality $\sigma(G) = \min_{x \in C} d(x) - 1$ follows.

(iv) By (i), it is enough to prove that a connected identifiable graph of zero surplus must be a single edge. Let G be a connected identifiable graph with $\sigma(G) = 0$. By (iii), there is a vertex $x \in C$ of degree 1. But then necessarily $|C| = 1$ since otherwise there would exist a vertex $y \in C \setminus \{x\}$ adjacent to the unique neighbor of x, which would mean that x is isolated in $\widetilde{G}_y$, contrary to the identifiability assumption.

(v) The addition of an R-vertex can only increase the matching numbers of the subgraphs $\widetilde{G}_x$, for all $x \in C$.

(vi) The addition of a dominating R-vertex to G increases $\sigma(G)$ by one, but does not change $\widetilde{G}_x$, for all $x \in C$.

□

In summary, in the study of identifiable graphs we can safely focus on identifiable graphs that are connected, have strictly positive surplus and contain no dominating R-vertices.

Assertion (ii), in conjunction with Hall's marriage theorem [Hal35], shows that identifiability implies non-negative surplus. There is also a connection between this observation and Lemma 3.1. Namely, if a source-sensor network is identifiable, then the corresponding zero pattern $\mathcal{Z}$ has full-column structural rank. This is a necessary condition for the requirement of Lemma 3.1 that A must have full column-rank.

Assertion (iii) shows that in identifiable graphs, the computation of the surplus is easier than in general graphs (cf. Section 6.2). We remark that in general, not every subset $X \subseteq \{x \in C : d(x) = \sigma(G) + 1\}$ in an identifiable graph is tight, and that the equality $\sigma(G) = \min_{x \in C} d(x) - 1$ is only a necessary, but not a sufficient condition for identifiability. For example, let us consider $C = \{c_1, c_2\}$, $R = \{r_1, r_2, r_3\}$ and $E = \{(c_1, r_1), (c_1, r_2), (c_2, r_2), (c_1, r_3), (c_2, r_3)\}$. Then, $\sigma(G) = 1 = d(c_2) - 1$, but G is not identifiable.

Assertion (iv) implies that a graph $G = (C, R; E)$ with $|R| = |C|$ is identifiable if and only if it consists of single edges. Assertion (v) motivates the following question: Can we characterize minimally identifiable graphs, that is, identifiable graphs $G = (C, R; E)$ such that the deletion of an arbitrary vertex from R results in a non-identifiable graph?

The fact that G is identifiable and $d(x) = \sigma(G) + 1$, for all $x \in X \subseteq C$, does not imply that X is tight. For example, let us consider G to be a path with $|C| = 3$ and $|R| = 4$. G is identifiable, $\sigma(G) = 1$ and for the subset $X = \{c_1, c_3\}$, where c_1 and c_3 are the two vertices in C that are of distance 4 in G, we have $d(c_1) = d(c_3) = \sigma(G) + 1$. However, X is not tight, because $|N(X)| = 4 > |X| + \sigma(G)$.

Proposition 3.14. *A connected bipartite graph $G = (C, R; E)$ with $|R| = |C| + 1$ is identifiable if and only if $d(x) = 2$ for all $x \in C$.*

Proof.
Forward: It has to be $d(x) \leq 2$, for all $x \in C$, otherwise $\widetilde{G}_x$ would have fewer than $|C| - 1$ R-vertices and this contradicts the identifiability assumption. Let us assume that there exists $x \in C$ with $d(x) = 1$ and call v the single neighbor of x. Since G is connected, v must have also a neighbor $y \neq x$. In that case x is isolated in $\widetilde{G}_y$, which contradicts the identifiability assumption. Therefore, $d(x) = 2$, for all $x \in C$.
Reverse: $|V(G)| = 2|C| + 1$, $|E(G)| = 2|C|$ and, since G is connected, it must be a tree. Let us assume that G is not identifiable. Then there exists a vertex $x \in C$ and a subset $Y \subset C$, such that $x \notin Y$ and $|Y| \geq |\widetilde{N}_x(Y)| + 1$. Let us consider the subgraph G' induced by $\{x\} \cup Y \cup N(x) \cup \widetilde{N}_x(Y)$: we have $|V(G')| = 1 + |Y| + 2 + |\widetilde{N}_x(Y)|$ and $|E(G')| = 2(|Y| + 1)$. We observe that $|E(G')| \geq |V(G')|$; therefore, G' contains a cycle and this contradicts the fact that G is a tree. Therefore, G is identifiable. □

3.5 Partially identifiable graphs

In Section 3.1 we defined identifiable graphs starting from Proposition 3.1 and the condition $\widehat{A} = AR \lhd \mathcal{Z}$. In particular, we observed that this condition, combined with the identifiability of $\mathcal{Z}$ (i.e. of the underlying source-sensor network) can force R to be diagonal, provided that A and X are generic. In this section, we turn our attention to the next natural question: Can we draw any conclusions about source-sensor networks that fail to be identifiable, just by studying their structure? Below we show that this is indeed the case; this gives rise to the notion of *partially identifiable graphs*. The discussion that follows is meant to be complemented by the example of Figure 3.3.

As in Section 3.1, our reasoning starts again from the basic condition $\widehat{A} = AR \lhd \mathcal{Z}$, which can be expanded to m homogeneous linear systems, one for each column of R. Let us consider the j-th column and the corresponding system $\widetilde{A}_j\widetilde{r}_j = 0$. We know that if $\widetilde{A}_j$ has full column structural rank and generic entries, then $\widetilde{r}_j = 0$, i.e., the j-th column of R is all zero apart from the diagonal element. On the other hand, if $\widetilde{A}_j$ is structurally rank-degenerate, it follows that, in general, $\widetilde{r}_j \neq 0$. Actually, it turns out that we can draw much more specific conclusions, by studying the solution of the system $\widetilde{A}_j\widetilde{r}_j = 0$ from a structural point of view. In this way, we can predict the sparsity pattern of the vector $\widetilde{r}_j$ and doing that for all columns j we can finally predict the zero pattern of R.

The theoretical machinery that allows us to do so is a canonical decomposition of bipartite graphs that was discovered by Dulmage and Mendelsohn (*DM decomposition*) [DM58]. For the moment, we will use as a black-box one crucial aspect of the DM decomposition that is relevant for our purpose; details about the internals of the decomposition will follow in Section 3.5.1. Namely, from this decomposition we can determine a permutation of the rows and columns of the zero pattern $\widetilde{\mathcal{Z}}_j$, such that the permuted matrix is block-triangular and exactly one of the blocks has reduced structural rank (see Figure 3.3).

From the block-triangular form of $\widetilde{\mathcal{Z}}_j$, and under the usual genericity assumption on the entries of $\widetilde{A}_j$, we can immediately read off which components of $\widetilde{r}_j$ will be zero. Applying this procedure for all columns of $\mathcal{Z}$, we can finally predict the zero pattern of R. Let us recall that according to Proposition 3.1 we have $\widehat{A} = AR$ and $\widehat{X} = R^{-1}X$, i.e., R and R^{-1} account for the non-identifiability of A and X, respectively. Therefore, we are motivated to ask the following natural question: From the zero pattern of R, can we draw any conclusions about the zero pattern of R^{-1}? It turns out that the theory of sparse matrix algorithms provides us with the tools to answer this question.

In general, a sparse matrix algorithm takes advantage of the zero patterns of the matrices involved in the computation. Such an algorithm typically has a first phase, in which it predicts the zero pattern of the solution from the zero pattern of the input.

Then, it performs the actual numerical computations in a static data structure. This is particularly useful in applications where many problem instances with the same zero pattern must be solved and, therefore, the prediction of the zero pattern of the result needs to be done only once. Graph theory is a useful language in which to state and prove structure prediction results. One reason for this is that the structural effect of a matrix computation often depends on some path structure, which is easier to describe in terms of graphs than in terms of matrices. For a good review of this exciting field that lies on the intersection of numerical computation and graph theory, we refer to [Gil94]. From this paper we also take Definition 3.15 and Proposition 3.17 (see also Figure 3.4).

Definition 3.15. Let $\mathcal{W}$ be a *square* $n \times n$ zero pattern with *non-zero* diagonal. We model $\mathcal{W}$ with the *directed* graph $G(\mathcal{W}) = (V, E)$, where $V = \{1, \dots, n\}$ and $E = \{(i, j) \in V^2 : i \neq j \text{ and } \mathcal{W}_{ij} = 1\}$.

Definition 3.16. Given a graph $G = (V, E)$, its *transitive closure* is the graph $G^* = (V, E^*)$, where $E^* = \{(i, j) \in V^2 : \text{there is a path in G from vertex } i \text{ to vertex } j\}$.

Proposition 3.17. *If a square matrix R satisfies a zero pattern $\mathcal{W}$, then R^{-1} satisfies the zero pattern that corresponds to the transitive closure of $G(\mathcal{W})$.*

One simple way to compute the transitive closure of a directed graph in $\Theta(|V|^3)$ time is to assign a unit weight to each edge and then compute all-pairs shortest paths by applying the Floyd-Warshall algorithm. If there is a path from vertex i to vertex j, then we get $d_{ij} < n$, otherwise we get $d_{ij} = \infty$. Of course, more elaborate methods are also known [Nuu95].

Returning to the example of Figure 3.3, we now use Proposition 3.17 to predict the zero pattern of R^{-1}. We observe that the directed graph that models the zero pattern of R is *already transitive* and, therefore, R^{-1} satisfies the same zero pattern as R. We strongly suspect that this is not a coincidence. In fact, the methodology described above has been implemented and applied on many random instances and the following conjecture always holds. However, a formal proof remains a topic for further research.

Conjecture. *The zero pattern of R as it is determined from the condition $AR \lhd \mathcal{Z}$ is always transitive.*

In Figure 3.3 we observe that the first two rows of R^{-1} are all zero, apart from the diagonal elements. From the equation $\widehat{X} = R^{-1}X$, it follows that the first two rows of X and $\widehat{X}$ differ only by diagonal scaling. Therefore, although the graph is not identifiable according to Definition 3.5 (actually there is not even a single identifiable column), the signals of the first two sources can still be uniquely reconstructed up to scaling. This observation motivates us to describe the network as *partially identifiable.* An interesting direction for further research would be to look for a combinatorial characterization of such networks.

We point out that the whole analysis done so far was based on purely graph-theoretic arguments and no numerical computation at all. The reason is that we want to draw our conclusions based solely on the network structure and the assumption that the (A, X) solution of `ZCAMF` is numerically generic. In practice, when we solve an instance of `ZCAMF` we get concrete numerical values for the matrices A and X. Then, at first we can (and we should) check if the genericity assumptions on A and X hold. Going one step further we can compute R and its inverse parametrically. This is done by determining the general solution of the linear system $A_j r_j = 0$, for each column $j = 1, \dots, m$.

This solution is $r_j = K_j u_j$, where K_j is a matrix whose columns form a basis for the nullspace of A_j and u_j is a vector of free parameters, as many as the dimension of the nullspace of A_j. We recall that the nullspace of an $n \times m$ matrix A is defined as the linear subspace $\mathcal{N}(A) = \{x \in \mathbb{R}^m : Ax = 0\}$. In the special case where the j-th column is identifiable (i.e. $\operatorname{rank}(A_j) = m - 1$), K_j has as only column the unit vector e_j and u_j is a scalar variable. We can compute the matrix K_j from the singular value decomposition $A_j = U \Sigma V^T$ (see Theorem 2.1). In particular, if A_j has dimensions $N \times M$ and $\operatorname{rank}(A_j) = r$, then the singular vectors $\{v_{r+1}, \dots, v_M\}$ form an orthonormal basis for the nullspace of A_j [GL96]. Finally, the entries of R will be linear functions of the free parameters u_j, as shown below.

$$R = \begin{pmatrix} K_1 & \dots & K_m \end{pmatrix} \begin{pmatrix} u_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & u_m \end{pmatrix}$$

In Figure 3.5 we give an example, where R has been computed in parametric form. Then, it is possible to also compute R^{-1} in parametric form with a computer algebra system; the entries of R^{-1} will be rational functions of the free parameters.

Topic for further research. It would be interesting to consider if these closed form expressions for R and R^{-1} can be subsequently used to draw any more conclusions about the space of feasible solutions for the `ZCAMF` problem. For example, let us consider the case where, additionally to the zero pattern $\mathcal{Z}$, we also have linear constraints $\mathcal{C}_A$ on the entries of A. Then, the condition $AR \lhd \mathcal{C}_A$ gives rise to a set of linear constraints on the free u-parameters, which can be used to further quantify the "difference" between $\widehat{A}$ and A. Note that the notion of "difference" must be properly defined such that it takes into account the unavoidable diagonal scaling.

3.5.1 The mechanics of Dulmage-Mendelsohn decomposition

For the sake of completeness, in this section we give some more details about the mechanics of the DM decomposition that was introduced in [DM58] as a canonical decomposition for bipartite graphs.

Theorem 3.18 (e.g. [LP86])**.** *Let $G = (C, R; E)$ be a bipartite graph. There exists a unique partitioning $C = C_1 \cup C_2 \cup C_3$ and $R = R_1 \cup R_2 \cup R_3$, that satisfies the properties stated below:*

1. *$N(C_3) = R_3$ and $N(R_1) = C_1$.*
2. *The subgraph $G[C_2, R_2]$ has a perfect matching.*
3. *Every maximum matching consists of a perfect matching between C_2 and R_2, a matching of all vertices of R_3 onto C_3 and a matching of all vertices of C_1 onto R_1.*
4. *The subgraphs $G[R_3, C_3]$ and $G[C_1, R_1]$ have positive surpluses.*
5. *For every minimum vertex cover T: $C_1 \cup R_3 \subseteq T \subseteq C_1 \cup R_3 \cup C_2 \cup R_2$. Moreover, $C_1 \cup R_3 \cup C_2$ and $C_1 \cup R_3 \cup R_2$ are minimum vertex covers.*

The DM decomposition has some immediate applications in sparse matrix computations. We already know that we can model a zero pattern as a bipartite graph, where the two partitions correspond to the rows and columns of the matrix and the edges correspond to the non-zero positions. The DM decomposition of this graph gives a row and column permutation for the zero pattern that makes it block-triangular and also separates the part that has reduced structural rank from the rest. This is illustrated in Figure 3.6.

As far as its algorithmic aspects are concerned, the DM decomposition can be efficiently computed by first finding any maximum matching in the bipartite graph G and then performing a depth-first search. Thereby, we are looking for alternating paths with respect to M, i.e., paths whose edges are matched and free in alternation [2]. The next theorem states exactly how the DM decomposition is computed. For its proof, more algorithmic details and applications in sparse matrix computations we refer to [PF90].

Theorem 3.19. *Let M be a maximum matching in G. With respect to M, the sets of the DM decomposition as they are defined in Theorem 3.18, are computed as follows:*

- *R_1 := R-vertices that can be reached through an alternating path from an unmatched R-vertex.*

[2]Compare to augmenting paths from Section 3.2. In fact, an augmenting path is a special case of alternating path.

- *R_3 := R-vertices that can be reached through an alternating path from an unmatched C-vertex.*
- *$R_2 = R \setminus (R_1 \cup R_3)$*
- *C_1 := C-vertices that can be reached through an alternating path from an unmatched R-vertex.*
- *C_3 := C-vertices that can be reached through an alternating path from an unmatched C-vertex.*
- *$C_2 = C \setminus (C_1 \cup C_3)$*

The sets R_1, R_2, R_3 and C_1, C_2, C_3 do not depend on the choice of the maximum matching M.

For a given zero pattern $\mathcal{Z}$ and a generic matrix A, such that $A \lhd \mathcal{Z}$, we write down $AR \lhd \mathcal{Z}$.

$$\overbrace{\begin{pmatrix} a_{11} & \mathbf{0} & \mathbf{0} & a_{14} & \mathbf{0} & \mathbf{0} & a_{17} & a_{18} \\ a_{21} & \mathbf{0} & \mathbf{0} & a_{24} & \mathbf{0} & a_{26} & \mathbf{0} & \mathbf{0} \\ a_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & a_{42} & \mathbf{0} & \mathbf{0} & a_{45} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{55} & \mathbf{0} & \mathbf{0} & a_{58} \\ \mathbf{0} & a_{62} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ a_{71} & a_{72} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & a_{83} & a_{84} & \mathbf{0} & a_{86} & \mathbf{0} & a_{88} \end{pmatrix}}^{A} \overbrace{\begin{pmatrix} r_{11} & \cdots & r_{18} \\ \vdots & \ddots & \vdots \\ r_{81} & \cdots & r_{88} \end{pmatrix}}^{R} \lhd \overbrace{\begin{pmatrix} 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & 1 \\ 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & 1 & \mathbf{0} & 1 & \mathbf{0} & 1 \end{pmatrix}}^{\mathcal{Z}}$$

To show the use of DM decomposition, we take, for example, the 7-th column and consider the corresponding linear system $\widetilde{A}_7 \widetilde{r}_7 = 0$.

$$\overbrace{\begin{pmatrix} a_{21} & \mathbf{0} & \mathbf{0} & a_{24} & \mathbf{0} & a_{26} & \mathbf{0} \\ a_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & a_{42} & \mathbf{0} & \mathbf{0} & a_{45} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{55} & \mathbf{0} & a_{58} \\ \mathbf{0} & a_{62} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ a_{71} & a_{72} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & a_{83} & a_{84} & \mathbf{0} & a_{86} & a_{88} \end{pmatrix}}^{\widetilde{A}_7} \overbrace{\begin{pmatrix} r_{17} \\ r_{27} \\ r_{37} \\ r_{47} \\ r_{57} \\ r_{67} \\ r_{87} \end{pmatrix}}^{\widetilde{r}_7} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

We can permute the rows and columns of $\widetilde{A}_7$ such that the resulting matrix is block-triangular. In this example we apply the row permutation $\pi_r = (7, 1, 4, 3, 5, 6, 2)$ and the column permutation $\pi_c = (6, 3, 4, 7, 5, 2, 1)$. The submatrix $[R_3, C_3]$ is horizontal (i.e. more columns than rows), the submatrix $[R_2, C_2]$ is square and the submatrix $[R_1, C_1]$ is vertical. $[R_3, C_3]$ has reduced structural rank, while $[R_2, C_2]$ and $[R_1, C_1]$ both have full structural ranks.

$$\left(\begin{array}{c|ccc|cc|cc} & & C_3 & & C_2 & & C_1 & \\ \hline R_3 & a_{86} & a_{83} & a_{84} & a_{88} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & a_{26} & \mathbf{0} & a_{24} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{21} \\ \hline R_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{58} & a_{55} & \mathbf{0} & \mathbf{0} \\ & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{45} & a_{42} & \mathbf{0} \\ \hline & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{62} & \mathbf{0} \\ R_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{72} & a_{71} \\ & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{31} \end{array}\right) \begin{pmatrix} r_{67} \\ r_{37} \\ r_{47} \\ r_{87} \\ r_{57} \\ r_{27} \\ r_{17} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

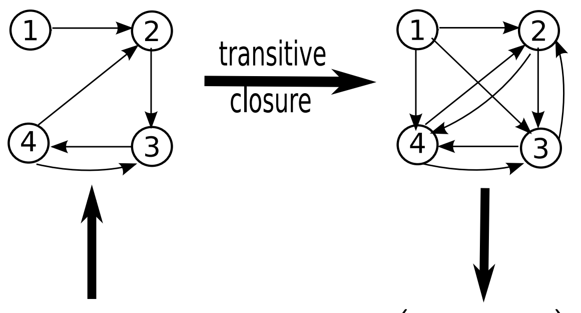
If the a_{ij} entries are generic, then the r_{ij} entries that correspond to the square and vertical parts will be zero. The r_{ij} entries that correspond to the horizontal part will be coupled to each other with a set of linear equalities and, in general, will be nonzero.

$$\begin{pmatrix} r_{87} \\ r_{57} \\ r_{27} \\ r_{17} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \quad , \quad \begin{pmatrix} a_{86} & a_{83} & a_{84} \\ a_{26} & \mathbf{0} & a_{24} \end{pmatrix} \begin{pmatrix} r_{67} \\ r_{37} \\ r_{47} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

So, we can predict the zero pattern of the 7-th column of R. Doing the same for all columns gives us the zero pattern of R. We observe that the corresponding digraph is transitive, so R^{-1} also satisfies the same zero pattern.

$$\begin{pmatrix} r_{17} \\ r_{27} \\ r_{37} \\ r_{47} \\ r_{57} \\ r_{67} \\ r_{77} \\ r_{87} \end{pmatrix} \lhd \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \\ 1 \\ \mathbf{0} \\ 1 \\ 1 \\ \mathbf{0} \end{pmatrix} \quad , \quad R \lhd \begin{pmatrix} 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 \end{pmatrix}$$

Figure 3.3: An example of a non-identifiable zero pattern $\mathcal{Z}$ and how the DM decomposition can be used to predict the sparsity pattern of R.



$$\mathcal{W} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad \mathcal{W}^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Figure 3.4: From the zero pattern of a matrix we can predict the zero pattern of its inverse. Here, $\mathcal{W}^{-1}$ does not denote numerical inverse, but the zero pattern of the inverse.

$$\overbrace{\begin{pmatrix} 6 & \mathbf{0} & \mathbf{0} & 5 & \mathbf{0} & \mathbf{0} & 8 & 7 \\ 6 & \mathbf{0} & \mathbf{0} & 2 & \mathbf{0} & 2 & \mathbf{0} & \mathbf{0} \\ 4 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 7 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 4 & \mathbf{0} & \mathbf{0} & 2 \\ \mathbf{0} & 7 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 7 & 2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 7 & 9 & \mathbf{0} & 2 & \mathbf{0} & 5 \end{pmatrix}}^{A} \overbrace{\begin{pmatrix} r_{11} & \cdots & r_{18} \\ \vdots & \ddots & \vdots \\ r_{81} & \cdots & r_{88} \end{pmatrix}}^{R} \lhd \overbrace{\begin{pmatrix} 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & 1 \\ 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & 1 & \mathbf{0} & 1 & \mathbf{0} & 1 \end{pmatrix}}^{\mathcal{Z}}$$

$$\Longrightarrow R = \begin{pmatrix} d_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{2}{7}p_{12} - \frac{9}{7}p_{13} & \frac{24}{35}p_{21} + \frac{8}{5}p_{22} & d_3 & p_{41} & \frac{24}{35}p_{51} + \frac{8}{5}p_{52} & p_{61} & p_{71} & p_{82} \\ p_{13} & -\frac{7}{5}p_{21} - \frac{8}{5}p_{22} & -\frac{8}{5}p_{31} & d_4 & -\frac{7}{5}p_{51} - \frac{8}{5}p_{52} & -\frac{8}{5}p_{62} & -p_{71} & -p_{81} \\ 0 & -\frac{1}{2}p_{21} & 0 & 0 & d_5 & 0 & 0 & 0 \\ p_{12} & \frac{7}{5}p_{21} + \frac{8}{5}p_{22} & \frac{8}{5}p_{31} & p_{43} & \frac{7}{5}p_{51} + \frac{8}{5}p_{52} & d_6 & p_{71} & p_{81} \\ p_{11} & p_{22} & p_{31} & p_{42} & p_{52} & p_{62} & d_7 & p_{83} \\ 0 & p_{21} & 0 & 0 & p_{51} & 0 & 0 & d_8 \end{pmatrix}$$

Figure 3.5: For concrete numerical values for A and from the condition $AR \lhd \mathcal{Z}$ we can compute R in parametric form. The variables d_i and p_{ij} are free parameters. Compare with Figure 3.3: There we have computed the zero pattern of R before we get concrete numerical values for A.

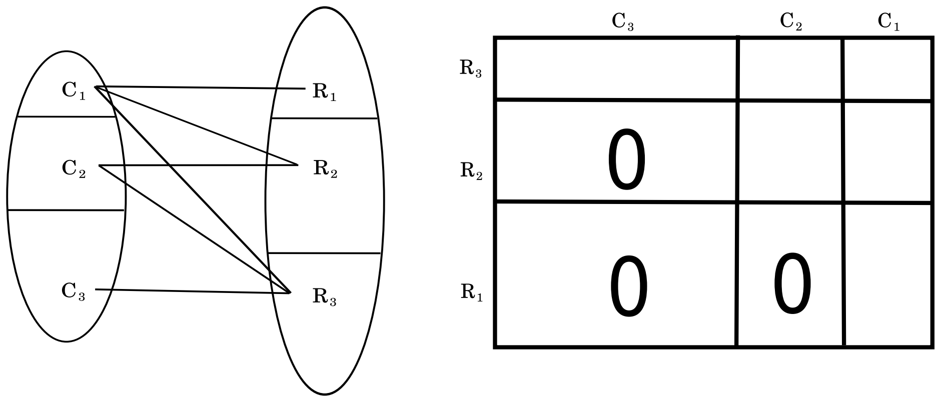


Figure 3.6: A schematic view of Dulmage-Mendelsohn decomposition. For the sizes of the partitions we have $|R_1| \geq |C_1| + 1$, $|C_2| = |R_2|$ and $|C_3| \geq |R_3| + 1$. The subgraph $G[C_1, R_1]$ has a matching that covers all C_1, i.e., the submatrix $[C_1, R_1]$ has full column-rank (structural rank). The subgraph $G[C_2, R_2]$ has a perfect matching, i.e., the submatrix $[C_2, R_2]$ has full rank. The subgraph $G[C_3, R_3]$ has a matching that covers all R_3, i.e., the submatrix $[C_3, R_3]$ has full row-rank but reduced column-rank.

Chapter 4

Optimal Manufacturing of Sensors

Summary of the chapter. We define the combinatorial problem `MINSENSOR` that arises in the context of source-sensor networks, when we need to manufacture sensors in an optimal way. We prove that this problem is NP-hard with a reduction from `HITTING SET`. Then, for its exact solution we develop a mixed integer linear program (MILP), where we use the connection of bipartite matchings to linear programming via totally unimodular matrices. For a practical solution of the MILP, we propose two cutting-plane approaches that take advantage of the fact that a maximum matching and a minimum vertex cover can be efficiently computed in bipartite graphs. We also present a greedy algorithm for `MINSENSOR` and show that it guarantees a logarithmic approximation ratio. We do that by showing that `MINSENSOR` is a special case of `SUBMODULAR SET COVER`. Finally, we experimentally test the different approaches on simulated problem instances of various sizes.

Assume that we want to manufacture a set of sensors, in order to monitor a set of signal sources C. If the design of sensors can be done at low cost, we can proceed by designing a large (and maybe redundant) list of candidate sensors R; this gives rise to a bipartite graph $G = (C, R; E)$. However, when it comes to the actual manufacturing of the sensors, which is usually more expensive than the design, we would like to select a *cheap subset* of sensors $I \subseteq R$. Moreover, we would like to select I appropriately such that the induced subgraph of G is identifiable; this will allow us to process the measurements in the `ZCAMF` framework, without worrying (too much) about identifiability issues. Although it is easy to check whether G is identifiable, it turns out that the task of selecting identifiable subgraphs is a hard one.

Problem 4. `MINSENSOR`
Given is a bipartite graph $G = (C, R; E)$ and a function c that assigns a nonnegative rational cost to each vertex of R. We want to find a set $I \subseteq R$ such that the subgraph $G[C, I]$ is identifiable and $\sum_{y \in I} c(y)$ is minimized.

4.1 MINSENSOR is NP-hard

We prove that the unit-cost restriction of MINSENSOR is NP-hard with a reduction from HITTING SET [1] [GJ79].

Problem 5. HITTING SET
Given is a finite ground set T, a family $\mathcal{F}$ of subsets of T and a positive integer $k \leq |T|$. Is there a subset $S \subseteq T$, with $|S| \leq k$, that intersects all sets of $\mathcal{F}$?

For the ease of presentation, we use the representation of a bipartite graph as a zero pattern. More specifically, we model a bipartite graph $G = (A, B; E)$ with the zero pattern in which the columns correspond to part A, the rows correspond to part B, and the ones correspond to the edges in E. The "translation" between graph and zero pattern terminology is then straightforward. When convenient, we shall identify the zero pattern with the graph it represents.

Theorem 4.1. *MINSENSOR is NP-hard.*

Proof. Let $(\mathcal{F}, T, k)$ be an arbitrary instance of HITTING SET with $\mathcal{F} = \{C_1, \ldots, C_m\}$; without loss of generality we assume that $T = \{1, \ldots, n\}$. First, we construct an $n \times m$ zero pattern $\mathcal{H}$ such that $\mathcal{H}_{ij} = 0$ if and only if $i \in C_j$. Then, HITTING SET is equivalent to asking: Is there a set of at most k rows that cover at least one zero in each column of $\mathcal{H}$? We can answer this question by solving an appropriately constructed instance of the decision version of the unit-cost restriction of MINSENSOR.

We construct a zero pattern $\mathcal{G}$ as follows: $\mathcal{G}$ has in total $2m$ columns, which we label as $x_1, \ldots, x_m; y_1, \ldots, y_m$, and $2m + n + 1$ rows divided in 4 blocks: $\mathcal{A}, \mathcal{B}_1, \mathcal{B}_2, \mathcal{C}$. Block $\mathcal{A}$ consists of two $n \times m$ sub-blocks: $\mathcal{A}_1 = \mathcal{H}$ and $\mathcal{A}_2 = \mathbf{1}$. Each of the blocks $\mathcal{B}_1$ and $\mathcal{B}_2$ is an $m \times m$ identity matrix. Finally, block $\mathcal{C}$ consists of a single row with its first m elements equal to one and the last m equal to zero. An example for this construction is illustrated in Figure 4.1. For notational convenience we set $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_m\}$ and denote by L the subset of rows from the blocks $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{C}$ (note that $|L| = 2m + 1$).

We now show that $\mathcal{H}$ contains a set of rows S, with $|S| \leq k$, that covers each column with at least one zero, if and only if $\mathcal{G}$ contains a set of rows I, with $|I| \leq 2m + 1 + k$, such that $\mathcal{G}[X \cup Y, I]$ is identifiable.

Forward: We take $I = S \cup L$ and show that $\mathcal{G}[X \cup Y, I]$ is identifiable. For each column x_j, there exists a row $s_{x_j} \in S$ such that $\mathcal{G}[s_{x_j}, x_j] = 0$. For column x_j we consider the subpattern $\widetilde{\mathcal{G}}_{x_j}$: in $\widetilde{\mathcal{G}}_{x_j}$ all columns, except for y_j, can be matched to the rows of blocks $\mathcal{B}_1$ and $\mathcal{B}_2$, and column y_j is matched to row s_{x_j}. For column y_j, we

[1]The hardness of HITTING SET immediately follows from the hardness of VERTEX COVER. In fact, HITTING SET is a generalization of VERTEX COVER to hypergraphs.

$$\begin{array}{rcl} T & = & \{1,\dots,6\} \\ \mathcal{F} & = & \left\{\begin{array}{c}\{2,4,5\}\\ \{2,3,4,6\}\\ \{1,6\}\end{array}\right\} \end{array} \implies \mathcal{H} = \begin{pmatrix} 1 & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \\ 1 & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & 1 & 1 \\ 1 & \mathbf{0} & \mathbf{0} \end{pmatrix} \implies \mathcal{G} = \left(\begin{array}{c|ccc|ccc} & x_1 & x_2 & x_3 & y_1 & y_2 & y_3 \\ \hline & 1 & 1 & \mathbf{0} & 1 & 1 & 1 \\ & \mathbf{0} & \mathbf{0} & 1 & 1 & 1 & 1 \\ \mathcal{A} & 1 & \mathbf{0} & 1 & 1 & 1 & 1 \\ & \mathbf{0} & \mathbf{0} & 1 & 1 & 1 & 1 \\ & \mathbf{0} & 1 & 1 & 1 & 1 & 1 \\ & 1 & \mathbf{0} & \mathbf{0} & 1 & 1 & 1 \\ \hline\hline & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ \mathcal{B}_1 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} \\ & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 \\ \hline & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ \mathcal{B}_2 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} \\ & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & 1 \\ \hline\hline \mathcal{C} & 1 & 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right)$$

Figure 4.1: An example for the reduction of `HITTING SET` to `MINSENSOR`.

consider the subpattern $\widetilde{\mathcal{G}}_{y_j}$: in $\widetilde{\mathcal{G}}_{y_j}$ all columns, except for x_j, can be matched to the rows of blocks $\mathcal{B}_1$ and $\mathcal{B}_2$, and column x_j is matched to the row $\mathcal{C}$.

Reverse: The identifiability of $\mathcal{G}[X \cup Y, I]$ implies that each column of $\mathcal{G}[X \cup Y, I]$ contains at least $2m-1$ zeros. In order for all columns from Y to contain $2m-1$ zeros, it must be $I \supseteq L$. Therefore, $I = L \cup S$, where $|S| \leq k$. But each column of X has exactly $2m-2$ zeros in $\mathcal{G}[X, L]$. Therefore, the set of rows S must cover each column from X with at least one zero.

So, we conclude that `HITTING SET` can be polynomially reduced to the decision version of the unit-cost restriction of `MINSENSOR`. Therefore, `MINSENSOR` is NP-hard. □

Remark 4.2. The decision version of `MINSENSOR` is NP-complete, because a-posteriori checking if a graph is identifiable can be done in polynomial time.

We did our reduction from `HITTING SET` and, therefore, this classical problem will be a valuable source of information. We should be aware that `HITTING SET` often appears under the disguise of the equivalent problem `SET COVER`.

Problem 6. `SET COVER`
Given is a finite ground set T, a family $\mathcal{F}$ of subsets of T and a positive integer $k \leq |\mathcal{F}|$. Is there a subfamily $\mathcal{S}$ of $\mathcal{F}$, with $|\mathcal{S}| \leq k$, such that every element of the ground set T is contained in some set of $\mathcal{S}$?

To see the equivalence between `HITTING SET` and `SET COVER`, observe that the sets of the family and the elements of the ground set have dual roles in the two problems. In `HITTING SET` we choose elements from the ground set in order to hit all sets of the family, while in `SET COVER` we choose sets from the family in order to cover all elements from the ground set.

4.2 An exact MILP solution for `MINSENSOR`

The NP-hardness of `MINSENSOR` does not leave us much hope for the existence of an algorithm that runs in polynomial time. Therefore, in this section we set out to develop a mixed integer linear program (MILP) for the exact solution of the problem. This approach is beneficial in two ways. Firstly, it allows us to use the rich theory and highly optimized tools of integer linear programming, in order to efficiently explore the search space. Secondly, the MILP framework, once constructed, makes it possible to tackle different problem variants with only slight modifications.

The power, and also the computational hardness, of integer linear programming lies in the fact that binary 0/1 variables can easily model logical decisions. Therefore, many combinatorial optimization problems can be formulated as ILPs, as long as the problem constraints can be cast as linear inequalities. In order to do so, we often need to translate statements of propositional logic into linear inequalities involving the variables. In what follows we expose step by step the logic that leads us to the final MILP formulation for `MINSENSOR`.

Our starting point is a well-known connection between bipartite matchings and linear programming. Namely, given a bipartite graph $G = (C, R; E)$, we can express the existence of a C-perfect matching as the feasibility of a linear program (LP) that contains $|E|$ real-valued variables, $|R|$ inequalities and $|C|$ equalities.

LP 1 Existence of a perfect matching as an LP feasibility problem

$$z_{ji} \geq 0 \qquad \forall j \in C, \forall i \in N(j) \tag{4.1}$$

$$\sum_{j \in N(i)} z_{ji} \leq 1 \qquad \forall i \in R \tag{4.2}$$

$$\sum_{i \in N(j)} z_{ji} = 1 \qquad \forall j \in C \tag{4.3}$$

Correctness (of LP 1). *The graph $G = (C, R; E)$ has a matching that covers all vertices of C if and only if there exists a vector z that is a feasible solution of LP 1.*

To prove the correctness of LP 1, we need to recall some standard facts about totally unimodular (TU) matrices that can be found, e.g., in [Wol98].

Definition 4.3. A matrix is called *totally unimodular* if every square submatrix of it has determinant equal to 0 or ± 1.

Proposition 4.4.

1. *If the rows of a $0/1$ matrix can be partitioned in two sets such that each column contains at most one 1 in each partition, then the matrix is TU.*
2. *A TU matrix remains TU under the following operations: append a copy of an existing row, append a unit row-vector, multiply a row with -1.*

Theorem 4.5. *If A is a TU matrix and b is an integer vector, then all vertices of the convex polyhedron $P = \{x : x \geq 0, Ax \leq b\}$ are integer.*

Now we are finally ready to prove the correctness of LP 1.

Proof. First, we show that there exists a feasible vector z if and only if there exists a feasible 0/1 vector $\widehat{z}$. The reverse direction is obvious; here we prove the forward direction. If z is feasible, then it lies in the polyhedron $P = \{z : z \geq 0, \begin{pmatrix} A & B & -B \end{pmatrix}^T z \leq \begin{pmatrix} 1 & 1 & -1 \end{pmatrix}^T\}$, where A and B are the matrices corresponding to (4.2) and (4.3), respectively. Each variable z_{ji} (recall that variables correspond to edges) appears with coefficient 1 exactly once in (4.2) (incidence to the R-vertex of the edge) and exactly once in (4.3) (incidence to the C-vertex of the edge). Therefore, from the first part of Proposition 4.4 it follows that the matrix $\begin{pmatrix} A & B \end{pmatrix}^T$ is TU and from the second part we finally get that $\begin{pmatrix} A & B & -B \end{pmatrix}^T$ is also TU. Since P is non-empty and bounded, it has at least one vertex $\widehat{z}$ and, due to Theorem 4.5, this vertex has to be integer. Furthermore, (4.1) and (4.2) imply that $\widehat{z}_{ji} \in [0, 1]$ for all j and i. Therefore, $\widehat{z}$ is a 0/1 feasible solution of LP 1.

Now, we show that a 0/1 feasible solution $\widehat{z}$ corresponds to a C-perfect matching in G. Since the variables z_{ji} correspond to the edges of G, a feasible 0/1 assignment of z indicates a subset of edges $M \subseteq E$ such that each vertex of C is incident to exactly one edge of M [equality (4.3)] and each vertex of R is incident to at most one edge of M [inequality (4.2)]. The edge set M is a C-perfect matching. □

We can immediately extend LP 1 in order to express *identifiability* of a bipartite graph as an LP feasibility problem. Recall that identifiability of a graph $G = (C, R; E)$ is defined as the existence of perfect matchings in certain ($|C|$ many) subgraphs of G. So, all we have to do is to enforce the existence of a complete matching in the subgraphs $\widetilde{G}_s$, for all $s \in C$. In LP 2 we do this, by explicitly writing down $|C|$ sets of inequalities of type LP 1.

Correctness (of LP 2)**.** *The graph $G = (C, R; E)$ is identifiable if and only if there exists a vector z that is a feasible solution of LP 2.*

Finally, we can now give a complete formulation for the exact solution of `MINSENSOR`. More specifically, we develop a mixed integer linear program (MILP 1) that contains $|R|$ 0/1 variables y_i such that $y_i = 1$ if and only if $i \in I$. It also contains $\mathcal{O}(|C||E|)$

LP 2 Identifiability as an LP feasibility problem

$$z^s_{ji} \geq 0 \qquad \forall s \in C, \forall j \in C \setminus \{s\}, \forall i \in N(j) \setminus N(s) \tag{4.4}$$

$$\sum_{j \in N(i) \setminus \{s\}} z^s_{ji} \leq 1 \qquad \forall s \in C, \forall i \in R \setminus N(s) \tag{4.5}$$

$$\sum_{i \in N(j) \setminus N(s)} z^s_{ji} = 1 \qquad \forall s \in C, \forall j \in C \setminus \{s\} \tag{4.6}$$

real-valued variables z^s_{ji}, in order to enforce the identifiability of the subgraph $G[C, I]$. For a vertex $s \in C$, the variables z^s_{ji} and the corresponding group of constraints (4.8)-(4.11) guarantee the identifiability of s in $G[C, I]$. The correctness proof of MILP 1 closely follows the correctness proof of LP 1.

MILP 1 Formulation for the exact solution of `MINSENSOR`

$$\text{minimize:} \quad \sum_{i \in R} c(i) y_i$$

subject to:

$$y_i \in \{0, 1\} \qquad \forall i \in R \tag{4.7}$$

$$z^s_{ji} \geq 0 \qquad \forall s \in C, \forall j \in C \setminus \{s\}, \forall i \in N(j) \setminus N(s) \tag{4.8}$$

$$z^s_{ji} \leq y_i \qquad \forall s \in C, \forall j \in C \setminus \{s\}, \forall i \in N(j) \setminus N(s) \tag{4.9}$$

$$\sum_{j \in N(i)} z^s_{ji} \leq 1 \qquad \forall s \in C, \forall i \in R \setminus N(s) \tag{4.10}$$

$$\sum_{i \in N(j) \setminus N(s)} z^s_{ji} = 1 \qquad \forall s \in C, \forall j \in C \setminus \{s\} \tag{4.11}$$

Correctness (of MILP 1). *A vector y defines a set $I \subseteq R$ such that $G[C, I]$ is identifiable if and only if there exists a vector z such that (y, z) is a feasible solution of MILP 1.*

Proof. First, we show that there exists a vector z such that (y, z) is feasible if and only if there exists a 0/1 vector $\widehat{z}$ such that $(y, \widehat{z})$ is feasible. The reverse direction is obvious; here we prove the forward direction. If (y, z) is feasible, then y is a 0/1 vector and z lies in the polyhedron $P(y)$ defined by (4.8)-(4.11). Each variable z^s_{ji} appears with coefficient 1 exactly once in (4.10) and exactly once in (4.11); combined with Proposition 4.4, this implies that the matrix defining $P(y)$ is totally unimodular. Since $P(y)$ is non-empty and bounded, it has at least one vertex $\widehat{z}$ and, due to Theorem 4.5, this vertex has to be integer. Furthermore, (4.8) and (4.9) imply that $\widehat{z}^s_{ji} \in [0,1]$ for all s, i and j. Therefore, $(y, \widehat{z})$ is a 0/1 feasible solution of MILP 1.

Now, we show that y defines a set $I \subseteq R$ such that $G[C, I]$ is identifiable if and only if there exists a 0/1 vector $\widehat{z}$ such that $(y, \widehat{z})$ is feasible. For each edge (i, j) of $G[C \setminus \{z\}, R \setminus N(z)]$ there is one variable z^s_{ji}, and condition (4.9) is equivalent to the statement: $i \notin I \Rightarrow z^s_{ji} = 0$. The graph $G[C, I]$ is identifiable if and only if for each $s \in C$ there exists a matching M_s in $G[C \setminus \{s\}, I \setminus N(s)]$ that covers all vertices of $C \setminus \{s\}$. This can be expressed as a system of linear inequalities: There is a set of edges M_s in $G[C \setminus \{s\}, I \setminus N(s)]$ such that each vertex of $C \setminus \{s\}$ is incident to exactly one edge of M_s and each vertex of $I \setminus N(s)$ is incident to at most one edge of M_s. This happens if and only if there is a 0/1 assignment of the variables z^s_{ji} that satisfies (4.9)-(4.11). □

4.3 Cutting-plane approaches to solve `MINSENSOR`

In MILP 1, there are a few binary variables y_i that are used to choose the optimal subset of sensors and a lot more continuous variables z^s_{ji}, whose only purpose is to guard the identifiability of the subgraph $G[C, I]$. We observe that the z-variables and the corresponding constraints have a simple structure. For a vertex $s \in C$, the variables z^s_{ij} and the constraints (4.8)-(4.11) enforce that s will be identifiable in $G[C, I]$. Moreover, for $s_1, s_2 \in C$ with $s_1 \neq s_2$, the corresponding z-variables and constraints are completely decoupled.

This observation motivates us to develop an incremental approach for `MINSENSOR`. Instead of setting out to solve the full-fledged MILP 1 at once, we first try to solve a relaxation of `MINSENSOR` (that we call `MINSENSOR-REL`) and add the matching constraints on demand, only if they are violated. In the relaxation we replace the requirement of $G[C, I]$ being identifiable with two necessary (but not sufficient) conditions that must hold in $G[C, I]$: (i) every vertex $s \in C$ must have at least $|C| - 1$ non-neighbors and (ii) for all $s_1, s_2 \subseteq C$ with $s_1 \neq s_2$ it must be $N(s_1) \nsubseteq N(s_2)$. The same construction as that of Section 4.1 shows that `MINSENSOR-REL` is also NP-hard. The corresponding ILP is much simpler, as it contains only the binary variables y_i and $|C|^2$ constraints. More specifically, condition (i) is equivalent to $\sum_{i \in R \setminus N(s)} y_i \geq |C| - 1$ and condition (ii) is equivalent to $\sum_{i \in N(s_1) \setminus N(s_2)} y_i \geq 1$.

Of course, a solution of the relaxed problem is not guaranteed to be identifiable. However, given a candidate solution $G[C, I]$, we can *efficiently* check a-posteriori, if any of the $|C|$ matching constraints is violated; this is done by solving $|C|$ bipartite maximum matching problems. If no constraint is violated, then `MINSENSOR` has been optimally solved, otherwise, for vertices $s \in C$ that are not identifiable in $G[C, I]$, we add the corresponding block of z-variables and constraints (4.8)-(4.11) and reiterate. It is important to note that we can check in *polynomial time* if a candidate solution violates any of the constraints and, if this is the case, also detect the violated constraints. Clearly, this approach of adding the constraints on demand does not offer any theoretical advantage, since it is in principle possible to iterate up to $|C|$ times, each time solving a larger MILP. However, in Section 4.5, we demonstrate that this approach eventually pays off for a wide range of randomly generated problem instances.

From a geometric point of view, replacing the original problem with a relaxation corresponds to optimizing the objective function over a polyhedron (call it P_r) that contains the polyhedron of the original problem (call it P_o). Given an optimal solution x^* of the relaxation, which lies of course in P_r, we need an oracle to decide if x^* also lies in P_o. If this is not the case, we also want the oracle to return a hyperplane that separates the point x^* from the interior of P_o; this corresponds to detecting a violated inequality in the ILP formulation. This hyperplane "cuts" from P_r the part towards the direction of x^* that does not belong to P_o and, therefore, it is called *cutting-plane*. For the oracle we say that it solves the *separation problem*. Finding a separation oracle that runs in polynomial time is not always possible, but in our case this is a simple consequence of the problem structure.

Below we present an alternative way to add cutting-planes, which is based on Hall's marriage theorem. As we explained before, given a solution $I \subseteq R$ that is optimal for the relaxed problem, we check a-posteriori, for all $s \in C$, if the subgraph $G[C \setminus \{s\}, I \setminus N(s)]$ has a matching of size $|C| - 1$. If this is the case, the optimal solution of the relaxed problem is also an optimal solution of `MINSENSOR` and we are done. Let us now assume that for some $s \in C$ the matching constraint fails. Then, Hall's marriage theorem (see Theorem 3.6) implies that there exists a set $X_v \subseteq C \setminus \{s\}$ such that $|N(X_v) \cap (I \setminus N(s))| < |X_v|$. And here comes the difference to the previous approach. Instead of adding the whole group of variables z^s_{ji} and the corresponding constraints, we now detect a particular set X_v that causes the problem. For these particular $s \in C$ and $X_v \subseteq C \setminus \{s\}$ we add to the ILP the following constraint that enforces the desired condition $|N(X_v) \cap (I \setminus N(s))| \geq |X_v|$:

$$\sum_{i \in N(X_v) \setminus N(s)} y_i \geq |X_v|.$$

In contrast to the first approach of adding the whole group of variables z^s_{ji}, adding only the above constraint does *not* exclude the existence of another set $X'_v \subseteq C \setminus \{s\}$ that violates Hall's condition. Theoretically, it is even possible to have exponentially many such violations, which would lead to exponentially many cutting-plane iterations and constraints. However, in Section 4.5 we show experimentally that, at least for randomly generated problem instances, this second cutting-plane approach is more efficient than the first one.

In the above discussion, there is one important point that we have left open: How can we actually find a set X_v that violates Hall's condition? In other words, given a bipartite graph $G = (A, B; E)$ that does *not* have an A-perfect matching, can we find efficiently a set $X_v \subseteq A$ such that $|N(X_v)| < |X_v|$? It turns out that we actually get such a set "for free" by taking a look at the internals of the maximum matching algorithm. Even more strongly, we get a minimizer of the surplus function $\sigma(X) = |N(X)| - |X|$, i.e., a set that maximally violates Hall's condition. In order to show that, we need Proposition 4.6; for its proof see, e.g., [LP86].

Proposition 4.6. *If $G = (A, B; E)$ is a bipartite graph with no A-perfect matching, then the size of a maximum matching $\nu(G) = |A| + \sigma(G)$* [2]

Proposition 4.7. *Let $G = (A, B; E)$ be a bipartite graph that has no A-perfect matching and no isolated vertices in A. If VC is a minimum vertex cover of G, then $\sigma(A \setminus VC) = \sigma(G)$.*

Proof. From König's theorem and the assumption that no A-perfect matching exists, we get $|VC| = \nu(G) < |A|$. If $VC \cap B = \emptyset$, then $VC \subset A$ and since VC covers all edges of G, it follows that A contains some isolated vertices. This contradicts the assumption and, therefore, it must be $VC \cap B \neq \emptyset$.

We set $B' := VC \cap B$, $A' := VC \cap A$ and $A'' := A \setminus VC = A \setminus A'$. Due to Proposition 4.6 and König's theorem, we have $\sigma(G) = \nu(G) - |A| = |VC| - |A| = |A'| + |B'| - |A| = |B'| - |A''|$. On the other hand, the fact that VC is a vertex cover implies that $N(A'') \subseteq B'$ and, therefore, $\sigma(A'') = |N(A'')| - |A''| \leq |B'| - |A''| = \sigma(G)$. The minimality of $\sigma(G)$ finally implies $\sigma(A'') = \sigma(G)$. □

And here comes the last piece of the puzzle: In a bipartite graph a minimum vertex cover is automatically computed along with a maximum matching (see Algorithm 2 in Section 3.2). So, when we use this algorithm to check whether a graph has an A-perfect matching, we also get for free a set that maximally violates Hall's condition, if no A-perfect matching exists.

4.4 A greedy approximation algorithm

Recall that the input to the `MINSENSOR` problem is a bipartite graph $G = (C, R; E)$ together with a function $c : R \to \mathbb{Q}^+$, and the goal is to find a set $I \subseteq R$ such that the subgraph $G[C, I]$ is identifiable and $\sum_{y \in I} c(y)$ is minimized. In this section we show that there exists an efficient greedy algorithm for `MINSENSOR` that guarantees a logarithmic approximation of the optimal solution. A variant of the algorithm that improves on the running time will be discussed in Sect. 4.4.1.

[2] Recall that $\sigma(G) = \min_{X \subseteq A}\{|N(X)| - |X|\}$. Since G has no A-perfect matching, it is $\sigma(G) < 0$.

Theorem 4.8. *There exists a greedy algorithm that computes an $H(\tau)$-approximate solution to* ***MINSENSOR****, where $\tau = |C| - \min_{y \in R} d(y)$. The algorithm runs in time $\mathcal{O}\left(|R|^2|C|\sqrt{|R|+|C|}|E|\right)$.*

In the above theorem, $d(y)$ denotes the degree of y and $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ the n-th harmonic number. Since $H(n) \leq \ln n + 1$ for $n \geq 1$, the algorithm given by the theorem provides an $(\ln|C| + 1)$-approximation to MINSENSOR. We will show later in this section that this bound on the approximation ratio is almost tight.

Before proceeding to the proof of Theorem 4.8, we must recall some preliminaries about submodular functions and state some of the related well-known results, which we will use in our proof.

Definition 4.9. A real-valued set function f defined on the power set of a finite ground set R is called *submodular* if $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$, for any two sets $X, Y \subseteq R$.

Submodular functions are a key concept in combinatorial optimization; see for example the books [Fuj05, Nar97]. For our purposes, the following problem related to submodular functions will be of interest, together with a greedy algorithm for it and the corresponding analysis by Wolsey.

Problem 7. MINIMUM SUBMODULAR COVER
We are given a finite ground set R, a nonnegative cost function c defined on R and an integer-valued, non-decreasing and submodular function f on R. Our task is to find a set $I \subseteq R$ that satisfies $f(I) = f(R)$ and minimizes $\sum_{y \in I} c(y)$.

Algorithm 3 Greedy algorithm for MINIMUM SUBMODULAR COVER

```
1: $I \leftarrow \emptyset$
2: repeat
3:     for all $y \in R \setminus I$ do
4:         $\Delta f(y) \leftarrow f(I \cup \{y\}) - f(I)$
5:     end for
6:     $y^* \leftarrow \text{argmax} \frac{\Delta f(y)}{c(y)}$
7:     $I \leftarrow I \cup \{y^*\}$
8: until $f(I) = f(R)$
9: return $I$
```

Theorem 4.10 (Wolsey [Wol82]). *Algorithm 3 is an $H(\tau)$-approximation algorithm for MINIMUM SUBMODULAR COVER, where $\tau = \max_{y \in R} f(\{y\}) - f(\emptyset)$.*

The following proposition will also be used in our proof of Theorem 4.8.

Proposition 4.11 (Lovász and Plummer [LP86]). *Let $G = (A, B; E)$ be a bipartite graph and for $X \subseteq B$ let $g(X) = \nu(G[A, X])$. Then, g is a submodular function.*

We are now ready to prove Theorem 4.8.

Proof. The proof relies on Theorem 4.10. In fact, by defining a suitable submodular function f, we will show that `MINSENSOR` is a special case of `MINIMUM SUBMODULAR COVER`. Therefore, we can apply the framework of Algorithm 3 together with Wolsey's analysis of its performance guarantee.

Given a bipartite graph $G = (C, R; E)$, we define an integer-valued potential function f on our search space, as follows: for a set $X \subseteq R$ the value of f at X is given by:

$$f(X) = \sum_{z \in C} \nu\left(G_z[X]\right). \tag{4.12}$$

(Recall that $G_z[X]$ denotes the graph $G[C \setminus \{z\}, X \setminus N(z)]$.)

Intuitively, our potential function f evaluates the goodness of a set $X \subseteq R$ by computing its total contribution to the matching numbers of the subgraphs $G_z[X]$, for all $z \in C$. By definition, a vertex $z \in C$ is identifiable in $G[C, X]$ if and only if the matching number of $G_z[X]$ equals $|C| - 1$. Therefore, $f(X) \leq |C|(|C| - 1)$, for all $X \subseteq R$, and equality holds if and only if $G[C, X]$ is identifiable.

We now show that the function f given by (4.12) defines a natural mapping from the instances of the `MINSENSOR` problem to those of the `MINIMUM SUBMODULAR COVER`. To this end, we need to verify that $f(I) = f(R)$ if and only if $G[C, I]$ is identifiable, and that f is non-decreasing and submodular.

- *f is non-decreasing.*

  This follows immediately from the definition, since adding an R-vertex to a set I cannot decrease the matching number of any $G_z[I]$. Therefore $f(I \cup \{y\}) \geq f(I)$, for every $I \subset R$ and every $y \in R \setminus I$.

- *f is submodular.*

  For $z \in C$ and $X \subseteq R$, we set $g_z(X) = \nu(G_z[X])$. Since f is defined as the sum of the functions g_z, and the sum of submodular functions is submodular, is suffices to show that the functions g_z are submodular, for all $z \in C$. We apply Proposition 4.11 to the subgraph G_z. Let $X, Y \subseteq R$. The submodularity condition given by Proposition 4.11, applied to the sets $X \setminus N(z)$ and $Y \setminus N(z)$, results in the following inequality:

$$\nu(G_z[X]) + \nu(G_z[Y]) \geq \nu(G_z[X \cup Y]) + \nu(G_z[X \cap Y]).$$

This gives $g_z(X) + g_z(Y) \geq g_z(X \cup Y) + g_z(X \cap Y)$, i.e., g_z is submodular. Therefore, f is submodular too.

- *For every $I \subseteq R$, $f(I) = f(R)$ if and only if $G[C, I]$ is identifiable.*

  Since f is non-decreasing, an instance of `MINSENSOR` is feasible if and only if the given graph $G = (C, R; E)$ is identifiable, that is, $f(R) = |C|(|C|-1)$. Assuming this equality, the fact that $f(I) = |C|(|C|-1)$ if and only if $G[C, I]$ is identifiable, establishes that $f(I) = f(R)$ if and only if $G[C, I]$ is identifiable.

Therefore, Theorem 4.10 implies that, with the function f given by (4.12), Algorithm 3 is an $H(\tau)$-approximation algorithm for `MINSENSOR`, where $\tau = \max_{y \in R} f(\{y\}) - f(\emptyset)$. In our case, $f(\emptyset) = 0$ and $f(\{y\}) = |C| - d(y)$ for all $y \in R$; therefore, we conclude that the greedy algorithm for `MINSENSOR` returns a solution that approximates the optimal one within a ratio of at most $H(|C| - \min_{y \in R} d(y))$.

The claimed time complexity of $\mathcal{O}\left(|R|^2|C|\sqrt{|R| + |C|}|E|\right)$ follows from a straightforward implementation of Algorithm 3 (with the function f given by (4.12)), computing the maximum matchings using, e.g., the algorithm by Hopcroft and Karp [HK73] (which computes the matching number of a bipartite graph with n vertices and m edges in time $\mathcal{O}(\sqrt{n}m)$). □

We now show that our bound from Theorem 4.8 for the performance of the greedy algorithm is tight up to an additive term of 2. Inspired by tight examples for the greedy algorithm for the *set cover* problem (see, e.g., [Vaz04]) we can construct instances for `MINSENSOR` on which the greedy algorithm returns a solution which is at least a factor of $\ln |C| - 1$ away from optimality.

Theorem 4.12. *The approximation ratio of the greedy algorithm for `MINSENSOR` (Algorithm 3 with f given by (4.12)) is at least* $\ln |C| - 1$.

Proof. Consider the following hitting set instance: $T = \{1, \ldots, n+1\}, \mathcal{F} = \{\{i, n+1\} : 1 \leq i \leq n\}$. We construct an instance $G = (C, R; E)$ and c for `MINSENSOR` as follows: G is the graph obtained by applying to $(T, \mathcal{F})$ the transformation used in the proof of Theorem 4.1. The costs of vertices in $R = \{v_1, \ldots, v_{3n+2}\}$ (respecting the order of corresponding rows in the zero pattern) are given as follows: for $1 \leq i \leq n$, we have $c(v_i) = 1/i$; moreover $c(v_{n+1}) = 1 + \epsilon$ (for some $\epsilon > 0$, to be specified later), while the remaining vertices are of zero cost. For small enough ϵ, there is a unique optimal solution of cost $1 + \epsilon$ given by $\{v_j : n + 1 \leq j \leq 3n + 2\}$. The solution returned by the greedy algorithm is $R \setminus \{v_{n+1}\}$ and is of cost $H(n) = H(|C|/2)$. The ratio is $H(|C|/2)/(1 + \epsilon)$, which is at least $\ln |C| - 1$ for small enough ϵ. □

4.4.1 Improving the running time

It turns out that it is possible to improve the running time of the greedy approximation algorithm for `MINSENSOR` from $\mathcal{O}\left(|R|^2|C|\sqrt{|R|+|C|}|E|\right)$ down to $\mathcal{O}\left(|R||C||E|\right)$. The key observation is that for the special case of the potential function defined by (4.12), the computation of line 4 in Algorithm 3 does not require the evaluation of $f(I \cup \{y\})$ from scratch at each iteration. Instead, we rewrite $\Delta f(y)$ as a sum of terms each of which is either 0 or 1:

$$\Delta f(y) = f(I \cup \{y\}) - f(I) = \sum_{z \in C} \Big(\nu\left(G_z[I \cup \{y\}]\right) - \nu\left(G_z[I]\right)\Big). \qquad (4.13)$$

Thus, we consider, for each $z \in C$, the subgraph $G_z[I]$ and we want to check if the addition of vertex y to I will increase the matching number of $G_z[I]$ by one. This computation can be performed efficiently using the properties of maximum matchings.

The pseudocode of this improved algorithm (Algorithm 4) is given below. Its main properties are summarized in Proposition 4.13. The algorithm relies on a well-known theorem by Berge [Ber57], a result of central importance for matching algorithms. Berge's Theorem holds for arbitrary graphs and states that a matching M in a graph G is maximum if and only if G has no *M-augmenting path*, i.e., a path whose edges alternate between matched and unmatched and whose endpoints are non-matched. If there exists an M-augmenting path P, then a matching M' larger than M can be immediately obtained by replacing in M the matched edges along this path with the unmatched ones. In formulae, $M' = M \triangle E(P)$ where $\triangle$ denotes the symmetric difference operator.

Proposition 4.13. *Algorithm 4 is correct and computes, in time $\mathcal{O}\left(|R||C||E|\right)$, an $H(\tau)$-approximate solution to `MINSENSOR`, where $\tau = |C| - \min_{y \in R} d(y)$.*

Proof. Observing that Algorithm 4 is closely related to Algorithm 3, its correctness follows from the following invariants of the algorithm:

(i) For every $z \in C$ and for every I, the set $M_z[I]$ is a subset of E that forms a maximum matching in $G_z[I]$.

(ii) For every $z \in C$ and for every I, the set $R_z[I]$, as defined in line 9, is equal to the set of all vertices y in $R \setminus I$ such that $\nu(G_z[I \cup \{y\}]) = \nu(G_z[I]) + 1$.

(iii) For every I and every $y \in R \backslash I$, the value of $\Delta f(y)$, as defined in line 12, is equal to $f(I \cup \{y\}) - f(I)$.

Property (iii) follows from (ii), using equation (4.13). Properties (i) and (ii) can be proved simultaneously by induction on $|I|$ and using Berge's Theorem together with the following observation, which follows directly by the construction of the graph $D_z[I]$:

Algorithm 4 Greedy algorithm for `MINSENSOR`

```
1: $I \leftarrow \emptyset$
2: for all $z \in C$ do
3:     $M_z[\emptyset] \leftarrow \emptyset$
4: end for
5: repeat
6:     for all $z \in C$ do
7:         Construct $D_z[I]$, the directed graph obtained from $G_z$ by orienting in it the edges
           of $M_z[I]$ from $R$ to $C$ and all the other edges from $C$ to $R$. In addition, there is
           a new vertex $s^*$ with no incoming arc and with an outgoing arc to each vertex in
           $C \setminus \{z\}$ unmatched by $M_z[I]$.
8:         $R_z[I] \leftarrow$ the set of all vertices in $R \setminus (I \cup N_G(z))$ reachable by a directed path from
           $s^*$ in $D_z[I]$.
9:     end for
10:    for all $y \in R \setminus I$ do
11:        $\Delta f(y) \leftarrow |\{z : z \in C, \quad y \in R_z[I]\}|$
12:    end for
13:    $y^* \leftarrow \text{argmax}\frac{\Delta f(y)}{c(y)}$
14:    for all $z \in C$ do
15:        $M_z[I \cup \{y^*\}] \leftarrow \begin{cases} M_z[I] \triangle P, & \text{if } y^* \in R_z[I]; \\ M_z[I], & \text{otherwise.} \end{cases}$
           Here, $P$ is an augmenting $M_z[I]$-path in $G_z[I \cup \{y^*\}]$.
16:    end for
17:    $I \leftarrow I \cup \{y^*\}$
18: until $f(I) = f(R)$
19: return $I$
```

Observation For every I, every $z \in C$ and every $y \in R \setminus (I \cup N_G(z))$, there exists an $M_z[I]$-augmenting path in $G_z[I \cup \{y\}]$ if and only if $y \in R_z[I]$.

It remains to analyze the algorithm's running time:

- For every I and every $z \in C$, the computation corresponding to lines 7–9 can be performed, for example using breadth-first search, in time $\mathcal{O}(|E|)$. Over all repeat iterations, the computation corresponding to lines 7–9 contributes for a total of $\mathcal{O}(|R||C||E|)$.
- The computation in line 12 contributes, over all repeat iterations, for a total of $\mathcal{O}(|R|^2|C|)$.
- The computation in line 14 contributes, over all repeat iterations, for a total of $\mathcal{O}(|R|^2)$.
- The computation in line 16 contributes, over all repeat iterations, for a total of $\mathcal{O}(|R||C||E|)$.

Therefore, the total time complexity is $\mathcal{O}(|R||C| \max\{|R|, |E|\})$, which is of the order $\mathcal{O}(|R||C||E|)$, if there are no isolated vertices in R. However, this assumption is easily justified as any such vertices can be a-priori detected and deleted from the graph. $\square$

4.5 Computational experiments on randomly generated instances

In this section we investigate how the methods from Section 4.3 and the greedy algorithm of Section 4.4 perform on simulated instances of the unit-cost restriction of `MINSENSOR`. We generated random bipartite graphs $G = (C, R; E)$, with 24 different parameter combinations: $|C| \in \{30, 60, 90\}$, $|R| \in \{2|C|, 3|C|\}$ and edge density $d \in \{10\%, 20\%, 30\%, 40\%\}$. For each combination we generated 100 random graphs by drawing each edge, independently from the others, with probability d.

- S-CP: This algorithm corresponds to the straightforward cutting plane approach of Section 4.3. More precisely, it is the incremental method where the identifiability requirements are replaced by two necessary (but not sufficient) conditions. Then, given a solution of the relaxed problem, we check if any of the matching constraints is violated and, if it is the case, we add the corresponding constraints to the MILP.
- H-CP: This is the cutting plane approach based on Hall's theorem described in Section 4.3. The main difference from S-CP is that the constraints that are added to the MILP (if violations exist) are based on Hall's theorem.
- GREEDY: This algorithm is the greedy approach described in Section 4.4.

The programs are solved on a 64-bit Sparc machine (450 MHz, 4 GB of RAM), using the AMPL/CPLEX software platform (version 10.2) with default CPLEX parameters, and an upper limit of 20 CPU minutes for the solution of a single ILP. The ILPs are simplified by the presolver of AMPL, before being sent to the optimization engine.

The results of the cutting-plane approaches of Section 4.3 are summarized in Table 4.1. As described in Section 4.2, we first try to solve the relaxed problem and we add the matching constraints on demand, only if they are violated. It turns out that in most cases the solution of the relaxed problem does not violate the matching constraints and is, therefore, a solution of the original problem. This is in accordance with our intuition: In a randomly generated, dense enough bipartite graph, large matchings exist with high probability. In the context of `MINSENSOR`, if we make sure that each $s \in C$ has enough (at least $|C| - 1$) non-neighbors in $G[C, I]$, then it is likely that $G[C \setminus \{s\}, I \setminus N(s)]$ has a matching of size $|C| - 1$. In other words, for these randomly generated instances the computational bottleneck is the "coverage" of each $s \in C$ with at least $|C| - 1$ non-neighbors and not the matching constraints.

S-CP: Straightforward Cutting Plane Approach						
	$\|C\| = 30$		$\|C\| = 60$		$\|C\| = 90$	
d (%)	$\|R\| = 60$	$\|R\| = 90$	$\|R\| = 120$	$\|R\| = 180$	$\|R\| = 180$	$\|R\| = 270$
10	0.98ci 5890si	1.09ci 21694si	0.05ci 906si	0.08ci 4674si	0ci 23640si (95)	0ci 35781si (71)
20	0.01ci 101si	0.08ci 2054si	0ci 1490si	0ci 28887si (95)	0ci 32332si (77)	0ci 44335si (55)
30	0ci 45si	0ci 129si	0ci 858si	0ci 57413si (80)	0ci 22644si (86)	0ci 49805si (17)
40	0ci 23si	0ci 197si	0ci 61si	0ci 65078si (76)	0ci 4144si	0ci 42516si (11)

H-CP: Cutting Plane Approach Based on Hall's Theorem				
	$\|C\| = 30$		$\|C\| = 60$	
d (%)	$\|R\| = 60$	$\|R\| = 90$	$\|R\| = 120$	$\|R\| = 180$
10	2.44ci 282si	6.16ci 1130si	0.05ci 225si	0.08ci 1467si
20	0.02ci 70si	0.08ci 122si		

Table 4.1: Each cell corresponds to a certain parameter combination $(|C|, |R|, d)$. It contains the average number of cutting-plane iterations (ci) and the average total number of simplex iterations (si). The number of instances solved within the time limit (out of 100, all solved if number not given) is shown in parentheses. In the table H-CP we monitor only the cases where the addition of any cutting-planes was necessary; in the rest of the cases it was sufficient to solve the relaxed problem.

For the sparse instances (upper-left corner of Table 4.1), where the relaxation does not immediately yield a solution, a few cutting-plane iterations are enough in practice. Comparing the two approaches, S-CP and H-CP, we observe that the one based on Hall's theorem (H-CP) is more efficient in practice, as it takes in total fewer simplex iterations.

Of course, `MINSENSOR-REL` is itself a hard problem. As we face bigger and more dense zero patterns, moving toward the lower-right corner of Table 4.1, its solution is more and more computationally intensive and in several cases cannot be achieved within the time limit.

Table 4.2 shows the experimental results for the GREEDY algorithm. The algorithm produces solutions of high quality since the average approximation ratio and the maximum approximation ratio are close to 1. As is often the case with approximation algorithms, the performance of the algorithm on randomly generated instances seems to be better than its worst-case performance guarantee of $\mathcal{O}(\ln |C|)$ established in Section 4.4. In fact, on the generated instances, the approximation ratios do not even increase with the size of the instances.

GREEDY algorithm						
	$\lvert C\rvert = 30$		$\lvert C\rvert = 60$		$\lvert C\rvert = 90$	
d (%)	$\lvert R\rvert = 60$	$\lvert R\rvert = 90$	$\lvert R\rvert = 120$	$\lvert R\rvert = 180$	$\lvert R\rvert = 180$	$\lvert R\rvert = 270$
10	33	32	65	64	99	97
	1.09, 1.19	1.11, 1.19	1.05, 1.09	1.06, 1.11	1.05, 1.09	1.05, 1.08
20	36	35	75	72	111	109
	1.06, 1.14	1.07, 1.20	1.06, 1.13	1.07, 1.12	1.05, 1.09	1.06, 1.12
30	42	39	86	82	130	124
	1.06, 1.12	1.08, 1.15	1.06, 1.10	1.07, 1.13	1.06, 1.10	1.07, 1.11
40	52	46	104	96	157	146
	1.04, 1.08	1.08, 1.17	1.04, 1.08	1.07, 1.11	1.04, 1.06	1.07, 1.10

Table 4.2: First row: average size of optimal solution (rounded to integer). Second row: average approximation ratio and maximum approximation ratio.

Chapter 5

Selection of Independent Subnetworks

Summary of the chapter. We analyze the case where from a source-sensor network we want to isolate subsets of sources that can be measured independently from the rest. Thereby, we require that the corresponding subnetwork is identifiable, so that we can process the measurements in the matrix factorization framework. We show how this requirement leads to the definition of “nicely separable” sets of sources and we define two related combinatorial problems. For one of them, that we call `MINSOURCE`, we manage to prove that it is NP-hard. The reduction is similar in spirit to the reduction for `MINSENSOR`, but more complicated due to the extra constraints. We present some simple data reduction rules that can be used to simplify instances of these problems and, finally, we develop a MILP formulation for their exact solution.

5.1 Motivation

The main object of our study in this chapter is again a source-sensor bipartite network $G = (C, R; E)$, where C is the set of sources and R is the set of sensors. As described in Chapter 2, we are measuring the linear signal mixtures on the sensors and from these measurements we want to infer the source signals across time and the non-zero mixing coefficients.

We firstly consider a scenario where, due to limited budget, we cannot afford to take measurements at more than $k < |C|$ time samples. Let us recall that Proposition 3.1 implies the basic condition $\widehat{A} = AR \lhd \mathcal{Z}$, which in turn forms the basis for the definition of identifiable networks. In order for the assumption $\operatorname{rank}(X) = m$ of Proposition 3.1 to hold, the *necessary* condition $m \leq k$ must be satisfied, where m is the number of monitored signal sources and k is the (fixed) number of time samples. Therefore, having $k < |C|$ time samples at our disposal, we want to isolate from G a *subnetwork* that contains at most k sources. Of course, we also want this subnetwork to be identifiable, so that we can process the measurements in the `ZCAMF` framework. Is it easy to find an identifiable subnetwork with at most k sources?

Let us secondly consider the case where the initial graph G is *not identifiable*. Then, one option is to make it identifiable, by adding some appropriately designed sensors. If we cannot afford any more sensors, but we still want G to be of any use, we have to leave some sources out of our measurements. Which sources should we omit? Can we find a maximum subset of sources $J \subset C$ that induces an identifiable subgraph?

In both cases described above, we want to *isolate* a subset of sources $J \subset C$ and this creates an additional complication. Namely, if a sensor $x \in R$ is connected to a source $y \notin J$, then we cannot use x in our `ZCAMF` computation: The measurements on x depend on the signal of y, but we are not including y in our computation. In other words, selecting a set of sources $J \subset C$ *automatically restricts* the set of allowed sensors to the ones whose neighborhood is completely contained in J. This observation motivates the following definition.

Definition 5.1. Let us consider a bipartite graph $G = (C, R; E)$ and two sets $J \subseteq C$ and $I \subseteq R$. If $N(I) = J$, we will say that I *measures* J. If there exists an I that measures J, we will call J *separable*.

Remark 5.2. If $J \subseteq C$ is separable and $N(I_1) = N(I_2) = J$, then $N(I_1 \cup I_2) = J$. Therefore, among the sets that measure J there is a unique one with maximum cardinality; we will denote this set with $s(J)$. We should keep in mind that $s(J)$ is defined *only for a separable* set J.

Remark 5.3. It is easy to test if a given $J \subseteq C$ is separable. We just have to go once through all vertices $y \in R$ and keep aside in a set Y the ones that have $N(y) \subseteq J$. Finally, if $N(Y) = J$, then J is separable and $s(J) = Y$, otherwise J is not separable.

In contrary to `MINSENSOR`, in this chapter we will not be concerned about minimizing the number (or the cost) of the used sensors. Therefore, having chosen a set of sources J, it is reasonable to monitor them with as many sensors as possible, i.e. with the sensor set $s(J)$, because this can only improve the identifiability of the resulting subgraph.

Definition 5.4. Let us consider a graph $G = (C, R; E)$ and a set $J \subseteq C$. We will call J *nicely separable*, if it is separable and, moreover, the subgraph $G[J, s(J)]$ is identifiable.

Below we show that the family of nicely separable sets is closed under union (such a set family is called semi-lattice). Therefore, this family contains a unique set J^* with maximum cardinality. Moreover, any nicely separable set is contained in J^*.

Proposition 5.5. *Let us consider a graph $G = (C, R; E)$ and two sets $J_1, J_2 \subseteq C$. If both J_1 and J_2 are nicely separable, then so is also their union.*

Proof. Since J_1 and J_2 are nicely separable, $s(J_1)$ and $s(J_2)$ exist and, by definition, $N(s(J_1)) = J_1$ and $N(s(J_2)) = J_2$. Moreover, the subgraphs $H = G[J_1, s(J_1)]$ and $F = G[J_2, s(J_2)]$ are identifiable. Since $N(s(J_1) \cup s(J_2)) = J_1 \cup J_2$, we have that $J_1 \cup J_2$ is separable; moreover $s(J_1 \cup J_2) \supseteq s(J_1) \cup s(J_2)$. We now show that the graph $K = G[J_1 \cup J_2, s(J_1) \cup s(J_2)]$ is identifiable; this implies that $G[J_1 \cup J_2, s(J_1 \cup J_2)]$ is identifiable and concludes the proof.

Let us consider an arbitrary vertex $x \in J_1$. Since H is identifiable, $\widetilde{H}_x$ contains a matching M_1 that matches all vertices of $J_1 \setminus \{x\}$ onto $s(J_1)$. M_1 is a matching also in $\widetilde{K}_x$, because $\widetilde{H}_x$ is a subgraph of $\widetilde{K}_x$.

The identifiability of F implies the existence of a matching M that matches all vertices of J_2 onto $s(J_2)$. In particular, M matches the vertices of $J_2 \setminus J_1$ onto $s(J_2) \setminus s(J_1)$, because there are no edges between $J_2 \setminus J_1$ and $s(J_1)$. Let M_2 be the restriction of M on $(J_2 \setminus J_1) \cup (s(J_2) \setminus s(J_1))$. If $x \in J_1 \setminus J_2$, then $\widetilde{K}_x$ contains M_2, because there are no edges between $J_1 \setminus J_2$ and $s(J_2) \setminus s(J_1)$. In that case, $M_1 \cup M_2$ is a matching in $\widetilde{K}_x$ that covers all vertices of $(J_1 \cup J_2) \setminus \{x\}$.

If $x \in J_1 \cap J_2$, by the definition of identifiability, $\widetilde{F}_x$ contains a matching M' that matches all vertices of $J_2 \setminus \{x\}$ onto $s(J_2)$. M' is a matching also in $\widetilde{K}_x$, because $\widetilde{F}_x$ is a subgraph of $\widetilde{K}_x$. In particular, M' matches the vertices of $J_2 \setminus J_1$ onto $s(J_2) \setminus s(J_1)$. Let M_2' be the restriction of M' on the vertices $(J_2 \setminus J_1) \cup (s(J_2) \setminus s(J_1))$. Then, $M_1 \cup M_2'$ is a matching in $\widetilde{K}_x$ that covers all $(J_1 \cup J_2) \setminus \{x\}$.

Finally, we conclude that for any arbitrary $x \in J_1$ there exists a matching in $\widetilde{K}_x$ that covers all vertices of $(J_1 \cup J_2) \setminus \{x\}$ and, therefore, x is identifiable in K. Applying the same argument symmetrically shows that any arbitrary vertex of J_2 is identifiable. Therefore, the subgraph K is identifiable. □

Remark 5.6. The family of nicely separable sets is *not* closed under intersection; Figure 5.1 contains a counter-example. This is also an example for the fact that the family of separable sets is not closed under intersection.

Below we define formally two subgraph selection problems:

Problem 8. Given is a non-identifiable bipartite graph $G = (C, R; E)$.

1. Find any nicely separable set $J \subset C$.
2. Find the nicely separable set $J^* \subset C$ of *maximum* cardinality.

Problem 9. `MINSOURCE`
Given is a graph $G = (C, R; E)$ and a positive integer $k \leq |C|$. Is there a nicely separable set $J \subseteq C$ with $|J| \leq k$?

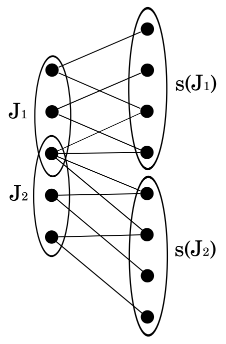


Figure 5.1: J_1 and J_2 are nicely separable: $G[J_1, s(J_1)]$ and $G[J_2, s(J_2)]$ are both identifiable. However, $J_1 \cap J_2$ is not nicely separable; in fact, it is not even separable.

Note that the problem becomes trivial if we relax the requirement "nicely separable" to just "separable". In order to find a minimum separable set, we just have to find an R-vertex with minimum degree and take its neighborhood. On the other hand, the difficulty of `MINSOURCE` intuitively arises from the fact that, as J becomes smaller, the set $s(J)$ also becomes smaller (if it exists at all) and, therefore, it is difficult for $G[J, s(J)]$ to contain all necessary matchings. In other words, it is not obvious how we can isolate small subsets of sources that can be measured *independently* from the rest with an identifiable design. The cardinality of the smallest such subset quantifies how strongly the sources are coupled to each other; in order to measure anything on the network, we must take measurements in at least that many time points.

In the next section, we show that `MINSOURCE` is NP-complete [1]. However, the complexity of (both parts of) Problem 8 remains open and this is, in fact, an interesting direction for further research. More specifically, it makes sense to investigate if the two parts are somehow related to each other. For example, if we have a polynomial-time algorithm that finds any nicely separable set in a graph, can we use it repeatedly and combine it with Proposition 5.5 in order to construct an algorithm that finds the maximum nicely separable set?

5.2 `MINSOURCE` is NP-complete

We present a reduction from `HITTING SET`, which is similar in spirit to the reduction that we did for `MINSENSOR`. However, the construction here is more complicated, because we also have to take into account the constraint of nice separability. For the ease of presentation, we use the representation of a bipartite graph as a zero pattern. More specifically, we model a bipartite graph $G = (A, B; E)$ with a zero pattern such

[1]Note that we have defined `MINSOURCE` as a decision problem.

that the columns correspond to part A, the rows correspond to part B, and the ones correspond to the edges in E.

Problem 10. `HITTING SET`
Given is a finite ground set T, a family $\mathcal{F}$ of subsets of T and a positive integer $\widehat{k} \leq |T|$. Is there a subset $S \subseteq T$, with $|S| \leq \widehat{k}$, that intersects all sets of $\mathcal{F}$?

Let $(\mathcal{F}, T, \widehat{k})$ be an instance of `HITTING SET` with $\mathcal{F} = \{C_1, \ldots, C_m\}$; without loss of generality we may assume that $T = [n] := \{1, \ldots, n\}$. Moreover, we assume that $m \geq 3$. As we did for `MINSENSOR`'s reduction, we firstly construct an $n \times m$ zero pattern $\mathcal{H}$, such that $\mathcal{H}_{ij} = 0$ if and only if $i \in C_j$. Then, `HITTING SET` is equivalent to asking if there is a set of at most $\widehat{k}$ rows that cover at least one zero in each column of $\mathcal{H}$.

In the following, we show how we can answer this question by solving an appropriately constructed instance of `MINSOURCE`. An example for this construction is illustrated in Figure 5.2. We construct another zero pattern $\mathcal{G}$ (that corresponds to graph G in `MINSOURCE`'s definition) as follows: $\mathcal{G}$ has $2m+n$ columns, which we label as $x_1, \ldots, x_m$; $y_1, \ldots, y_m$; $z_1, \ldots, z_n$, and $2m^2 - m + 3n$ rows divided in $m + 4$ blocks: $\mathcal{A}$, $\mathcal{B}_1, \ldots, \mathcal{B}_m$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$. Block $\mathcal{A}$ consists of three sub-blocks: $\mathcal{A}_1 = \widehat{\mathcal{H}}$ $(n \times m)$, $\mathcal{A}_2 = \mathbf{1}$ $(n \times m)$ and $\mathcal{A}_3 = I$ $(n \times n)$. For $j \in [m]$, block $\mathcal{B}_j$ consists of two identical $(2m - 2) \times m$ sub-blocks, filled with ones except for the j-th column which is all zero, and a third $n \times n$ block of zeros. Block $\mathcal{C}$ consists of three sub-blocks: $\mathcal{C}_1 = \mathbf{1}$ $(m \times m)$, $\mathcal{C}_2 = \mathbf{1} - I$ $(m \times m)$ and $\mathcal{C}_3 = \mathbf{0}$ $(m \times n)$. Block $\mathcal{D}$ consists of three sub-blocks: $\mathcal{D}_1 = \mathbf{0}$ $(n \times m)$, $\mathcal{D}_2 = \mathbf{1}$ $(n \times m)$ and $\mathcal{D}_3 = I$ $(n \times n)$. Finally, block $\mathcal{E}$ consists of three sub-blocks: $\mathcal{E}_1 = \mathbf{1}$ $(n \times m)$, $\mathcal{E}_2 = \mathbf{0}$ $(n \times m)$ and $\mathcal{E}_3 = I$ $(n \times n)$. For notational convenience we set $X := \{x_1, \ldots, x_m\}$, $Y := \{y_1, \ldots, y_m\}$, $Z := \{z_1, \ldots, z_n\}$ and $L :=$ the set of rows from the blocks $\mathcal{B}_1, \ldots, \mathcal{B}_m, \mathcal{C}$.

Lemma 5.7. *If J is a nicely separable set of columns in $\mathcal{G}$, then $J \supseteq X \cup Y$.*

Proof. We say that row i intersects (avoids) column j if $\mathcal{G}_{ij} = 1$ $(\mathcal{G}_{ij} = 0)$. First we show the following statements:

$$\forall j \in [m] : y_j \in J \Rightarrow J \supseteq X \setminus \{x_j\} \tag{5.1}$$

$$\forall j \in [m] : x_j \in J \Rightarrow J \supseteq Y \setminus \{y_j\} \tag{5.2}$$

Because J is nicely separable, there exists a set of rows $s(J)$ such that $N(s(J)) = J$ and $\mathcal{H}[J, s(J)]$ is identifiable. Let us assume that $y_j \in J$; then the identifiability of $\mathcal{H}[J, s(J)]$ implies that $s(J)$ contains at least one row that avoids y_j. But every row that avoids y_j intersects every column from the set $X \setminus \{x_j\}$. Therefore, $N(s(J)) \supseteq X \setminus \{x_j\}$ and (5.1) follows. Similarly, we can show (5.2).

Now, we show that if J contains *any* column from $X \cup Y$, then it contains them all. Thereby, we make use of the assumption that $m \geq 3$. Without loss of generality we assume that $x_1 \in J$ and from (5.1) and (5.2) we get the following sequence of implications:

$$x_1 \in J \Rightarrow J \supseteq Y \setminus \{y_1\} \Rightarrow y_2 \in J$$

$$y_2 \in J \Rightarrow J \supseteq X \setminus \{x_2\} \Rightarrow x_3 \in J$$

$$x_3 \in J \Rightarrow J \supseteq Y \setminus \{y_3\} \Rightarrow y_1 \in J$$

$$y_1 \in J \Rightarrow J \supseteq X \setminus \{x_1\} \Rightarrow x_2 \in J$$

From the above implications it follows that $J \supseteq X \cup Y$.

To complete the proof, it remains to show that every nicely separable set of columns contains at least one column from $X \cup Y$. For the sake of contradiction, suppose that there is a nicely separable set $J \subseteq Z$. The identifiability of $\mathcal{H}[J, s(J)]$ implies that for each column of J, $s(J)$ contains at least one row that intersects this column. But for every column $j \in Z$, every row that intersect j also intersects a column from $X \cup Y$. Therefore, $N(s(J)) \cap (X \cup Y) \neq \emptyset$ and this implies that $J \cap (X \cup Y) \neq \emptyset$, which is a contradiction. □

Lemma 5.8. *Block $\mathcal{A}$ contains a row subset I, with $|I| = t$, that covers each column of X with at least one zero if and only if there exists a nicely separable column set J, with $|J| = 2m + t$.*

Proof.
Forward: Let us assume that $I = \{i_1, \dots, i_t\} \subseteq [n]$. We take $J = X \cup Y \cup \{z_{i_1}, \dots, z_{i_t}\}$; J is separable, with $s(J) = L \cup \mathcal{A}[\{i_1, \dots, i_t\}] \cup \mathcal{D}[\{i_1, \dots, i_t\}] \cup \mathcal{E}[\{i_1, \dots, i_t\}]$. Now, we show that J is nicely separable, by verifying that $\mathcal{G}[J, s(J)]$ is identifiable. For each column x_j there exists a row index $i_j \in I$ with $x_j[i_j] = 0$. For column x_j consider the subpattern $\widetilde{\mathcal{G}}_{x_j}$: In $\widetilde{\mathcal{G}}_{x_j}$ all columns of $(X \cup Y) \setminus \{y_j\}$ are matched to the rows of $\mathcal{B}_j$, column y_j is matched to row i_j and the columns $\{z_{i_1}, \dots, z_{i_t}\}$ are matched to the rows of $\mathcal{D}[\{i_1, \dots, i_t\}]$. For column y_j, consider the subpattern $\widetilde{\mathcal{G}}_{y_j}$: In $\widetilde{\mathcal{G}}_{y_j}$ all columns of $(X \cup Y) \setminus \{x_j\}$ are matched to the rows of $\mathcal{B}_j$, column x_j is matched to the j-th row of $\mathcal{C}$ and the columns $\{z_{i_1}, \dots, z_{i_t}\}$ are matched to the rows of $\mathcal{E}[\{i_1, \dots, i_t\}]$. Finally, for each column $z \in \{z_{i_1}, \dots, z_{i_t}\}$ consider the subpattern $\widetilde{\mathcal{G}}_z$: In $\widetilde{\mathcal{G}}_z$ all columns of $X \cup Y$ can be matched to the rows of $\mathcal{B}_1$ and $\mathcal{B}_2$ and all columns of $\{z_{i_1}, \dots, z_{i_t}\} \setminus \{z\}$ are matched to the rows of $\mathcal{D}[\{i_1, \dots, i_t\}]$. Finally, we conclude that $\mathcal{G}[J, s(J)]$ is identifiable.

Reverse: Since J is nicely separable, Lemma 5.7 implies that $J \supseteq X \cup Y$. Since, $|J| = 2m + t$, it must be $J = X \cup Y \cup \{z_{i_1}, \dots, z_{i_t}\}$. Then, $s(J) = L \cup \mathcal{A}[\{i_1, \dots, i_t\}] \cup \mathcal{D}[\{i_1, \dots, i_t\}] \cup \mathcal{E}[\{i_1, \dots, i_t\}]$. The fact that $\mathcal{G}[J, s(J)]$ is identifiable implies that $s(J)$ covers each column of J with at least $|J| - 1 = 2m + t - 1$ zeros. But each column from

X has *exactly* $2m-2+t$ zeros in $L \cup \mathcal{D}[\{i_1,\dots,i_t\}] \cup \mathcal{E}[\{i_1,\dots,i_t\}]$. Therefore, the row subset $\mathcal{A}[\{i_1,\dots,i_t\}]$ must cover each column from X with at least one zero. □

Lemma 5.8 establishes the reduction of `HITTING SET` to `MINSOURCE` and, therefore, the NP-hardness of the latter. Furthermore, `MINSOURCE` is in NP, because for a given $J \subseteq C$, we can find $s(J)$ (if it exists) and, then, check if $G[J, s(J)]$ is identifiable, in polynomial time. Therefore, we arrive at the following conclusion:

Theorem 5.9. *`MINSOURCE` is NP-complete.*

$$
\begin{aligned}
T &= \{1,\dots,6\}\\
\mathcal{F} &= \left\{\begin{matrix}\{2,4,5\}\\ \{2,3,4,6\}\\ \{1,6\}\end{matrix}\right\}
\end{aligned}
\implies \mathcal{H} = \begin{pmatrix}
1 & 1 & \mathbf{0}\\
\mathbf{0} & \mathbf{0} & 1\\
1 & \mathbf{0} & 1\\
\mathbf{0} & \mathbf{0} & 1\\
\mathbf{0} & 1 & 1\\
1 & \mathbf{0} & \mathbf{0}
\end{pmatrix}
\implies \mathcal{G} =
\left(\begin{array}{c|ccc|ccc|cccccc}
 & x_1 & x_2 & x_3 & y_1 & y_2 & y_3 & z_1 & z_2 & z_3 & z_4 & z_5 & z_6\\
\hline
 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\
 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0\\
\mathcal{A} & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0\\
 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0\\
 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0\\
 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1\\
\hline\hline
 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
\mathcal{B}_1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
\hline
 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
\mathcal{B}_2 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
\hline
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\mathcal{B}_3 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\hline\hline
 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
\mathcal{C} & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\hline\hline
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\
 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0\\
\mathcal{D} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0\\
 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0\\
 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0\\
 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1\\
\hline\hline
 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
\mathcal{E} & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
$$

Figure 5.2: An example for the reduction of `HITTING SET` to `MINSOURCE`.

5.3 Data reduction rules

In this section, we present three rules, which can be used to simplify problem instances. In fact, applying these rules to regulatory networks (see Chapter 8) reduces the problems of subnetwork selection to trivial instances.

According to the following, we can safely eliminate neighborhood inclusions.

Proposition 5.10. *We consider a bipartite graph $G = (C, R; E)$ and two vertices $x, y \in C$. If $N(x) \subseteq N(y)$, then x cannot belong to a nicely separable set.*

Proof. Assume that x belongs to a nicely separable set $J \subseteq C$ and consider the subgraph $H = G[J, s(J)]$. Since all neighbors of x are also neighbors of y in G, the same also holds in H. Therefore, x is isolated (and non-matchable) in $\widetilde{H}_y$ and, therefore, H is not identifiable. This contradicts the assumption of J being nicely separable. □

If a C-vertex has more than one private (i.e. degree-one) neighbors, we can remove all of them, except for one, without changing the family of nicely separable sets. This simplification is particularly useful, if we decide to use integer linear programming, because it can significantly reduce the number of variables.

Proposition 5.11. *We consider a bipartite graph $G = (C, R; E)$. Let $G' = (C, R'; E')$ be the graph that results from removing the degree-one R-vertices of G such that each C-vertex has at most one private neighbor. A set $J \subseteq C$ is nicely separable in G if and only if it is nicely separable in G'.*

Proof. We consider two vertices $x, y \in C$. The claim easily follows from the following two observations: (i) removing the private neighbors of x does not affect the neighborhood of any other C-vertex (and therefore the matching number of $\widetilde{G}_x$) and (ii) in order to match x in $\widetilde{G}_y$ a single private neighbor of x is sufficient. □

Now we show how the Dulmage-Mendelsohn decomposition (DM decomposition) can be used in order to prune the search space of `MINSOURCE`. We remind the reader that the DM decomposition and its main properties were described in Section 3.5.1. Using the variable names of that section, we show below that when trying to solve `MINSOURCE`, we can safely discard the sources from partition C_3 and the sensors from partition R_3.

Proposition 5.12. *Let $G = (C, R; E)$ be a bipartite graph, decomposed in the Dulmage-Mendelsohn form. If $J \subseteq C$ is nicely separable, then $J \cap C_3 = \emptyset$ and $s(J) \cap R_3 = \emptyset$.*

Proof. For notational convenience we set $J^* = J \cap C_3$, $Z = G[J, s(J)]$ and $W = G[C_3, R_3]$. Let us assume that $J^* \neq \emptyset$. Since J is nicely separable, the subgraph Z is identifiable, therefore, it has non-negative surplus and, therefore, $|N_Z(J^*)| \geq |J^*| \geq 1$. From point (1) of Theorem 3.18, it follows that $N_Z(J^*) \subseteq R_3$. Then, we can apply point (3) of Theorem 3.18 on $N_Z(J^*)$ to get: $|N_W(N_Z(J^*))| \geq |N_Z(J^*)| + 1 \geq |J^*| + 1$.

On the other hand, $N_Z(J^*) \subseteq s(J)$ and, since W is a subgraph of G we can write: $N_W(N_Z(J^*)) \subseteq N_G(N_Z(J^*)) \subseteq N_G(s(J)) = J$. This implies that $|N_W(N_Z(J^*))| \leq |J^*|$, which is a contradiction. Therefore, we conclude that $J^* = \emptyset$. From point (3) of Theorem 3.18, it follows that every vertex $x \in R_3$ has at least two neighbors in C_3. Since $J \cap C_3 = \emptyset$, we conclude that no $x \in R_3$ can belong to $s(J)$. $\square$

5.4 An exact MILP solution for `MINSOURCE`

In this section we develop a mixed integer linear program (MILP) for the exact solution of (the minimization version of) `MINSOURCE`. Problem 8 defined at the beginning of this chapter can also be tackled with the same set of constraints. MILP 2 contains $|C|$ binary indicator variables x_j, such that $x_j = 1$ if and only if $j \in J$. It also contains $|R|$ real-valued variables y_i; we will see that the constraints enforce that $y_i = 1$ if and only if $i \in s(J)$. Finally, it contains $\mathcal{O}(|C||E|)$ real-valued variables z^s_{ji} that guard the identifiability of $G[J, s(J)]$. In the same spirit as for `MINSENSOR`, it is rather easy to develop an incremental cutting-plane approach also for `MINSOURCE`. Since this is quite straightforward, we do not show this approach here.

MILP 2 Complete formulation for `MINSOURCE`.

$$\text{minimize:} \quad \sum_{j \in C} x_j$$

subject to:

$$x_j \in \{0,1\} \qquad \forall j \in C \tag{5.3}$$

$$0 \leq y_i \leq 1 \qquad \forall i \in R \tag{5.4}$$

$$y_i \leq x_j \qquad \forall i \in R, \forall j \in N(i) \tag{5.5}$$

$$y_i \geq 1 - \sum_{j \in N(i)} (1 - x_j) \qquad \forall i \in R \tag{5.6}$$

$$z^s_{ji} \geq 0 \qquad \forall s \in C, \forall j \in C \setminus \{s\}, \forall i \in N(j) \setminus N(s) \tag{5.7}$$

$$z^s_{ji} \leq y_i \qquad \forall s \in C, \forall j \in C \setminus \{s\}, \forall i \in N(j) \setminus N(s) \tag{5.8}$$

$$\sum_{j \in N(i) \setminus \{s\}} z^s_{ji} \leq 1 \qquad \forall s \in C, \forall i \in R \setminus N(s) \tag{5.9}$$

$$\sum_{i \in N(j) \setminus N(s)} z^s_{ji} \leq 1 + (2 - x_s - x_j)(|N(j) \setminus N(s)| - 1) \qquad \forall s \in C, \forall j \in C \setminus \{s\} \tag{5.10}$$

$$\sum_{i \in N(j) \setminus N(s)} z^s_{ji} \geq x_s + x_j - 1 \qquad \forall s \in C, \forall j \in C \setminus \{s\} \tag{5.11}$$

Correctness (of MILP 2)**.** *A vector x defines a set $J \subseteq C$, such that $G[J, s(J)]$ is identifiable if and only if there exist vectors y and z such that (x, y, z) is a feasible solution of MILP 2.*

Proof. First we show that for any feasible (x, y), y is a 0/1 vector that "marks" the vertices of R that belong to $s(J)$. Inequalities (5.5) and (5.6) are equivalent to the logical statements (5.12) and (5.13), respectively [2].

$$(\exists j \in N(i) : j \notin J) \Rightarrow y_i \leq 0 \tag{5.12}$$

$$(\forall j \in N(i) : j \in J) \Rightarrow y_i \geq 1 \tag{5.13}$$

Combining (5.12), (5.13) and (5.4), we get that, for all $i \in R$, y_i behaves as a binary indicator variable: $y_i = 1$, if $i \in s(J)$ and $y_i = 0$, otherwise.

Now, we show that there exists a vector z such that (x, y, z) is feasible if and only if there exists a 0/1 vector $\widehat{z}$ such that $(x, y, \widehat{z})$ is feasible. The reverse direction is obvious; here we prove the forward direction. If (x, y, z) is feasible, then *both* x and y are 0/1 vectors and z lies in the polyhedron $P(x, y)$ defined by (5.7)-(5.11). With the same arguments as for the MILP of `MINSENSOR` (see Section 4.2), we can show that the matrix defining $P(x, y)$ is totally unimodular and, since both x and y are integer, the vertices of the polyhedron are integer. We take one vertex $\widehat{z}$ (that exists because the polyhedron is non-empty and bounded). Furthermore, (5.7) and (5.9) imply that $z^s_{ji} \in [0, 1]$; therefore $\widehat{z}$ is a 0/1 vector.

Inequalities (5.10) and (5.11) guarantee that $H = G[J, s(J)]$ is identifiable, namely that for all $s \in J$ and $j \in J \setminus \{s\}$ there exists a matching in $\widetilde{H}_s$ that covers j. If $x_s = x_j = 1$, that is both s and j are in J, these two inequalities imply the equality $\sum_{i \in N(j) \setminus N(s)} z^s_{ji} = 1$. This enforces that there exists a matching in $\widetilde{H}_s$ that covers j. Otherwise, if $x_s = 0$ or $x_j = 0$, these inequalities become degenerate and do not impose any constraints on the variables. □

[2] If the condition on left-hand side of the statement is true, then the inequality of the MILP reduces to the inequality on the right-hand side. Otherwise, the inequality reduces to a trivial one and does not impose any constraint on the variables.

Chapter 6

Modeling Uncertainty in the Network Structure

Summary of the chapter. When the structure of a source-sensor network is predicted with some uncertainty, it makes sense to study the robustness of identifiability. This is defined as the minimum number of edge modifications that are necessary for a graph to lose the property. We show that the computation of robustness reduces to the computation of surplus in bipartite graphs and, using an existing graph-theoretical result, we give a polynomial-time algorithm for this task. Then, by studying more closely the properties of the surplus function, we also present a polynomial-time algorithm for the computation of a tight set (this is a minimizer of the surplus function). To the best of our knowledge, there are no existing algorithms for the computation of surplus and tight sets in the literature. Therefore, the algorithms that we present here may be of independent interest also outside the context of identifiable graphs.

6.1 Robustness of identifiability in the unit-cost case

In many applications, the sets of sensors and sources are known exactly, but the structure of the bipartite network is predicted with some statistical or empirical method. For example, in the microarray setting, the network structure is predicted from the sequence similarities between probes and targets and, in this case, the threshold that separates binding from non-binding may be difficult to determine. Such a prediction inevitably involves some uncertainty, i.e., some source-sensor connections that have been predicted as significant may not exist in reality and, vice versa, some existing connections may have been missed by the prediction. Having that in mind, a natural question arises: How many prediction mistakes can a given bipartite network tolerate, before it loses the property of identifiability?

Definition 6.1. Let $G = (C, R; E)$ be an identifiable graph. For a vertex $x \in C$, we define its *robustness*, denoted by $\rho(x)$, as the minimum number of *edge addi-*

tions/deletions that are required to destroy the identifiability of x, i.e., to make $\nu(\widetilde{G}_x) < |C| - 1$. The robustness of the whole graph is $\rho(G) = \min_{x \in C} \rho(x)$.

In practice, this measure can be used in order to select among different sensor sets the one that gives rise to the "most identifiable" network. We can imagine an immediate generalization of `MINSENSOR`, where we replace the identifiability requirement with the requirement for a lower bound for the robustness.

Problem 11. `ROBUST-MINSENSOR`
Given is a bipartite graph $G = (C, R; E)$, an integer $\rho \geq 1$ and a function c that assigns a nonnegative rational cost to each vertex of R. We want to find a set $I \subseteq R$ such that the subgraph $G[C, I]$ has robustness at least ρ and $\sum_{y \in I} c(y)$ is minimized.

Clearly, the above problem is NP-hard, because for $\rho = 1$ it is reduced to `MINSENSOR`.

Topic for further research. An ILP formulation for `ROBUST-MINSENSOR` is an interesting thing to consider, although the robustness requirement seems harder to model with linear constraints than the identifiability requirement.

Given an identifiable graph $G = (C, R; E)$, we now set out to develop an algorithm for the computation of its robustness. To this end, we first show that computing the robustness reduces to computing the (non-negative) surpluses of the graphs $\widetilde{G}_x$, for all $x \in C$.

Proposition 6.2. *For every $x \in C$, $\rho(x) = \sigma(\widetilde{G}_x) + 1$.*

Proof. Consider a vertex $x \in C$. Adding (deleting) an edge incident to x, say (x, y), results in deleting (adding) the vertex y in $\widetilde{G}_x$ and this can only decrease (increase) $\nu(\widetilde{G}_x)$. Adding (deleting) an edge (z, y), where $z \neq x$ and $y \in N(x)$, has no influence on $\widetilde{G}_x$. Finally, adding (deleting) an edge (z, y), where $z \neq x$ and $y \notin N(x)$, results in adding (deleting) the edge (z, y) in $\widetilde{G}_x$ and this can only increase (decrease) $\nu(\widetilde{G}_x)$. Therefore, in order to decrease $\nu(\widetilde{G}_x)$, we must either add edges that are incident to x or delete edges that are incident to some non-neighbor of x. Since *both operations are allowed and have the same cost*, we can safely focus only on edge additions, because they correspond to vertex deletions in $\widetilde{G}_x$. More specifically, $\rho(x)$ equals the minimum number of vertices that we must delete from $R \setminus N(x)$, such that the matching number of the remaining subgraph of $\widetilde{G}_x$ becomes less than $|C|-1$. By Hall's marriage theorem, this can only be achieved by choosing a nonempty set $Y \subseteq C \setminus \{x\}$ and deleting $|N_{\widetilde{G}_x}(Y)| - |Y| + 1$ vertices from $N_{\widetilde{G}_x}(Y)$. Therefore, $\rho(x) = \min_{\emptyset \neq Y \subseteq C \setminus \{x\}} \{|N_{\widetilde{G}_x}(Y)| - |Y| + 1\} = \sigma(\widetilde{G}_x) + 1$. □

Remark 6.3. Observe that $\nu(\widetilde{G}_x)$ is decreasing with respect to the addition of edges that are incident to x and increasing with respect to the addition of non-incident edges. This shows, in turn, that the identifiability of G is not a monotone property with respect to edge addition; in fact, it is not even convex. A property is called monotone with respect to edge additions if: $G = (V, E_1)$ having the property implies that, for all $E \supseteq E_1$, $G = (V, E)$ also has the property. A property is called convex if $G = (V, E_1)$ and $G = (V, E_2)$ having the property implies that, for all E such that $E_1 \subseteq E \subseteq E_2$, $G = (V, E)$ also has the property.

From Proposition 6.2, any upper bound for the surplus $\sigma(\widetilde{G}_x)$, immediately gives an upper bound for the robustness of a vertex $x \in C$. Recall that $\widetilde{G}_x$ is the subgraph that arises from G by removing x and its neighborhood. For example, we have the following straightforward bounds:

1. $\sigma(\widetilde{G}_x) \leq \sigma_{\widetilde{G}_x}(C \setminus \{x\}) = |N_{\widetilde{G}_x}(C \setminus \{x\})| - |C \setminus \{x\})| = |R| - d(x) - (|C| - 1)$. Therefore, $\rho(x) \leq |R| - |C| + 2 - d(x)$.

2. For all $y \in C \setminus \{x\}$, we have $\sigma(\widetilde{G}_x) \leq \sigma_{\widetilde{G}_x}(\{y\}) = |N(y) \setminus N(x)| - 1$. Therefore, $\rho(x) \leq \min_{y \in C \setminus \{x\}} |N(y) \setminus N(x)|$.

In general, these two bounds can be very loose, as shown in the following examples.

1. For every $d \geq 1$, there exists an identifiable graph $G = (C, R; E)$ such that for every $x \in C$, $\rho(x) = 1$, while $|R| - |C| + 2 - d(x) \geq d$. For example, such a graph is obtained by taking the disjoint union of four stars with centers in C, two of which are of degree d and the remaining two of degree 1.

2. For every $d \geq 1$, there exists an identifiable graph $G = (C, R; E)$ such that for every $x \in C$, $\rho(x) = 1$, while $\min_{y \in C \setminus \{x\}} |N(y) \setminus N(x)| = d$. For example, such a graph is obtained as follows: Take $d + 1$ disjoint stars with d leaves (star centers are in C and leaves are in R) and then take G to be the bipartite complement of this graph. By construction, $\min_{y \in C \setminus \{x\}} |N(y) \setminus N(x)| = d$. For every $x \in C$, the graph $\widetilde{G}_x$ is a complete bipartite graph with d vertices on each side and, therefore, $\sigma(\widetilde{G}_x) = 0$. So, by Proposition 6.2 we conclude that $\rho(x) = 1$.

In summary, Proposition 6.2 shows that the problem of computing $\rho(G)$ reduces to computing the surplus of the (nonnegative surplus) graphs $\widetilde{G}_x$, for all $x \in C$. The proof also shows that, in order to destroy the identifiability of $x \in C$ with a minimum number of edge modifications, it is enough to find a tight set X in $\widetilde{G}_x$ and then add to G all the edges $\{(x, y) : y \in N_{\widetilde{G}_x}(X)\}$. But how do we actually compute the surplus and find a tight set in a bipartite graph? In the following section we present polynomial-time algorithms for these two tasks.

6.2 Computing the surplus and finding tight sets

The results of this section apply to arbitrary bipartite graphs and might be of *independent interest* also outside the context of robustness. Algorithm 5 below computes the surplus $\sigma(G)$ of a bipartite graph $G = (L, R; E)$. Its correctness is based on the following result from the book of Lovász and Plummer (Theorems 1.3.1 and 1.3.6 in [LP86]).

Lemma 6.4. *Let $G = (L, R; E)$ be a bipartite graph. If $\sigma(G) < 0$, then $\sigma(G) = \nu(G) - |L|$. If $\sigma(G) \geq 0$, then $\sigma(G)$ equals the largest integer s satisfying the following property, for every $x \in L$: if we add s new vertices to L and connect them to all neighbors of x, the resulting graph has non-negative surplus.*

For the algorithm's implementation Berge's theorem (see Theorem 3.8) again proves to be very useful.

Algorithm 5 Computation of the surplus in $G = (L, R; E)$

```
1: Compute a maximum matching $M$ in $G$.
2: if $|M| = |L|$   (i.e., $\sigma(G) \geq 0$) then
3:   for all $x \in L$ do
4:     $s_x \leftarrow 0$, $M_x \leftarrow M$, $G_x \leftarrow G$
5:     repeat
6:         $G_x \leftarrow$ the graph obtained from $G_x$ by adding to it a new vertex $v^*$ and
       connecting it to all neighbors of $x$
7:         if $G_x$ has an $M_x$-augmenting path $P$ then
8:             $M_x \leftarrow (M_x \cup P) \setminus (M_x \cap P)$ and $s_x \leftarrow s_x + 1$
9:         else exit the repeat loop
10:        end if
11:    end repeat
12:  end for
13:  $\sigma(G) \leftarrow \min_{x \in L} s_x$
14: else
15:  $\sigma(G) \leftarrow |M| - |L|$
16: end if
17: return $\sigma(G)$
```

Proposition 6.5. *Algorithm 5 computes the surplus of a bipartite graph $G = (L, R; E)$. It can be implemented so that it runs in time $\mathcal{O}\left(|E|(\sqrt{|L| + |R|} + |L| + |E|)\right)$ (which is $\mathcal{O}(|E|^2)$ if G has no isolated vertices).*

Proof. By Hall's Theorem, the augmented graph G_x (line 6) has non-negative surplus if and only if there exists a matching that covers all vertices of $L_x \cup \{v^*\}$. By Berge's

Theorem this happens if and only if there exists an M_x-augmenting path (which has to start at v^*). Finally, the correctness of the algorithm follows from Lemma 6.4.

It remains to analyze the running time. The computation of a maximum matching in line 1 can be done in $\mathcal{O}\left(|E|\sqrt{|L|+|R|}\right)$ time, using, e.g., the algorithm by Hopcroft and Karp [HK73]. For each $x \in L$, the internal repeat-loop can be executed up to $d(x)$ times and, therefore, $|E(G_x)| \leq |E| + (d(x))^2$. In line 7, checking if G_x has an M_x-augmenting path and if yes, finding one, can be done as follows: First we orient all unmatched edges in G_x from L to R and all matched edges from R to L and then we look for a directed path from v^* to an unmatched vertex. This can be done with breadth-first search in $\mathcal{O}(|E(G_x)|)$ time. In line 8, the symmetric difference can be computed in $\mathcal{O}(|E(G_x)|)$ time. So, the total running time is in $\mathcal{O}\left(|E|\sqrt{|L|+|R|} + \sum_{x\in L}(|E| + (d(x))^2)\right) \subseteq \mathcal{O}\left(|E|\sqrt{|L|+|R|} + |L||E| + |E|^2\right)$. The last inclusion follows from: $\sum_{x\in L}(d(x))^2 \leq \left(\sum_{x\in L} d(x)\right)^2 = |E|^2$. $\square$

Remark 6.6. The algorithm for computing the surplus can also be used to compute the *weighted surplus* $\sigma_w(G) = \min_{\emptyset\neq X\subseteq A}(w(N(X)) - w(X))$, where each vertex v is assigned a positive integer weight $w(v)$ and $w(S) = \sum_{v\in S} w(v)$, for all $S \subseteq A \cup B$. It is easy to verify that the following (pseudo-polynomial) construction reduces the problem to the unweighted case: Split each vertex v into $w(v)$ vertices and replace an edge (x, y) by a complete bipartite graph $K_{w(x),w(y)}$.

In a practical setting, apart from computing the value of the surplus, we would also like to find a tight set of G. In the context of the robustness computation, this corresponds to finding a set of edge additions that destroy the identifiability of a given graph. As we show next, we can find a tight set of G by using an algorithm for surplus computation as a black-box routine in a greedy fashion. First we need some preliminary technical facts.

Proposition 6.7. *Let $G = (A, B; E)$ be a bipartite graph and $A' \subset A$. We denote with $G - A'$ the subgraph induced by $(A \setminus A') \cup B$.*

1. *For all $X \subseteq A \setminus A'$, $\sigma_{G-A'}(X) = \sigma_G(X)$. Therefore, we will omit the index and write $\sigma(X) := \sigma_{G-A'}(X) = \sigma_G(X)$.*
2. *$\sigma(G - A') \geq \sigma(G)$.*
3. *If $\sigma(G - A') = \sigma(G)$, then, for all $A'' \subseteq A'$, any tight set of $G - A''$ is also a tight set of G. In particular, it follows that $\sigma(G - A'') = \sigma(G)$.*
4. *Let $x \in A$ such that $\sigma(G - \{x\}) = \sigma(G)$. Any tight set of $G - x$ is also a tight set of G.*
5. *$\sigma(G - \{x\}) > \sigma(G)$ for all $x \in A$, if and only if A is the only tight set of G.*

Proof.

1. The neighborhood of any set $X \subseteq A \setminus A'$, is the same in G and in $G - A'$. The claim follows from the definition $\sigma(X) = |N(X)| - |X|$.
2. $\sigma(G - A') = \min_{X \subseteq A \setminus A'}\{\sigma(X)\} \geq \min_{X \subseteq A}\{\sigma(X)\} = \sigma(G)$. The inequality follows from the fact that we are minimizing the same objective function over a larger ground set.
3. Let $X^* \subseteq A \setminus A''$ be a tight set of $G - A''$. So, we have $\sigma(X^*) = \min_{X \subseteq A \setminus A''}\{\sigma(X)\} \leq \min_{X \subseteq A \setminus A'}\{\sigma(X)\} = \min_{X \subseteq A}\{\sigma(X)\}$. The inequality follows from the fact that $A \setminus A'' \supseteq A \setminus A'$ and the last equality follows from the hypothesis $\sigma(G - A') = \sigma(G)$. So, we finally have $\sigma(X^*) \leq \min_{X \subseteq A}\{\sigma(X)\}$ and equality follows. This shows that X^* is tight in G.
4. This is a special case of item 3 for $A'' = A' = \{x\}$.
5. Forward: For the sake of contradiction, assume that there exists a tight set $Y \subset A$. Then, Y is also tight in $G - \{x\}$, where $x \in A \setminus Y$. That is, $\sigma(Y) = \sigma(G) = \sigma(G - \{x\})$, which contradicts the hypothesis.
   Reverse: For the sake of contradiction, assume that there exists $x \in A$ with $\sigma(G - \{x\}) = \sigma(G)$ (it cannot be $\sigma(G - \{x\}) < \sigma(G)$ due to item 2). Then, there exists a set $Y \subseteq A \setminus \{x\}$ that is tight in both $G - \{x\}$ and G, and this contradicts the hypothesis.

□

Algorithm 6 computes a tight set of a bipartite graph $G = (A, B; E)$. It requires $|A|$ iterations, each one containing a surplus computation that can be done in $\mathcal{O}\left(|A|^{2.5}|B|^{2.5}\right)$ time with Algorithm 5. Therefore, the total running time of the algorithm is bounded from above by $\mathcal{O}\left(|A|^{3.5}|B|^{2.5}\right)$.

Algorithm 6 Finding a tight set X

```
1: $X \leftarrow A$
2: for all $x \in A$ do
3:     if $\sigma(G - \{x\}) = \sigma(G)$ then
4:         $G \leftarrow G - \{x\}$
5:         $X \leftarrow X \setminus \{x\}$
6:     end if
7: end for
8: return $X$
```

Below we prove the correctness of Algorithm 6.

Proposition 6.8. *Let $H = (A_H, B_H; E'_H)$ be a bipartite graph. Let $v^* \in A_H$ such that $\sigma(H - v^*) > \sigma(H)$. If $A' = \{v_1, \ldots, v_i\} \subset A_H$ and $\sigma(H - \{v_1, \ldots, v_{j-1}\}) = \sigma(H - \{v_1, \ldots, v_j\})$ for all $j = 1, \ldots, i$, then $\sigma((H - A') - v^*) > \sigma(H - A')$.*

Proof. Assume that it is not true. By item 2 of Proposition 6.7, we have $\sigma((H - A') - v^*) = \sigma(H - A')$. By hypothesis, $\sigma(H - A') = \sigma(H - \{v_1, \ldots, v_i\}) = \cdots = \sigma(H)$. Then, $\sigma((H - A') - v^*) = \sigma(H)$. Using item 3 of Proposition 6.7, we get that $\sigma(H - v^*) = \sigma(H)$, a contradiction with the hypothesis. □

Proposition 6.9. *The set X output by Algorithm 6 is tight.*

Proof. If $X = A$, the statement follows by item 5 of Proposition 6.7. So assume that $\overline{X} = \{v_{i_1}, \ldots, v_{i_k}\}$ for some $k \geq 1$, and these vertices are deleted in this order by the algorithm. Denote by G' the subgraph induced by $X \cup B$. Then, it is enough to show the following claim.

Claim. For every $x \in X$, $\sigma(G' - x) > \sigma(G')$.

Assume that the claim holds. Then, item 5 of Proposition 6.7 implies that X is the only tight set of G'. Applying inductively item 4 of Proposition 6.7 on the vertices deleted by the algorithm (in reverse order), we deduce that X is a tight set of G.

Proof of Claim. Let $x \in X$. Let v_{i_j} for $j \leq k$ be the last vertex deleted by the algorithm before x is encountered by the algorithm (or $j = 0$ if there is no such vertex). If $j = k$, the inequality of the claim follows by the algorithm's rule. So, assume $j < k$. Let $G'' = G - \{v_{i_1}, \ldots, v_{i_j}\}$. The algorithm's rule implies that $\sigma(G'' - x) > \sigma(G'')$. Now, we apply Proposition 6.8 with $H = G''$, $v^* = x$ and $A' = \{v_{i_{j+1}}, \ldots, v_{i_k}\}$. We have $G'' - A' = G'$ (the subgraph induced by $X \cup B$) and $\sigma(G' - x) = \sigma((G'' - A') - x) > \sigma(G'' - A') = \sigma(G')$, proving inequality $\sigma(G' - x) > \sigma(G')$. This completes the proof of the claim and with it the proof of the proposition. □

Remark 6.10. The surplus and a tight set can also be computed by minimizing $|A|$ submodular functions of the form $f_x(X) = \sigma_{\widetilde{G}_x}(X)$ (over all $X \subseteq A - \{x\}$), for all $x \in A$ (we omit the details). For instance, using as a black-box the algorithm for submodular function minimization by Iwata [Iwa03], or the one by Orlin [Orl07], we can compute the surplus and find a tight set in time $\mathcal{O}(|A|^6|B|\log|B|)$ or $\mathcal{O}(|A|^7|B|)$, respectively. If $|B|$ is considerably bigger than $|A|$, then this approach is faster than the simple algorithms proposed above.

6.3 On the robustness of identifiability in the weighted case

Let us recall our motivation for studying the robustness of identifiable graphs: The sets of sensors and sources are known exactly and the structure of the bipartite network is predicted with some statistical method. A natural generalization of this scenario occurs when the prediction for each edge and non-edge comes together with a reliability score. In this section, we sketch some ideas for modeling this case and formalize two combinatorial problems that naturally appear.

There are two ways to model robustness in this setting. First, we can look for the least reliable subset of edges and non-edges whose flipping destroys the identifiability of the network. This results in the notion of weighted robustness, whose computation is reduced to the following problem.

Problem 12. Given a bipartite graph $G = (A, B; E)$, a cost $c_e(e)$ for each edge $e \in E$ and a cost $c_v(v)$ for each vertex $v \in B$, find a subset $B' \subseteq B$ and a subset $E' \subseteq E$, such that $G - B' - E'$ has no A-perfect matching and the cost $\sum_{e \in E'} c_e(e) + \sum_{v \in B'} c_v(v)$ is minimal.

Another possibility for modeling robustness is to set two thresholds on the reliability scores and characterize the edges as existing (if the score is above both thresholds), non-existing (if the score is below both thresholds), or undetermined (otherwise). This gives rise to the following sandwich problem:

Problem 13. Given two bipartite graphs $G_1 = (A, B; E_1)$ and $G_2 = (A, B; E_2)$ such that $E_1 \subseteq E_2$, is there an edge set E such that $E_1 \subseteq E \subseteq E_2$ and the graph $G = (A, B; E)$ is identifiable?

For a nice discussion about sandwich problems for various graph properties (some of them also relevant to computational biology), we refer to [GKS95].

Topic for further research. To the best of our knowledge, both problems defined above are open in terms of complexity. As far as Problem 12 is concerned, the generalization where an upper bound K for the cardinality of a maximum matching in $G - B' - E'$ is given as part of the input, is NP-hard, even for unit costs [1].

[1]Personal communication with Jerôme Monnot.

Chapter 7

Modeling cross-hybridization in microarray experiments

Summary of the chapter. The analysis of microarray data under the presence of non-specific probes can be abstracted as a signal separation problem in a source-sensor network. Thereby, we regard the mRNA molecules as signal sources, the probes on the microarray plate as sensors and the connectivity of the network can be predicted from the known probe and target sequences. Here, we present a case study of the matrix factorization approach on a real-world dataset. This was obtained with a custom microarray that was designed for the study of alternative splicing events. Although no training set is available for extensive cross-validation, we empirically observe an agreement with some (very few) experimentally verified test cases and the results from another, completely different, computational approach.

7.1 Technological motivation

During the last fifteen years, microarrays have become an indispensable tool in genome research, because they offer the possibility to monitor the expression of a large number of genes simultaneously. The processing of the massive amount of data generated by a microarray experiment poses many statistical challenges. However, if carried out successfully, it opens many possibilities for the comparison of gene transcription profiles between different kinds of cells.

From a simplifying point of view, a microarray experiment consists of three basic components: a small-scale rectangular grid (microarray plate), a set of short DNA sequences (probes) that are fixed on the grid, and a mixture of mRNA sequences (targets) that are labeled with a fluorescent dye. The goal is to measure the quantities of the targets by taking advantage of the high specificity of DNA hybridization. During an experiment, the labeled targets are allowed to hybridize on the surface of the microarray; whenever a target comes across a complementary probe it will bind onto it. Then, the microarray plate is scanned and the light intensity on each one of the probe spots is measured. Finally, the light intensities are used to infer the quantities of the targets.

A design principle that lies in the heart of a typical microarray experiment is probe specificity, i.e., the requirement that each probe should be complementary to exactly one target. Due to this requirement, the task of designing probes for a given set of transcripts is complicated enough to deserve extensive study [Rah04]. Lack of specificity, also known as cross-hybridization, means that the light intensity measured on a probe cannot be attributed any more to the presence of a single target, but is rather a mixture due to multiple targets. This complication makes the demultiplexing of the probe signals significantly more difficult. Therefore, almost all computational methods for low-level processing of microarray data rely on the fact that the probes are almost perfectly specific by design. So, they focus on the remaining statistical challenges due to the many sources of variation that are inherent in a series of microarray experiments.

Cross-hybridization is usually considered a side-effect, whose contribution to the measured signal is almost negligible and can be either quantified by ad-hoc methods or absorbed by the stochastic components of the various models. For example, on Affymetrix chips each probe has a counterpart, called mismatch probe, that differs in exactly one nucleotide in the middle of the probe sequence. This mismatch probe does not perfectly bind to the intended target and its purpose is to capture the cross-hybridization signal and thus contribute to a more accurate data analysis. However, in the meanwhile many questions have been posed concerning the utility of the mismatch probes, e.g, when it was observed that almost 20% of them demonstrate higher intensities than the corresponding perfect-match probes [WMP+07]. As it can be seen from the plethora of published methods, the task of inferring the target quantities from the probe intensities is far from trivial, even without taking into account the complications that arise from cross-hybridizing probes. In fact, quite elaborate statistical methods have been proposed, in order to compare gene expression across different samples in a statistically rigorous way; for an overview of this huge field we refer to the book by Wit and McClure [WM04].

Although absolutely desirable, perfect probe specificity cannot be always achieved. We consider, for example, the use of microarrays for the quantitative study of alternative splicing, where the goal is to measure the relative amounts of different splice variants in different tissues [LR04]. For this application, it is important to have probes throughout all regions of the gene: standard probes designed to match the exons and also probes that are designed to match the exon-exon junction that might be brought together by an alternative splicing event. Since different splice variants might share some exons, some of the probes will be non-specific and the respective analysis methods will have to explicitly deal with cross-hybridization. Wang *et al.* [WHH+03] take a matrix factorization approach to deal with this problem, similar to the one that we describe below. They empirically observe that the approach works well in practice, but lacking the notion of identifiability they do not attempt to explain its success theoretically. In Section 7.2 we also present a case study, using the data from a microarray platform that was designed for the study of alternative splicing.

Non-specific probes are also unavoidable when we want to measure the expression of transcripts that exhibit very high sequence similarity. For example, microarrays that use species-specific probes can contribute to high-throughput composition analysis of microbial communities. However, it is rather difficult to design highly specific probes, firstly because most of the genomes in an environmental community are unknown and secondly, because many of them have very similar sequences, e.g., at the organism level [MRM+07]. Therefore, apart from very careful probe design, in such an application it is also necessary to take into account the effect of cross-hybridization.

But how is the modeling of cross-hybridization connected to the generic `CAMF` problem that we introduced in Chapter 2? We can think of a microarray as a multiple-input, multiple-output system, where the input signals are the target concentrations and the output signals are the light intensities measured on the probes. At least for some reasonable ranges of target concentrations it has been experimentally observed that the output signals depend linearly on the input signals [LW01]. In the following, we assume that normalization and optical background adjustment have been carried out succesfully and, if necessary, some linearizing transformation, e.g. logarithmic, has been applied.

Let us consider an experiment with n probes, m targets and k samples and define the following real-valued matrices. Matrix $A = (a_{ij})$ has dimensions $n \times m$ and a_{ij} is the affinity coefficient of the i-th probe for the j-th target. Matrix $X = (x_{ij})$ has dimensions $m \times k$ and x_{ij} is the concentration of the i-th target at the j-th sample. Finally, $Y = (y_{ij})$ has dimensions $n \times k$ and y_{ij} is the light intensity measured on the i-th probe at the j-th sample. Assuming linearity and that background has been removed (i.e., the measured probe intensities are only due to the presence of the targets) it should hold $Y \approx AX$. In general, we do not expect all probes to bind to all targets, but only the pairs that demonstrate a high enough sequence complementarity. In this way, from the pairwise sequence similarities between the probes and the targets we can predict (with some uncertainty) the zero pattern of A.

For the non-zero affinity coefficients it is hard to obtain exact values, because this would require an accurate modeling of the hybridization process on the microarary plate, which is notoriously difficult. In fact, there is a whole community studying the hybridization kinetics on a microarray plate using elaborate physicochemical models; for a recent example see [OSF+08]. However, at a more elementary level we may be able to derive a rough estimation for the affinity coefficients from the knowledge of the probe and target sequences. Systematic ways to do so have been studied in [CCL+06], where several sequence-based features have been considered as potential predictors of the affinity coefficients.

In practice, we measure the intensity matrix Y and in order to draw biological conclusions, our primary goal is to estimate X. In fact, we usually do not need absolute values for the entries of X, but it suffices to compute changes of concentration with respect to a reference sample (i.e. column of X). Thereby, the affinity coefficents A are unknown, but as we described above, it may be possible to derive some constraints for them. Moreover, the entries of X must be nonnegative. Having said that, it is clear that the problem of "decoding" the probe signals reduces to a constrained matrix factorization problem, as we described it in Chapter 2.

7.2 A case study for alternative splicing arrays

As a case study, we use the matrix factorization framework described in Chapter 2 in order to process a publicly available microarray dataset. This dataset was generated by a microarray platform that was specifically designed to perform quantitative analysis of tissue specific alternative splicing events (AS events) [PSM+04]. More specifically, Pan *et al.* have developed a custom microarray to represent sequence validated AS events that were mined from mouse expressed sequence tags databases. In particular, they selected 3126 simple AS events that involve only three exons and two alternative splice variants, as shown in Figure 7.1. We refer to the splice variant that contains exon B as *included* and to the one that skips it as *excluded.* The concentrations of the two variants were monitored across ten different tissue samples by six oligonucleotide probes. They were designed as shown in the figure: one body probe for each exon and one junction probe for each one of the three splice junctions. In order to analyze the measurements produced by this microarray platform, Shai *et al.* [SMBF06] developed an algorithm (called GenASAP) that is based on a generative probabilistic model. Later we compare our results with the ones from GenASAP and observe a significant agreement, although the two computational methods are different in nature.

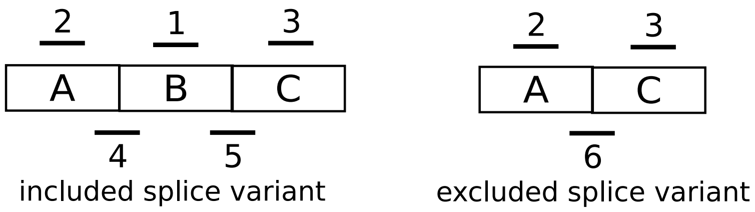


Figure 7.1: Probe design for quantification of alternative splicing. The junction probes bind perfectly to the one variant and partially to the other.

We downloaded the probe intensity data from the Gene Expression Omnibus (GEO) database [BTW+09] and we fit a bilinear model using Algorithm 1. In this particular application, the affinity matrix A has dimensions 6×2, where the rows correspond to probes (as they are numbered in Figure 7.1) and the columns to targets: the first column corresponds to the included and the second column to the excluded splice variant. The concentration matrix X has dimensions 2×10. From our empirical understanding of the physical system we derive the following constraints for the matrices A and X:

1. $A, X \geq 0$.
2. Probe 1 does not bind at all to excluded target: $a_{12} = 0$.
3. Probe 2 binds equally strongly to both targets: $a_{21} \approx a_{22}$.
4. Probe 3 binds equally strongly to both targets: $a_{31} \approx a_{32}$.
5. Probe 4 binds fully to included and only partially to excluded target: $a_{42} < a_{41}$.
6. Probe 5 binds fully to included and only partially to excluded target: $a_{52} < a_{51}$.
7. Probe 6 binds fully to excluded and only partially to included target: $a_{62} > a_{61}$.

We have no a-priori knowledge about the order of magnitude of the affinity coefficients, so, in order to express constraint 3, we bound the relative difference of the coefficients (the bounds are chosen empirically). In the same way, we express constraint 4.

$$0.9 \leq \frac{a_{21}}{a_{22}} \leq 1.1$$

We have only one fixed zero position in A, namely $a_{12} = 0$. According to the discussion of Chapter 3, this means that only the second column of A is identifiable. Let us recapitulate what this means: If we consider two solutions of the factorization problem $Y \approx AX = \widehat{A}\widehat{X}$, then it holds $\widehat{A} = AR$ and $\widehat{X} = R^{-1}X$, where both R and R^{-1} satisfy the following zero pattern.

$$R, R^{-1} \lhd \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Therefore, among equally good factorizations of Y, the second column of A and the first row of X are uniquely determined up to scaling.

We implemented Algorithm 1 in Matlab, using the CVX library [GB08] to solve the quadratic programming subproblems. This is a particularly convenient software for convex optimization that allows constraints and objectives to be specified using standard Matlab expression syntax. CVX first checks if the input problem satisfies some formal convexity requirements and, if yes, automatically transforms it to standard form. Then, it calls a suitable external optimization engine and finally returns the solution in the form of standard Matlab variables.

In total there are 3126 AS events available. For each one of them, we run Algorithm 1 for 20 different random initializations of the A matrix and we keep as final solution the (A, X) pair that achieves the best fit to the measured probe intensities Y. As a

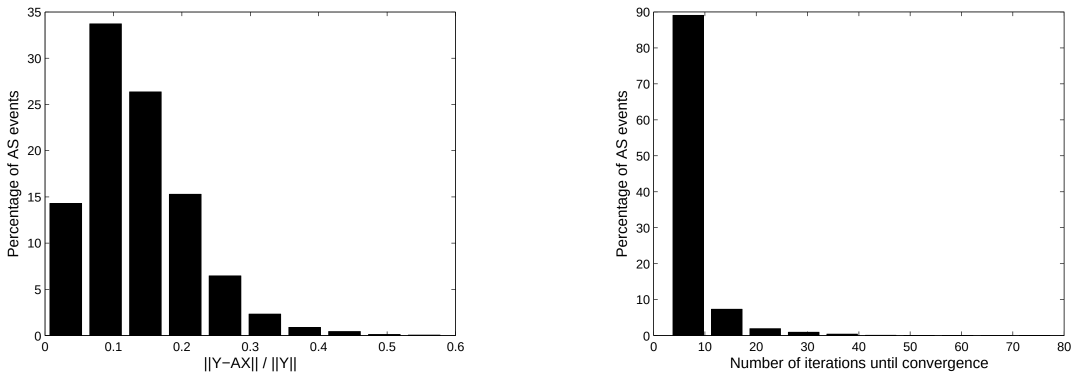


Figure 7.2: Some diagnostics for the alternating minimization. Left: histogram of the fit ratio. Right: histogram of the number of iteratons until convergence.

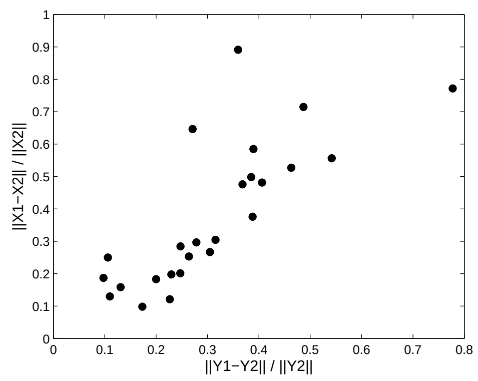
convergence criterion we check if the relative decrease of the objective function between two subsequent iterations falls below 0.5%. Figure 7.2 illustrates some diagnostics about the performance of the minimization: A histogram of the fit ratio $\frac{\|Y-AX\|_F}{\|Y\|_F}$ and a histogram of the number of iterations until convergence.

Let us recall that our whole discussion about identifiability is based on the assumption that an (A, X) solution of `CAMF` is not rank-degenerate; in our case this means $\text{rank}(A) = \text{rank}(X) = 2$. For a particular solution (A, X), it is important to check a-posteriori if this assumption is satisfied. In our computational experiments, we observed that this is always the case, which validates our intuition: Rank-degeneracies, although certainly possible in theory, should be rare in practical applications, where the numbers represent physical quantities.

We now make an attempt to validate our results, using 25 duplicate AS events that the designers have included on the microarray. For each one of them there are two different probe sets that are designed to hit different parts of the target sequences. We compare the values of the measured intensities Y between the two duplicates and we do the same for the estimated affinities and target concentrations, A and X. More specifically, for the comparison we use the relative difference

$$dY = \frac{\|Y_1 - Y_2\|_F}{\|Y_1\|_F}$$

where Y_1 and Y_2 are the measurements for the first and second duplicate. Similarly, we define dA and dX. In Figure 7.3 we plot dY versus dX and dA. We observe that, for all 25 duplicate AS events, $dY \leq 0.8$ and if we are willing to consider one measurement as outlier, then $dY \leq 0.6$. In order to get an empirical p-value, we calculate dY for ten thousand randomly selected probeset pairs from the microarray: only 0.5% of them have $dY \leq 0.6$ and 7% have $dY \leq 0.8$. This provides some evidence that the



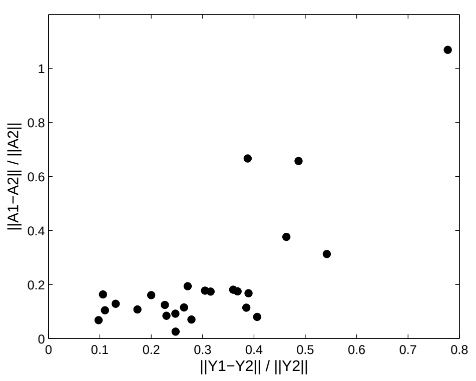


Figure 7.3: Using the duplicate AS events for validation. Left: dY versus dX. Right: dY versus dA.

experimental setting is consistent, i.e., that the same AS event, when measured with different probes, gives rise to probe intensities that do not differ much.

A second observation from the plots gives a hint for the consistency of the computational method. Namely, in the dY-dX plot we see that the "closer to each other" are the Y-matrices, the closer to each other are also the X-solutions (we observe an almost linear dependence). As far as the A-solutions are concerned, their difference does not depend on dY, for $dY \leq 0.4$. This might be attributed to the fact that many more constraints are imposed on matrix A than on X.

In order to assess the accuracy of the method, we would ideally need a spike-in dataset, with known values for both the target concentrations X and the probe intensities Y. Unfortunately, for this particular platform such a dataset is not available. On the other hand, in [PSM$^+$04] the authors mention that they performed real-time PCR (RT-PCR) experiments for 20 AS events, in order to verify the performance of their computational approach GenASAP. From the supplementary material of the paper, we were able to extract the RT-PCR measurements for 9 of these events [1] and in the following discussion we use them as "training set". Clearly, the size of this training set is very small, but this is the best we could do in this case. As a second best option, we compare our results to the ones obtained with GenASAP (that are provided as supplementary material in the authors' website).

There is also another complication. For the AS events that were experimentally measured and for the results of GenASAP, we do not have the values for the entries of the X matrix, but rather a different quantity that seems to be of more interest for the authors. Given the 2×10 matrix X, we define the vector $q(X)$ to contain the relative abundances of the excluded variant to the abundances of the included variant across the tissues, i.e., $q(X)_i = \frac{X_{2i}}{X_{1i}}$, for $i = 1, \ldots, 10$. So, for the experimentally verified

[1]We asked from the authors the measurements for all 20 events, but we received no answer.

AS events and for the GenASAP predictions, we have available the q-vector for each monitored AS event.

We have already seen that the best we can do with the matrix factorization framework is to determine A and X up to diagonal scaling and this is the case (only) if the zero pattern of A is identifiable. Therefore, assuming that there exists a "true" concentration matrix X_t, the computed X_c solution will be, in the best case, a scaled version of X_t: $X_c = RX_t$, for a diagonal R. In that case, $q(X_c)$ and $q(X_t)$ will be collinear: $q(X_c) = \frac{r_{22}}{r_{11}} q(X_t)$. Therefore, it makes sense to compare the q vectors using a statistical measure that captures linear relationships; the Pearson correlation coefficient is perfectly suited for that purpose. Although in our case the zero pattern of A is not identifiable, we still use Pearson correlation as an empirical measure of agreement between two q-vectors.

Now we compare the correlation between our results and the RT-PCR measurements. The correlations for the nine measured AS events are: 0.07, 0.86, 0.96, 0.98, 0.87, 0.98, 0.77, 0.82, 0.94. Only one out of nine is low and this is at first approximation a good sign. In Figure 7.4 we compare our predictions to the predictions of GenASAP.

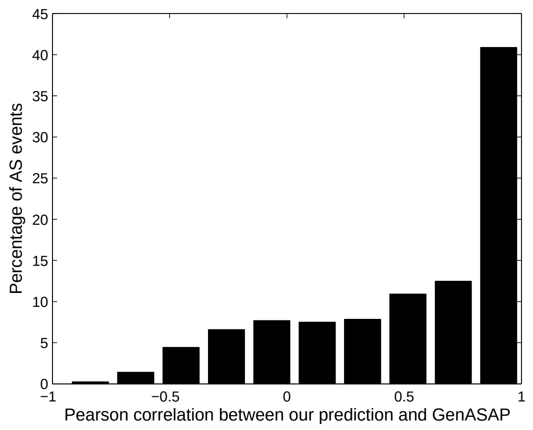


Figure 7.4: q(X): correlation between our prediction and GeneASAP.

The main obstacle for a thorough evaluation of this approach is the lack of a training set of high quality. If this was available, we would be able to try out several learning approaches. For example, we could parametrize the physical constraints 5, 6 and 7 with a shrinkage parameter $\lambda < 1$: $a_{42} \leq \lambda a_{41}$, etc. Then, we could sample different values for λ, in order to get the best prediction performance on the training set. In any case, the formalism of identifiable graphs would play an important role for a better interpretation of the factorization results.

Chapter 8

Estimating transcription factor activities in regulatory networks

Summary of the chapter. The source-sensor abstraction can also model simple bipartite regulatory networks. In this application, the signal sources are the transcription factors, the sensors are the regulated genes and the task is to infer the activities of the transcription factors from the gene expression profiles. The matrix factorization approach applied on this problem has been known for a while in the bioinformatics community under the name of Network Component Analysis (NCA). Here, we study the regulatory networks of *E. coli* and *S. cerevisiae* from a structural point of view. In particular, following the discussion of Chapter 5, we want to find out how hard it is to isolate groups of sources that can be monitored independently from the rest. We observe that some simple data reduction rules are very effective on these instances. In this way, we come up with a simple deterministic solution to the problem of subnetwork selection, while the creators of NCA solve it in their papers with an ad-hoc randomized method.

8.1 Biological motivation

A gene regulatory network is a collection of DNA molecules in a cell, which interact with each other through their RNA and protein expression products. In general, each gene is transcribed to an mRNA molecule, which is in turn translated to a protein. The proteins are the building blocks of the cell, having a multitude of functions, e.g., structural or enzymatic. Some proteins, called transcription factors (TF), serve a special purpose: By binding to the promoter regions at the start of other genes, they either activate or deactivate their transcription. In this way, the transcription factors control the production rate of other proteins and this property makes them the main players in regulatory networks. A key aspect of systems biology is to develop mathematical models for describing the dynamics of regulatory networks.

Many existing methods that use gene expression data to study regulatory networks assume that the quantity of a TF that interacts with promoters of other genes is the same as the measured mRNA level of the TF. However, this assumption is not accurate for most biological systems, since the amount of a TF that actively participates in DNA binding (called TF activity) is only a fraction of its total quantity. In practice, the TF activities are difficult to measure experimentally, especially when a TF is regulating more than one gene. However, as we describe below, it is possible to reconstruct unobserved TF activities from the expression profiles of target genes.

In some cases, the regulatory relationships between TFs and regulated genes can be represented with a bipartite network $G = (C, R; E)$, where C is the set of TFs, R is the set of genes and E are the regulation pairs. The assumption that G is bipartite excludes the existence of feedback loops or self-regulation. For the networks of simple organisms this assumption is not too restrictive, because large parts of these networks indeed have such a simple bipartite structure. The edges that connect a TF to the regulated genes have weights that indicate the strength of regulation. These weights quantify the relative contribution of the active TF quantity to the control of different genes.

In order to reconstruct the unobserved TF activities, we approximate the relationship between them and gene expression levels with a log-linear model; such models are used in several disciplines as a standard tool to approximate nonlinear systems. For the physical reasoning behind this equation see [LBY$^+$03].

$$\frac{\widehat{y}_i(t)}{\widehat{y}_i(0)} = \prod_{j=1}^{m} \left(\frac{\widehat{x}_j(t)}{\widehat{x}_j(0)} \right)^{a_{ij}} \tag{8.1}$$

In Equation (8.1), $\widehat{y}_i(t)$ is the expression level of the i-th gene at time point t, $\widehat{x}_j(t)$ is the activity of the j-th TF at time point t and a_{ij} is the regulation strength of the j-th TF for the i-th gene. In the case of microarray data, it is particularly convenient to work with relative quantities as in Equation (8.1), because gene expression levels are typically measured with respect to a reference level.

Below we assume that we have n genes, m TFs and k time points, other than zero. Then, taking logarithms in Equation (8.1) gives $Y = AX$, where Y, A and X are $n \times k$, $n \times m$ and $m \times k$ matrices, respectively. The entries of A are the coefficients a_{ij} and for Y and X we have $y_{it} = \log \frac{\widehat{y}_i(t)}{\widehat{y}_i(0)}$ and $x_{jt} = \log \frac{\widehat{x}_j(t)}{\widehat{x}_j(0)}$. The connection to the `CAMF` problem of Chapter 2 should now become clear. Having measured the expression levels Y of the regulated genes, we would like to infer the TF activities X and the regulation strengths A. Thereby, from experimental evidence or from sequence-based prediction of promoter binding we know a-priori that there is no regulation between certain (actually the most) pairs of TFs and genes. In other words, matrix A satisfies a known zero pattern $\mathcal{Z}$. Apart from that, no other constraints can be immediately derived for the factors, because in the log-space nonnegativity is not any more a requirement.

8.2 A review of Network Component Analysis

Network Component Analysis (NCA) is a method that has been developed for inferring the hidden TF activities in bipartite regulatory networks. This is not the only ansatz; see also [PW07]. Since its introduction in 2003, several extensions and improvements have appeared in the bioinformatics literature.

The method was introduced in [LBY$^+$03] by Liao *et al.*, who formalized the problem of TF activity inference in a matrix factorization framework. Furthermore, they pointed out that a method is needed for its solution which, in contrast to Independent Component Analysis and Principal Component Analysis, enforces the zero pattern of A as an explicit constraint. So, for the solution they propose an alternating minimization approach that is essentially the same as Algorithm 1. Most importantly, the authors observed that the zero pattern of A plays a central role for the identifiability of the model, but they did not derive any formal combinatorial characterization. Let us note that it was this last point that motivated the characterization of identifiable graphs that we gave in Chapter 3.

Subsequently, in [BSLR05] Boscolo *et al.* gave some combinatorial conditions for identifiability that can be expressed in terms of the network structure, assuming that the matrices A and X are numerically generic. Their conditions, however, seem to be rather ad hoc, since the authors missed the connection to structural rank. In fact, it turns out that these conditions only capture special cases of structural rank deficiencies and are, therefore, necessary but not sufficient. In this paper, the authors also investigate empirically the sensitivity of NCA to inaccuracies in the hypothesized network topology and, finally, they propose a randomized heuristic for the selection of identifiable subnetworks.

In their preprint paper [NRV07], Narasimhan *et al.* recognized the connection of identifiability to bipartite matchings via the notion of structural rank; so they arrived at a definition that is the same as our basic Definition 3.5. However, they do not study further either the properties of identifiable graphs or any related combinatorial optimization problems.

Apart from the three papers cited above, a number of other NCA-related works have appeared in the literature; here we point out two of them. In [CDHF08], Chang *et al.* propose a fast heuristic for solving the factorization problem, without using an iterative scheme, but rather an analytical solution based on projection matrices. In [TBK$^+$05], Tran *et al.* generalize NCA for the case where in the factorization $Y \approx AX$, there is a zero pattern available not only for A but also for X.

As far as the applications are concerned, we note that the inventors of the method have developed an NCA toolbox for Matlab that is publicly available [1]. The method was first applied by Liao *et al.* on the regulatory network of *S. cerevisiae* for monitoring some TFs that are known to be related to the cell cycle regulation. In a subsequent paper, Kao *et al.* applied NCA to monitor some *E. coli* TFs during the transition from glucose to acetate medium [KYB+04]. The regulatory networks of these two model organisms have been used as a testbed for all NCA-related methods that appeared in the literature in the last years. In the case study of the next section we will make no exception to this rule.

8.3 A case study on subnetwork selection

In this section, we put the theoretical considerations of Chapter 5 about source selection into the context of real-world regulatory networks. In a typical application of NCA, we have available a limited number of time-point measurements, say k. Therefore, and according to the considerations of Section 5.1, from the whole regulatory network we need to choose a subnetwork that consists of *at most k* TFs and their *specific* regulated genes. In most NCA-related papers cited above this is a recurring problem, whose solution, however, is not addressed formally. Actually, it is this observation that motivated our theoretical discussion.

For the subnetwork selection problem the authors of [BSLR05] propose a randomized heuristic, which seems to have been implemented also in the NCA toolbox. Here, we approach the problem with the theoretical observations of Chapter 5 and, in particular, we apply the simplification rules of Section 5.3. We find out that in the three regulatory networks that we studied the task becomes almost trivial after applying these simplifications. This also explains why the randomized heuristic of Boscolo *et al.* works in practice without too many difficulties.

For our experiments we are using the regulatory networks of *E. coli* and *S. cerevisiae*, whose structure is already available in the NCA toolbox. The network of *E. coli* (ECOLI) was obtained from the RegulonDB database [2], while for *S. cerevisiae* there are two networks available, obtained from [HGL+04] (SCEREV-1) and [LRR+02] (SCEREV-2). All three networks have a bipartite structure $G = (C, R; E)$, where C is the set of TFs, R is the set of genes and E are the regulation pairs. Before any further processing, we apply the following data reduction rules.

Rule 1: Let us consider two TFs $x, y \in C$. From Proposition 5.10 we know that if $N(x) \subseteq N(y)$, then we can safely eliminate x and $N(x)$. The elimination of $N(x)$ changes the neighborhoods of some vertices of $C \setminus \{x\}$, so some new neighborhood

[1]http://www.seas.ucla.edu/∼liaoj/download.htm
[2]http://regulondb.ccg.unam.mx

inclusions, that did not exist before, may appear among the remaining TFs. Therefore, we have to repeat the procedure until no neighborhood inclusions can be found among the TFs. If finally some TFs are disconnected, we eliminate them.

Rule 2: We find the connected components of the remaining network and work on each one separately; see also Proposition 3.13. For the factorization $Y \approx AX$ it is an overkill to mix TFs (and the related specific genes) that belong to different connected components. In that case A can be permuted to block-diagonal form and the problem can be reduced to many decoupled subproblems of smaller sizes, which can be solved independently. Surprisingly, the authors of [BSLR05] and apparently also the subnetwork selection routine of the NCA toolbox do not seem to take this simple observation into account.

Rule 3: We find the DM decomposition, which allows us to safely discard some TFs and genes; see Proposition 5.12.

In the three networks that we studied it turns out that the great majority of TFs have *private* genes, i.e., genes that are regulated by a single TF. Such TFs (we call them star-TFs) are important, because they are singleton nicely separable sets (recall Definition 5.4) and, due to Proposition 5.5, can be arbitrarily combined to build larger nicely separable sets. Therefore, the fact that most TFs have private genes almost trivializes subnetwork selection in these regulatory networks. Difficulties can arise only for TFs that have no private genes; in that case the MILP of Section 5.4 can be used for subnetwork selection. If we decide to do so, we can first apply the following simplification rule.

Rule 4: If a TF has private genes, we prune it so that only one private gene remains. This simplification does not change the family of nicely separable sets, but significantly reduces the number of variables in the MILP.

In Table 8.1 we present some results. In all three networks we get some small components that are trivial and one large component; then, we focus on this large component.

	ECOLI	SCEREV-1	SCEREV-2
Initial size (TFs $\times$ genes)	120×828	169×2845	113×6270
Edge density (%)	1.45	1.28	0.63
After rule 1	81×660	161×2822	102×6231
Connected components	37	7	5
Largest component	41×473	155×2810	98×2340
Star-TFs in largest component	36	133	94

Table 8.1: Some statistics for the regulatory networks. For these networks DM decomposition does not help.

In summary, subnetwork selection turns out to be easy for these real-world networks. In fact, we do not need any heuristics or complicated algorithms; an understanding of the problem structure and a few simplification rules are enough for an exact and efficient solution. Although in the presence of these datasets the theoretical discussion seems to be redundant, it is the only way to understand the structure of the problem, recognize that we are in fact dealing with easy instances of the general problem and come up with the simplification rules. Moreover, the power of the general approach allows the processing of more complicated networks that are expected to appear in the future.

8.4 Some directions for further research

We found out that subgraph selection becomes almost trivial in the regulatory networks that we studied. In this section, we sketch some ideas for further research that seem to be more relevant in this setting.

For each TF we count how many private genes it has and for each pair of TFs we find how many neighbors they have in common. With this information we can construct a weighted graph where the vertices correspond to TFs and the edges correspond to co-regulations. The vertices are weighted by the number of private genes, and the edges are weighted by the number of common neighbors of the corresponding TFs. We call this graph co-regulation graph (CR-graph); see Figure 8.1 for an example. Note that if A is the 0/1 matrix that models the original bipartite graph, then $A^T A$ contains the edge weights of the CR-graph.

For a TF that has private genes (i.e. singleton nicely separable set), the estimation of its activity profile across time reduces to a rank-one factorization, if we only consider the expression of the private genes. Furthermore, we intuitively expect that the more specific genes are available for a TF, the more reliable is the estimation of its activity profile. Therefore, in the CR-graph we would prefer to pick nodes with large weights, i.e., TFs with many private genes.

An idea would be to first pick a set of “heavy” TFs and get an initial estimation of their activity profiles across time from the private genes only. Then, at a second pass, we could also take into account the non-private genes, in order to refine the estimation. Thereby, the paths in the CR-graph can provide some useful information, because they encode the coupling of the TFs through co-regulated genes. TFs that are connected with an edge share some common genes, TFs that have distance two in the CR-graph interact with each other through a common TF with which they co-regulate, and so on. Intuitively, the closer two TFs are in the CR-graph, the more important it is to take into account their interaction.

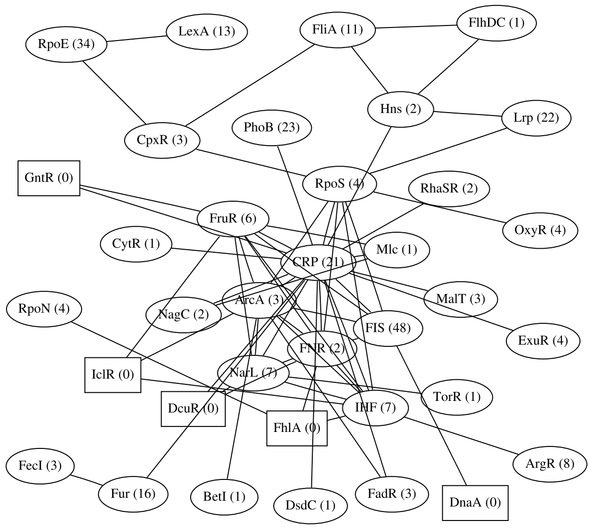


Figure 8.1: Part of the CR-graph for the regulatory network of *E. coli.* The weights of the vertices are shown in parentheses. The weights of the edges (omitted for clarity) are encoded in the 2D embedding of the graph: the heavier an edge is, the closer the incident vertices are drawn on the plane.

Therefore, we could refine the estimation of the TF activities iteratively, following the paths in the CR-graph. That is, we could first combine the TFs that are at distance one from each other by taking into account their common genes and then we could continue by combining TFs in increasing order of distance. In other words, instead of solving a large, loosely coupled matrix factorization problem at once, we divide it in smaller subproblems and combine the solutions of the subproblems by progressively introducing their couplings.

Of course, the preliminary ideas sketched above need to be formalized. What is the exact role of cliques, independent sets and paths in the CR-graph? Can we use this graph in the context of `CAMF` in order to “order” and speed up the computations?

Chapter 9

Conclusions

9.1 Summary

The following problem formed the basis of our work: Given a real-valued matrix Y, we want to approximate it with a low-rank product $Y \approx AX$, where both factors are free variables that have to be determined. Furthermore, we assume that from our a-priori knowledge about the application at hand we can impose some constraints on the factors A and X. It is clear that the above problem is ill-defined, i.e., there are (infinitely many) different factorizations that approximate a given Y equally well. This motivated us to ask the question: Do the constraints on A and X contribute anything to the uniqueness of the factorization and, if yes, how exactly? In particular, we considered an important class of constraints that arise in signal processing applications: In the factorization $Y \approx AX$, A must have zeros at certain positions. Such constraints arise in applications that involve a bipartite network of sources that are emitting some signals over discrete time and sensors that are monitoring these signals.

We arrived at a characterization of uniqueness up to diagonal scaling that is purely combinatorial, in the sense that it is based solely on the structure of the source-sensor network. This is encoded in the definition of *identifiable bipartite graphs*, which is crucially based on bipartite matchings. These are well-studied objects and at many points we benefited from the large body of existing theoretical results related to them.

After establishing the basic definition of identifiable graphs, we focused on two optimization problems, `MINSENSOR` and `MINSOURCE`, that arise in the context of source-sensor networks. Roughly speaking, the goal of both problems is the selection of identifiable subgraphs. We showed the NP-hardness of both problems and presented mixed integer linear programs for their exact solution. Moreover, for `MINSENSOR` we presented a greedy approximation algorithm and performed computational experiments for randomly generated problem instances.

Then, we also asked another natural question that arises from our need to model uncertainty in the network structure. Given an identifiable graph G, where the edges have been predicted with some uncertainty, how many edge deletions and additions does it take, so that G loses the property? We reduced this robustness question to the computation of surplus in bipartite graphs and we show how this can be done in polynomial time. The computation of surplus and tight sets in bipartite graphs may be also of independent interest outside the context of identifiable graphs.

Finally, we presented case-studies for two applications from bioinformatics that can be modeled in the framework of a source-sensor network. The first one is dealing with the processing of microarray data under the presence of cross-hybridization and the second one is dealing with the quantification of transcription factor activities in simple regulatory networks.

9.2 Limitations

For the three basic problems that we studied, `MINSENSOR`, `MINSOURCE` and robustness, the theoretical complications arise due to the high density that is in general possible in the connectivity of the source-sensor network. On the other hand, for problem instances with many specific sensors, i.e., sensors connected to exactly one source, practical heuristics may be sufficient. In most applications, this specificity requirement is something for which much effort is invented in the design phase of the system. On the other hand, understanding the structure of the general problem is not only theoretically appealing, but may also be practically useful in cases where it is difficult to design many specific sensors.

The discussion about identifiable graphs in this thesis was motivated, most of the time, in the signal processing context of a source-sensor network. However, in Chapter 2 we saw that low-rank matrix factorization also has applications in data mining for dimension reduction. Therefore, some parts of our discussion could also have applications in this field. However, we should point out that there is an important limitation. On the one hand, sparsity of the factors is a desired property in dimension reduction, because it contributes to an easier interpretation of the results. On the other hand, in most applications the exact sparsity pattern of the factors is *not known a-priori*, because the purpose of empirical data mining is exactly to explore the structure that is hidden in the data, when there are no strong a-priori assumptions.

9.3 Directions for further research

In the flow of the text we have already pointed out some ideas for future extensions. Here, we collect the most important of them.

Let us recall that, in general, in the factorization $Y \approx AX$, apart from the zero-pattern constraint $A \lhd \mathcal{Z}$, we can also have additional constraints on the factors. In our approach so far we have exclusively concentrated on the conclusions that we can draw solely from the structure of $\mathcal{Z}$. But even if the zero pattern on itself cannot offer identifiability, the extra constraints should also somehow restrict the space of multiple solutions. How exactly can we combine these constraints with the zero-pattern, in order to draw stronger conclusions? We have already sketched this idea in Section 3.5. What happens if in the factorization $Y = AX$ both factors must satisfy known zero patterns? Can we come up with a generalization of identifiability? The next natural step beyond zero patterns would be sign patterns, i.e., constraints enforcing the positivity or negativity of certain entries of the matrices. The study of sign patterns is, in fact, a large field demonstrating many nice connections between linear algebra and graph theory. For an extensive treatment we refer to the book by Brualdi and Shader [BS95]. In fact, sign patterns are much stronger than zero patterns; in some cases they make it possible to draw strong conclusions without even assuming numerical genericity of the involved matrices.

In Section 3.5, we saw that, even in the case of graphs that do not satisfy the identifiability conditions, it is possible to draw structural conclusions about matrix R, using the Dulmage-Mendelsohn decomposition. Moreover, we strongly suspect that the directed graph that models the zero pattern of R is always transitive and, therefore, the inverse R^{-1} also satisfies the same zero pattern as R. Firstly, a formal proof of this conjecture is pending and secondly, can we understand if there is a deeper combinatorial reason that makes this happen?

As far as robustness questions are concerned we point out that the complexity of the weighted robustness questions remained open. More importantly, incorporating robustness as an optimization criterion into the design process (sensor selection) and in the selection of good subgraphs seems to be worth investigating. This would also have practical importance for networks with many specific sensors. Finding identifiable subgraphs in such networks is, in general, easy, but what about finding identifiable subgraphs with a lower bound on their robustness?

Another idea that would be useful in networks with many specific sensors is the following. Assume that each sensor monitors at most k sources, where k is a constant, i.e., in the bipartite graph $G = (C, R; E)$ the degree of all R-vertices is at most k. What is then the complexity of the (generally NP-hard) problems `MINSENSOR` and `MINSOURCE`? Can we probably come up with a fixed-parameter tractable (FPT) algorithm with respect to k? What other parameters does it make sense to consider for an FPT approach?

As a new property of bipartite graphs that has not been studied before, identifiability can be looked at from many perspectives. For example, one may ask what happens with random graphs. How probable is it that they have the property? If we could answer this question then this would also open the possibility for statistical reasoning on real networks. We have made a first attempt to approach these questions, but their study requires rather sophisticated (and hard to grasp) methods from the theory of random graphs. Moreover, the fact that identifiability is not a convex property is definitely discouraging.

Until now, the minimization method that computes the factorization $Y \approx AX$ and the uniqueness questions have been considered independent from each other. We have assumed that a solution to the factorization problem is computed with a black-box method and we focused our attention to the identifiability questions. A method to actually compute the factorization was described (see Algorithm 1) and implemented for our case study of cross-hybridization. However, we have not systematically investigated how this minimization algorithm scales to large problems. How exactly do the structure and type of the constraints influence the performance of the numerical computations? Can we speed up the computations by taking advantage of the known zero pattern and "ordering" the computations appropriately?

In this thesis, we saw how constrained matrix factorization, a problem that is continuous in its own nature, naturally led to discrete graph-theoretical investigations. Our motivation was the question about the uniqueness of solutions under the presence of zero constraints and the key for the answer was the definition of *identifiable bipartite graphs*. We hope that this new connection, apart from being theoretically appealing, will also contribute to a better understanding of problems that can be modeled in the framework of linear, bipartite source-sensor networks.

Bibliography

[BBL$^+$07] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and applications for approximate non-negative matrix factorization. *Computational Statistics and Data Analysis*, 2007.

[BDJ99] M.W. Berry, Z. Drmač, and E.R. Jessup. Matrices, vector spaces and information retrieval. *SIAM Review*, 41(2):335–362, 1999.

[Beh03] S. Behnke. Discovering hierarchical speech features using convolutional non-negative matrix factorization. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pages 2758–2763, 2003.

[Ber57] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences USA*, 43:842–844, 1957.

[BS95] R.A. Brualdi and B.L. Shader. *Matrices of Sign-Solvable Linear Systems.* Cambridge University Press, 1995.

[BSLR05] R. Boscolo, C. Sabatti, J.C. Liao, and V.P. Roychowdhury. A generalized framework for network component analysis. *IEEE Trans. Comp. Biol. Bioinf.*, 2(4):289–301, 2005.

[BTGM04] J.-P. Brunet, P. Tamayo, T.R. Golub, and J.P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *PNAS*, 101(12):4164–4169, 2004.

[BTW$^+$09] T. Barrett, D.B. Troup, S.E. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I.F. Kim, A. Soboleva, M. Tomashevsky, K.A. Marshall, K.H. Phillippy, P.M. Sherman, R.N. Muertter, and R. Edgar. NCBI GEO: archive for high-throughput functional genomic data. *Nucleic Acids Research*, 37(Database issue):D885–D890, 2009.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

[CCL$^+$06] Y.A. Chen, C.-C. Chou, X. Lu, E.H. Slate, K. Peck, W. Xu, E.O. Voit, and J.S. Almeida. A multivariate prediction model for microarray cross-hybridization. *BMC Bioinformatics*, 7:101, 2006.

[CCSS$^+$06] Monica Chagoyen, Pedro Carmona-Saez, Hagit Shatkay, Jose M Carazo, and Alberto Pascual-Montano. Discovering semantic features in the literature: a foundation for building functional associations. *BMC Bioinformatics*, 7:41, 2006.

[CDHF08] C. Chang, Z. Ding, Y.S. Hung, and P.C.W. Fung. Fast network component analysis (FastNCA) for gene regulatory network reconstruction from microarray data. *Bioinformatics*, 24(11):1349–1358, 2008.

[CF02] M. Cooper and J. Foote. Summarizing video using non-negative similarity matrix factorization. In *IEEE Workshop on Multimedia Signal Processing*, pages 25–28, 2002.

[CSPMT$^+$06] P. Carmona-Saez, R.D. Pascual-Marqui, F. Tirado, J.M. Carazo, and A. Pascual-Montano. Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinformatics*, 7:78, 2006.

[Die05] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2005.

[DM58] A.L. Dulmage and N.S. Mendelsohn. Coverings of bipartite graphs. *Canad. J. Math.*, 10:517–534, 1958.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[Fod02] I.K. Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Laboratory, 2002.

[FRSR08] E. Fritzilas, Y.A. Rios-Solis, and S. Rahmann. Structural identifiability in low-rank matrix factorization. In X. Hu and J. Wang, editors, *Computing and Combinatorics*, volume 5092 of *LNCS*, pages 140–148. 2008. Proceedings of the 14th International Conference COCOON 2008 Dalian, China.

[Fuj05] S. Fujishige. *Submodular Functions and Optimization.* Elsevier, 2005.

[GB08] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming (web page and software). http://stanford.edu/∼boyd/cvx, December 2008.

[Gil94] J.R. Gilbert. Predicting structure in sparse matrix computations. *SIAM Journal on Matrix Analysis and Applications*, 15(1):62–79, 1994.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[GKS95] M.C. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *J. Algorithms*, 19:449–473, 1995.

[GL96] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[GS00] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26:127–136, 2000.

[Hal35] P. Hall. On representatives of subsets. *J. London Math. Society*, 10:26–30, 1935.

[HGL$^+$04] C.T. Harbison, D.B. Gordon, T.I. Lee, N.J. Rinaldi, K.D. Macisaac, T.W. Danford, N.M. Hannett, J.-B. Tagne, D.B. Reynolds, J. Yoo, E.G. Jennings, J. Zeitlinger, D.K. Pokholok, M. Kellis, P.A. Rolfe, K.T. Takusagawa, E.S. Lander, D.K. Gifford, E. Fraenkel, and R.A. Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431(7004):99–104, 2004.

[HH82] R.J. Hanson and K.H. Haskell. Two algorithms for the linearly constrained least-squares problem. *ACM Transactions on Mathematical Software*, 8(3):323–333, 1982.

[HK73] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2:225–231, 1973.

[HMSG08] L.N. Hutchins, S.M. Murphy, P. Singh, and J.H. Graber. Position-dependent motif characterization using non-negative matrix factorization. *Bioinformatics*, 24(23):2684–2690, 2008.

[Hoy04] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning*, 5:1457–1469, 2004.

[HS06] M. Heiler and C. Schnorr. Learning sparse representations by non-negative matrix factorization and sequential cone porgramming. *Journal of Machine Learning Research*, 7:1385–1407, 2006.

[Iwa03] S. Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM J. Comput.*, 32:833–840, 2003.

[KP07] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23:1495–1502, 2007.

[KYB$^+$04] K.C. Kao, Y.-L. Yang, R. Boscolo, C. Sabatti, V. Roychowdhury, and J.C. Liao. Transcriptome-based determination of multiple transcription regulator activities in escherichia coli by using network component analysis. *Proc Natl Acad Sci U S A*, 101(2):641–646, Jan 2004.

[LBY$^+$03] J.C. Liao, R. Boscolo, Y.-L. Yang, L.M. Tran, C. Sabatti, and V.P. Roychowdhury. Network component analysis: reconstruction of regulatory signals in biological systems. *PNAS*, 100(26):15522–15527, 2003.

[Lin07] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19:2756–2779, 2007.

[LP86] L. Lovász and M.D. Plummer. *Matching Theory.* North-Holland, 1986.

[LR04] C. Lee and M. Roy. Analysis of alternative splicing with microarrays: successes and challenges. *Genome Biology*, 5:231, 2004.

[LRR$^+$02] T.I. Lee, N.J. Rinaldi, F. Robert, D.T. Odom, Z. Bar-Joseph, G.K. Gerber, N.M. Hannett, C.T. Harbison, C.M. Thompson, I. Simon, J. Zeitlinger, E.G. Jennings, H.L. Murray, D.B. Gordon, B. Ren, J.J. Wyrick, J.-B. Tagne, T.L. Volkert, E. Fraenkel, D.K. Gifford, and R.A. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(5594):799–804, 2002.

[LS99] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[LW01] C. Li and W.H. Wong. Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. *Proceedings of National Academy of Sciences*, 98(1):31–36, 2001.

[LXY03] J. Lu, B. Xu, and H. Yang. Matrix dimensionality reduction for mining web logs. In *IEEE/WIC International Conference on Web Intelligence, 2003*, page pp.405, 2003.

[MRM$^+$07] C. Militon, S. Rimour, M. Missaoui, C. Biderre, V. Barra, D. Hill, A. Mon, G. Gagne, H. Meier, E. Peyretaillade, and P. Peyret. Phylarray: phylogenetic probe design algorithm for microarray. *Bioinformatics*, 23(19):2550–2557, 2007.

[Mur00] K. Murota. *Matrices and Matroids for Systems Analysis.* Springer, 2000.

[Nar97] H. Narayanan. *Submodular Functions and Electrical Networks.* Elsevier, 1997.

[NRV07] S. Narasimhan, R. Rengaswamy, and R. Vadigepalli. Structural properties of gene regulatory networks: definitions and connections. *IEEE Trans. Comp. Biol. Bioinf.*, 2007. accepted.

[Nuu95] E. Nuutila. *Efficient Transitive Closure Computation in Large Digraphs*, volume 74 of *Acta Polytechnica Scandinavica, Mathematics and Computing in Engineering Series.* Finnish Academy of Technology, 1995.

[Orl07] James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. In *Proceedings of IPCO'07*, volume 4513 of *LNCS*, pages 240–251, 2007.

[OSF$^+$08] N. Ono, S. Suzuki, C. Furusawa, T. Agata, A. Kashiwagi, H. Shimizu, and T. Yomo. An improved physico-chemical model of hybridization on high-density oligonucleotide microarrays. *Bionformatics*, 24:1278–1285, 2008.

[PF90] A. Pothen and C.-J. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Software*, 16(4):303–324, 1990.

[PSM$^+$04] Q. Pan, O. Shai, C. Misquitta, W. Zhang, A.L. Saltzman, N. Mohammad, T. Babak, H. Siu, T.R. Hughes, Q.D. Morris, B.J. Frey, and B.J. Blencowe. Revealing global regulatory features of mammalian alternative splicing using a quantitative microarray platform. *Molecular Cell*, 16(6):929–941, 2004.

[PW07] I. Pournara and L. Wernisch. Factor analysis for gene regulatory networks and transcription factor activity profiles. *BMC Bioinformatics*, 8(1):61, 2007.

[Rah04] S. Rahmann. *Algorithms for probe selection and DNA microarray design.* PhD thesis, Free University of Berlin, Department of Mathematics and Computer Science, February 2004.

[Sar42] A. Sard. The measure of the critical values of differentiable maps. *Bulletin of the American Mathematical Society*, 48(12):883–890, 1942.

[SBPP06] F. Shahnaz, M.W. Berry, V.P. Pauca, and R.J. Plemmons. Document clustering using nonnegative matrix factorization. *Inf. Proc. & Manag.*, 42(2):373–386, 2006.

[SMBF06] O. Shai, Q.D. Morris, B.J. Blencowe, and B.J. Frey. Inferring global levels of alternative splicing isoforms using a generative model of microarray data. *Bioinformatics*, 22(5):606–613, 2006.

[TBK$^+$05] L.M. Tran, M.P. Brynildsen, K.C. Kao, J.K. Suen, and J.C. Liao. gnca: a framework for determining transcription factor activity based on transcriptome: identifiability and numerical implementation. *Metab Eng*, 7(2):128–141, 2005.

[Vaz04] V. Vazirani. *Approximation Algorithms.* Springer, 2004.

[VBB08] E. Vincent, N. Bertin, and R. Badeau. Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 109–112, 2008.

[WHH$^+$03] H. Wang, E. Hubbell, J. Hu, G. Mei, M. Cline, G. Lu, T. Clark, M.A. Siani-Rose, M. Ares, D.C. Kulp, and D. Haussler. Gene structure-based splice variant deconvolution using a microarray platform. *Bioinformatics*, 19:i315–i322, 2003.

[WM04] E. Wit and J. McClure. *Statistics for Microarrays: Design, Analysis and Inference.* Wiley, 2004.

[WMP$^+$07] Y. Wang, Z.-H. Miao, Y. Pommier, E.S. Kawasaki, and A. Player. Characterization of mismatch and high-signal intensity probes associated with affymetrix genechips. *Bioinformatics*, 23(16):2088–2095, 2007.

[Wol82] L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[Wol98] L.A. Wolsey. *Integer Programming.* Wiley Interscience, 1998.