The Discrete Basis Problem

Pauli Miettinen

Helsinki 23rd December 2005

M.Sc. Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author
Pauli Miettinen

Työn nimi — Arbetets titel — Title

The Discrete Basis Problem

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level M.Sc. Thesis	Aika — Datum — Month and year 23rd December 2005	Sivumäärä — Sidoantal — Number of pages 54 pages

Tiivistelmä — Referat — Abstract

We consider the Discrete Basis Problem, which can be described as follows: given a collection of Boolean vectors find a collection of k Boolean basis vectors such that the original vectors can be represented using disjunctions of these basis vectors. We show that the decision version of this problem is NP-complete and that the optimization version cannot be approximated within any finite ratio. We also study two variations of this problem, where the Boolean basis vectors must be mutually otrhogonal. We show that the other variation is closely related with the well-known Metric k-median Problem in Boolean space.

To solve these problems, two algorithms will be presented. One is designed for the variations mentioned above, and it is solely based on solving the k-median problem, while another is a heuristic intended to solve the general Discrete Basis Problem. We will also study the results of extensive experiments made with these two algorithms with both synthetic and real-world data. The results are twofold: with the synthetic data, the algorithms did rather well, but with the real-world data the results were not as good.

In addition we will study some of the related work, namely discrete principal component analysis, database tiling and bicliques. We will see that there are commonalities between the Discrete Basis Problem and some problems involving database tiling or bicliques.

ACM Computing Classification System (CCS):
F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*,
H.2.8 [Database Management]: Database Applications—*Data mining*,
I.5.3 [Pattern Recognition]: Clustering—*Algorithms*.

Avainsanat — Nyckelord — Keywords
Discrete basis, k-medians, SVD, PCA, database tiling, data mining, bicliques.

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpula Science Library, serial number C-

Muita tietoja — Övriga uppgifter — Additional information

Acknowledgments

Hast thou not dragged Diana from her car?
   And driven the Hamadryad from the wood
To seek a shelter in some happier star?
   Hast thou not torn the Naiad from her flood,
The Elfin from the green grass, and from me
The summer dream beneath the tamarind tree?

Sonnet—To Science
Edgar Allan Poe

I wish to thank the following persons for making this thesis possible: professor Heikki Mannila who originally proposed the problem and who has also worked as my supervisor; Taneli Mielikäinen for the original ideas of the proofs in Section 2.2 and for many other valuable ideas; professor Gautam Das from University of Texas at Arlington for the ideas presented in Section 2.3; Aristides Gionis for so many valuable ideas and corrections for preliminary versions of this thesis while working as my other supervisor.

To Anu, who tolerated me and my continuing absence.

Contents

1 Introduction

As the name proposes, the Discrete Basis Problem is about finding a basis from given data. A basis of a data means another set of data, usually considerably smaller, that can be used to (approximately) reconstruct the original data. It can be very useful if one can find a good basis for a data. Such basis can be used, e.g., to summarize the properties of data or to reduce the space needed to save the data.

Probably the currently best known method for finding this kind of bases is the Principal Component Analysis. The Principal Component Analysis is a method used in continuous space, i.e., when calculating with real numbers, while the Discrete Basis Problem is to be used with Boolean values, i.e., in discrete spaces. Many variations to PCA have been proposed in order to make it work with discrete data. These methods include, but are not restricted to, multinomial PCA by Wray Buntine et al. [Bun02], probabilistic Latent Semantic Indexing by Thomas Hofmann [Hof99], non-negative matrix factorization by Daniel Lee and H. Sebastian Seung [LS99] and Aspect Bernoulli Model by Ata Kabán, Ella Bingham and Teemu Hirsimäki [KBH04]. The Discrete Basis Problem differs from these as it requires also the results to be in Boolean space. However, it can also be viewed as an analogue for the Principal Component Analysis.

In addition to describing the Discrete Basis Problem, we study the computational complexity of it. We prove that the Discrete Basis Problem is $\mathcal{NP}$-complete. We also study a variation of the problem and prove that it is also $\mathcal{NP}$-complete. We propose two algorithms for the problem and report the results of exhaustive experiments made with these algorithms. Naturally, the most important related work is also covered.

The rest of the thesis is organized as follows. The notational conventions and some initial definitions are given in Section 1.1. Section 2 is about the computational complexity of the problem. In Section 2.1 we will see the formal definition of the problem, and in Section 2.2 we will prove some hardness results for it. The Section 2.3 concentrates on variations of the Discrete Basis Problem, viz. the Disjoint Discrete Basis Problem and the Discrete Basis Partition Problem.

Section 3 summarizes the related work. Section 3.2 is about the most obvious related subject, Principal Component Analysis. We will discuss about the relation between the bipartite graphs and the Discrete Basis Problem in Section 3.3. Some related problems from the field of data mining are presented in Section 3.4.

We describe the two proposed algorithms in Section 4. Section 4.1 presents the LocalSearch algorithm, especially designed for the Disjoint Discrete Basis Problem and Discrete Basis Partition Problem, and Section 4.2 presents the Association algorithm for the general Discrete Basis Problem. Section 5 reports the experiments made with these algorithms: Section 5.1 describes the test settings and Sections 5.2 and 5.3 reports the results for the LocalSearch and Association algorithms, respectively. Finally, we will make some conclusions and list some of the possible future work in Section 6.

1.1 Notation

Some knowledge on theory of probability and Bayesian data analysis may be helpful in Section 3.2. Otherwise, the mathematics used is mainly basic matrix and set algebra. In the following, we will make some initial definitions and notational conventions, which we will use through the rest of the thesis.

Matrices are written in upper case boldface letters, e.g., $\mathbf{M}$. Vectors are denoted by lower case boldface letters, e.g., $\mathbf{v}$. The ith row and column vectors of matrix $\mathbf{M}$ are denoted by $\mathbf{m}_{i\cdot}$ and $\mathbf{m}_{\cdot i}$, respectively. An element of matrix or vector is denoted by corresponding lower case italics letter with appropriate subscript denoting the placement, i.e., m_{ij} is in ith row and jth column of matrix $\mathbf{M}$ and v_i is the ith element of vector $\mathbf{v}$.

Matrices used in this thesis are mainly Boolean matrices, i.e., their elements are only 1s or 0s. The number of columns in matrix is called the *dimension* of the matrix. Matrices and vectors may be constructed using usual parenthesis notation, i.e., $(v_i)_{i=1}^{d}$ is a d-dimensional vector $\mathbf{v}$. The transpose of matrix $\mathbf{M}$ is denoted by $\mathbf{M}^T$. The d-dimensional identity matrix is denoted by $\mathbf{I}_d$, and if the dimension is clear from context, or the dimension is not important, $\mathbf{I}$ is used as a shorthand.

Sets are denoted by upper case letters, e.g., S. Collections of sets are denoted, as usual, by upper case calligraphic letters, e.g., $\mathcal{S}$. As with matrices and vectors, using the same letter with a collection and a set denotes that the set is in the collection. The elements of sets are denoted by lowercase letters, usually without subscripts. The notation $\cup\mathcal{S}$ is used as a shorthand for an union over the sets in collection $\mathcal{S}$. That is to say,

$$\cup\mathcal{S} = \bigcup_{S\in\mathcal{S}} S.$$

The cardinality of a set S is denoted by $|S|$. Sets we consider in this thesis are finite sets of some finite universe. If the cardinality of the universe is d, then all sets in that universe may be presented by d-dimensional Boolean vectors, such that the ith element in vector is 1 if and only if the corresponding ith element in the universe belongs to the corresponding set. Thus, sets and vectors—and collections and matrices—are used interchangeably.

Logical operators used are $\vee$, $\wedge$ and $\sqcup$ meaning *or*, *and* and *exclusive or*, respectively. If $\mathbf{v}$ and $\mathbf{w}$ are Boolean vectors of dimension d,

$$\mathbf{v} \vee \mathbf{w} = (v_i \vee w_i)_{i=1}^{d}$$

is used as a shorthand.

Definition 1.1 (Symmetric difference). The *symmetric difference* between sets A and B, $A \triangle B$, is defined as

$$A \triangle B = (A \backslash B) \cup (B \backslash A) = (A \cup B) \backslash (A \cap B).$$

Definition 1.2 (Boolean matrix multiplication). A Boolean matrix multiplication between Boolean matrices $\mathbf{A} \in \{0,1\}^{m \times n}$ and $\mathbf{B} \in \{0,1\}^{n \times p}$ is $\mathbf{A} \otimes \mathbf{B} = \mathbf{C}$, where $\mathbf{C}$ is in space $\{0,1\}^{m \times p}$ and

$$c_{ij} = \bigvee_{k=1}^{n} \left(a_{ik} \wedge b_{kj}\right).$$

The discrete L_1-norm is used in this thesis and it is denoted by $\|\cdot\|_1$. It is defined as follows.

Definition 1.3 (L_1-norm). The L_1-norm of d-dimensional vector $\mathbf{v} \in X^d$, for some set X, is

$$\|\mathbf{v}\|_1 = \sum_{i=1}^{d} |v_i|.$$

The L_1-norm also defines a *distance metric* between vectors, referred as L_1-metric and defined as

$$\|\mathbf{v} - \mathbf{w}\|_1 = \sum_{i=1}^{d} |v_i - w_i|.$$

The L_1-metric between vectors is expanded to matrices in natural way, i.e., if $\mathbf{A}$ and

B are matrices in $X^{n\times m}$, for some set X, then

$$\|\mathbf{A}-\mathbf{B}\|_1 = \sum_{i=1}^{n}\|\mathbf{a}_{i\cdot}-\mathbf{b}_{i\cdot}\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}-b_{ij}|.$$

The names of probability distributions are given in upright letters followed by the parameters of distribution. If x is a normally distributed random variable with mean μ and variance σ^2, then we denote it by

$$x \sim \mathrm{N}(\mu,\sigma^2) \quad \text{or} \quad x \sim \mathrm{Gaussian}(\mu,\sigma^2).$$

2 The Discrete Basis Problem

This section defines the Discrete Basis Problem, and is organized as follows. At first, Section 2.1 gives a formal definition of the Discrete Basis Problem. It also defines another, closely related problem known as the Basis Usage Problem. To emphasize many aspects of the Discrete Basis Problem, we give another equivalent definition for the Discrete Basis Problem at the end of Section 2.1. In Section 2.2 we study the complexity of the problem, and show that the Discrete Basis Problem is $\mathcal{NP}$-complete and that it cannot be approximated within any finite ratio in polynomial time unless $\mathcal{P}$=$\mathcal{NP}$. Section 2.3 concentrates on important variations of the Discrete Basis Problem, viz. the Disjoint Discrete Basis Problem and the Discrete Basis Partition Problem. That section also proves that the Discrete Basis Partition Problem is $\mathcal{NP}$-complete, and that it can be approximated within a constant approximation factor.

2.1 Problem definition

The Discrete Basis Problem (DBP) is the following: given a collection of Boolean vectors find a collection of k Boolean basis vectors such that the original vectors can be represented using disjunctions of the basis vectors. More formally, we can define the problem as follows.

Problem 1 (Discrete Basis Problem). Given a matrix $\mathbf{C} \in \{0,1\}^{n\times d}$ and a positive integer $k < \min\{n,d\}$, find a matrix $\mathbf{B} \in \{0,1\}^{k\times d}$ minimizing

$$\ell_{\otimes}(\mathbf{C},\mathbf{B}) = \min_{\mathbf{S}\in\{0,1\}^{n\times k}} \|\mathbf{C}-\mathbf{S}\otimes\mathbf{B}\|_1. \tag{2.1}$$

Matrix $\mathbf{B}$ is called the *basis* and its row vectors $\mathbf{b}_{i\cdot}$ are called *basis vectors*. We will say that a column j *belongs* to a basis vector $\mathbf{b}_{i\cdot}$ if $b_{ij} = 1$. Vectors $\mathbf{c}_{i\cdot}$ are input vectors. An element c_{ij} in input matrix $\mathbf{C}$ is *covered* by $\mathbf{B}$, if for the matrix $\mathbf{S}$ minimizing (2.1) we have $(\mathbf{S} \otimes \mathbf{B})_{ij} = 1$. Equation (2.1) defines the *loss function* $\ell_\otimes$ for the Discrete Basis Problem and, thus, in the DBP the objective is to minimize the number of differences between the original matrix and the matrix constructed by using the found basis.

L_1-metric is not the only possible function to be used in $\ell_\otimes$. We could, for instance, count only the number of 1s that are not covered or the number of 0s that are covered. While these loss functions may fit in some situations, they lack one major property: they are not metrics. There are, of course, other metrics to be used. However, we think that the L_1-metric is the most intuitive one for this problem and thus it is used (see Section 2.3 for some other reasons for selecting it). Some other possible loss functions are presented in Section 3.

Example 2.2 (example matrices for the DBP). We can see a simple example with matrices $\mathbf{C}$, $\mathbf{B}$, $\mathbf{S}$ and $\mathbf{S} \otimes \mathbf{B}$ in Figure 1. Matrices $\mathbf{B}$ and $\mathbf{S}$ are optimal if $k = 2$. The value of the loss function is

$$\ell_\otimes(\mathbf{C}, \mathbf{B}) = \|\mathbf{C} - \mathbf{S} \otimes \mathbf{B}\|_1 = 1.$$

$\square$

In Problem 1, the upper bound for k prevents the problem from reducing to trivial cases. If $k = n$, selecting $\mathbf{B} = \mathbf{C}$ is the answer. If, on the other hand, $k = d$, selecting $\mathbf{B} = \mathbf{I}_d$, a d-dimensional identity matrix, is the answer.

Note that the definition of the Problem 1 only asks for matrix $\mathbf{B}$. It does not require that we find the matrix $\mathbf{S}$ minimizing (2.1). In fact, finding $\mathbf{S}$ is a problem of its own.

$$\underset{\mathbf{S}}{\begin{pmatrix}1 & 0\\0 & 1\\1 & 1\end{pmatrix}} \qquad \underset{\mathbf{B}}{\begin{pmatrix}1 & 0 & 1\\0 & 1 & 1\end{pmatrix}} \qquad \underset{\mathbf{S}\otimes\mathbf{B}}{\begin{pmatrix}1 & 0 & 1\\0 & 1 & 1\\1 & 1 & 1\end{pmatrix}} \qquad \underset{\mathbf{C}}{\begin{pmatrix}1 & 0 & 1\\0 & 1 & 1\\1 & 1 & 0\end{pmatrix}}$$

Figure 1: An example input matrix $\mathbf{C}$ for the DBP and one possible optimal basis matrix $\mathbf{B}$ and matrices $\mathbf{S}$ and $\mathbf{S} \otimes \mathbf{B}$.

Problem 2 (Basis Usage Problem). Given a matrix $\mathbf{C} \in \{0,1\}^{n\times d}$ and a matrix $\mathbf{B} \in \{0,1\}^{k\times d}$, find a matrix $\mathbf{S} \in \{0,1\}^{n\times k}$ minimizing

$$\|\mathbf{C} - \mathbf{S} \otimes \mathbf{B}\|_1.$$

Separating the Basis Usage Problem from the Discrete Basis Problem makes the DBP more applicable for different problems. On one hand, if we consider the DBP e.g., as a data compression problem, then we must also solve the Basis Usage Problem. On the other hand, if we consider the DBP as a summarization problem, then we may as well chose not to solve the Basis Usage Problem.

The Discrete Basis Problem can also be described using sets and set theory. For many practical applications it is a much more intuitive way of describing the problem.

Problem 3 (DBP, a set version). Given a finite set U (a universe) and a collection $\mathcal{C}$ of subsets of U and a positive integer $k < \min\{|\mathcal{C}|, |U|\}$, find a collection $\mathcal{B}$ of k subsets of U minimizing

$$\ell_{\triangle}(\mathcal{C}, \mathcal{B}) = \sum_{C\in\mathcal{C}} \min_{\mathcal{S}\subseteq\mathcal{B}} |C \triangle \cup\mathcal{S}|.$$

It is rather straightforward to see that the definition of Problem 3 is equivalent to that of Problem 1. Also the Basis Usage Problem (Problem 2) can be easily formulated by using the notations of Problem 3. Both of these definitions for the DBP have their own advantages and disadvantages, and they are used interchangeably through the rest of the thesis.

2.2 Complexity of the problem

When a new problem is proposed, one of the first issues to address is the computational complexity of it. Knowing the problem's computational complexity is crucial for selecting correct approaches when trying to solve it. The most commonly used classes for computational complexity are the classes $\mathcal{P}$ and $\mathcal{NP}$. For problems in class $\mathcal{P}$ there exists a polynomial time (with respect to the input size) algorithm to solve them, while the existence of such algorithms for problems in class $\mathcal{NP}$ is an open problem. However, most computer scientists believe that no such algorithms exists. Informally, problems in class $\mathcal{NP}$ are referred as probably *intractable* problems. We will use the term *algorithm* throughout this section as an informal characterization of what can be computed with normal computer.

In order to study the computational complexity of the Discrete Basis Problem, we must first make some initial definitions.

Definition 2.3 (decision problems and their solutions [ACG$^+$03, p. 1, 10]). A *decision problem* $\mathscr{P}$ is a relation $\mathscr{P} \subseteq I_{\mathscr{P}} \times \{\text{"yes"}, \text{"no"}\}$, where $I_{\mathscr{P}}$ is the set of all *instances* of $\mathscr{P}$. Instances are partitioned into a set $Y_{\mathscr{P}} = \{x \in I_{\mathscr{P}} \mid \mathscr{P}(x, \text{"yes"})\}$ of *positive instances* and a set $N_{\mathscr{P}} = \{x \in I_{\mathscr{P}} \mid \mathscr{P}(x, \text{"no"})\}$. The problem $\mathscr{P}$ asks, for any instance $x \in I_{\mathscr{P}}$, to verify whether $x \in Y_{\mathscr{P}}$. A decision problem $\mathscr{P}$ is *solved* by an algorithm $\mathscr{A}$ if the algorithm halts for every instance $x \in I_{\mathscr{P}}$, and returns "yes" if and only if $x \in Y_{\mathscr{P}}$.

To establish relations among the complexities of different problems, we need *reductions.* Using a reduction we can solve a problem $\mathscr{P}_1$ using an algorithm for a problem $\mathscr{P}_2$. A type of reduction often used is the so-called *polynomial time many-to-one reduction.*

Definition 2.4 (polynomial time many-to-one reducibility and reductions [ACG$^+$03, p. 17–18]). A decision problem $\mathscr{P}_1$ is said to be *polynomial time many-to-one reducible* to a decision problem $\mathscr{P}_2$ if there exists a polynomial time algorithm $\mathscr{R}$ which given any instance $x \in I_{\mathscr{P}_1}$ of $\mathscr{P}_1$, transforms it into an instance $y \in I_{\mathscr{P}_2}$ of $\mathscr{P}_2$ in such a way that $x \in Y_{\mathscr{P}_1}$ if and only if $y \in Y_{\mathscr{P}_2}$. In such case, $\mathscr{R}$ is said to be a *polynomial time many-to-one reduction* from $\mathscr{P}_1$ to $\mathscr{P}_2$ and we write $\mathscr{P}_1 \leq_m^p \mathscr{P}_2$.

The only complexity class that we are interested about in this thesis is the class $\mathcal{NP}$, defined as follows.

Definition 2.5 (class $\mathcal{NP}$ [Pap95, p. 181]). A decision problem $\mathscr{P}$ is in class $\mathcal{NP}$ if and only if there is a relation R such that

1. there is a polynomial time algorithm to solve the problem whether $(x, y) \in R$ for any pair $\langle x, y\rangle$ in the set of that problem's instances;
2. if $(x, y) \in R$, then $|y| \leq |x|^k$ for some $k \geq 1$; and
3. $Y_{\mathscr{P}} = \{x \mid (x, y) \in R \text{ for some } y\}$.

The y for x such that $(x, y) \in R$ is the *polynomial witness* of x.

Polynomial time many-to-one reductions are used to establish relations between arbitrary problems and problems known to be in $\mathcal{NP}$. If problem $\mathscr{P}_1$ is polynomial time many-to-one reducible to problem $\mathscr{P}_2$, then, broadly speaking, $\mathscr{P}_2$ is at least as hard as $\mathscr{P}_1$. Thus we say that a decision problem $\mathscr{P}$ is *$\mathcal{NP}$-hard*, if any decision problem $\mathscr{P}_1$ in class $\mathcal{NP}$ can be reduced to it. Formally the definition is as follows.

Definition 2.6 ($\mathcal{NP}$-hardness [ACG+03, p. 21]). A decision problem $\mathscr{P}$ is said to be *$\mathcal{NP}$-hard* if, for any decision problem $\mathscr{P}_1 \in \mathcal{NP}$, $\mathscr{P}_1 \leq_m^p \mathscr{P}$.

The definition of *$\mathcal{NP}$-completeness* is the last definition we need in order to study the computational complexity of the Discrete Basis Problem.

Definition 2.7 ($\mathcal{NP}$-completeness [ACG+03, p. 21]). A decision problem $\mathscr{P}$ is said to be *$\mathcal{NP}$-complete* if it is in class $\mathcal{NP}$, and it is $\mathcal{NP}$-hard.

In order to prove that a problem $\mathscr{P}$ is $\mathcal{NP}$-hard, it is enough, by definition of $\mathcal{NP}$-hardness, to prove that there is one $\mathcal{NP}$-hard problem $\mathscr{P}_1$ such that $\mathscr{P}_1 \leq_m^p \mathscr{P}$. As an immediate consequence, to prove that $\mathscr{P}$ is $\mathcal{NP}$-complete, it is enough to prove that $\mathscr{P} \in \mathcal{NP}$ and that $\mathscr{P}_1 \leq_m^p \mathscr{P}$ for some $\mathcal{NP}$-hard problem $\mathscr{P}_1$.

The Discrete Basis Problem is not a decision problem, but an optimization problem. The formal definition of an optimization problem is as follows.

Definition 2.8 (optimization problems [ACG+03, p. 22]). An *optimization problem* $\mathscr{P}$ is characterized by the following quadruple of objects $(I_{\mathscr{P}}, \mathrm{SOL}_{\mathscr{P}}, \ell_{\mathscr{P}}, \mathrm{goal}_{\mathscr{P}})$, where:

1. $I_{\mathscr{P}}$ is the set of instances of $\mathscr{P}$;
2. $\mathrm{SOL}_{\mathscr{P}}$ is a function that associates to any input instance $x \in I_{\mathscr{P}}$ the set of *feasible solutions* of x;
3. $\ell_{\mathscr{P}}$ is the *loss function*, defined for pairs (x, y) such that $x \in I_{\mathscr{P}}$ and $y \in \mathrm{SOL}_{\mathscr{P}}(x)$. For every such pair (x, y), $\ell_{\mathscr{P}}(x, y)$ provides a non-negative integer which is the value of the feasible solution y;
4. $\mathrm{goal}_{\mathscr{P}} \in \{\textsc{min}, \textsc{max}\}$ specifies whether $\mathscr{P}$ is a maximization or a minimization problem.

To study the computational complexity of optimization problems, such as the Discrete Basis Problem, we must first note that for any optimization problem $\mathscr{P}$, where

$\text{goal}_{\mathscr{P}} = \text{MIN}$, there is a corresponding decision problem asking, whether there is $y \in \text{SOL}_{\mathscr{P}}(x)$ for given input $x \in I_{\mathscr{P}}$ such that $\ell_{\mathscr{P}}(x, y) \leq a$ for given positive integer a. For maximization problems the direction of the inequality is naturally reversed. We can use this corresponding decision problem to study the computational complexity of an optimization problem $\mathscr{P}$: If the corresponding decision problem is $\mathcal{NP}$-hard, we say that $\mathscr{P}$ is $\mathcal{NP}$-hard optimization problem. Furthermore, if the corresponding decision problem is $\mathcal{NP}$-complete, we say (informally), that $\mathscr{P}$ is $\mathcal{NP}$-complete optimization problem.

The decision version of the DBP (dDBP) is as follows.

Problem 4 (decision DBP). Given a finite set U (a universe) and a collection $\mathcal{C}$ of subsets of U and integers $0 < k < \min\{|\mathcal{C}|, |U|\}$ and $a \geq 0$, is there a collection $\mathcal{B}$ of k subsets of U such that

$$\sum_{C \in \mathcal{C}} \min_{\mathcal{S} \subseteq \mathcal{B}} |C \bigtriangleup \cup \mathcal{S}| \leq a?$$

To prove that the dDBP is in fact $\mathcal{NP}$-complete, we will first prove that it is $\mathcal{NP}$-hard. For that, let us consider the following problem [GJ79, problem SP7].

Problem 5 (Set Basis Problem). Given a finite set U and a collection $\mathcal{C}$ of subsets of U and a positive integer $k < \min\{|\mathcal{C}|, |U|\}$, is there a collection $\mathcal{B}$ of subsets of U with $|\mathcal{B}| = k$ such that

$$\sum_{C \in \mathcal{C}} \min_{\mathcal{S} \subseteq \mathcal{B}} |C \bigtriangleup \cup \mathcal{S}| = 0?$$

Using the fact that the Problem 5 is $\mathcal{NP}$-complete [GJ79, page 222], it is quite straightforward to show that the dDBP is $\mathcal{NP}$-hard.

Lemma 2.9. *The decision version of the Discrete Basis Problem (Problem 4) is $\mathcal{NP}$-hard.*

Proof. The Set Basis Problem is clearly a special case for the dDBP (a case when $a = 0$) and thus the dDBP is at least as hard as the Set Basis Problem. The Set Basis Problem, on the other hand, is an $\mathcal{NP}$-complete problem. ■

To prove the $\mathcal{NP}$-completeness of the dDBP we still have to prove that it is in class $\mathcal{NP}$.

Lemma 2.10. *The decision version of the Discrete Basis Problem is in class $\mathcal{NP}$.*

Proof. Consider the collections $\mathcal{B}$ and $\mathcal{S}_C \subseteq \mathcal{B}$ for each $C \in \mathcal{C}$. Now the overall size of the collections $\mathcal{S}_C$ is at most quadratic to the size of the $\mathcal{C}$. Thus, it can be decided in polynomial time (with respect to the size of the $\mathcal{C}$) whether

$$\sum_{C \in \mathcal{C}} |C \triangle \cup \mathcal{S}_C| \leq a. \qquad \blacksquare$$

Lemmas 2.9 and 2.10 prove the following theorem.

Theorem 2.11. *The decision version of the Discrete Basis Problem is $\mathcal{NP}$-complete.*

The $\mathcal{NP}$-completeness of the dDBP indicates that the DBP is probably an intractable problem. Thus, it seems that there is no exact algorithm to solve the problem in polynomial time. This, on the other hand, indicates that one should consider approximation algorithms for the problem. Alas, the DBP does not seem to be easy to approximate. To prove this, we first need the following lemma.

Lemma 2.12. *Given a matrix $\mathbf{C} \in \{0,1\}^{n \times d}$ and a matrix $\mathbf{B} \in \{0,1\}^{k \times d}$, we can determine in polynomial time whether there exists a matrix $\mathbf{S} \in \{0,1\}^{n \times k}$ such that*

$$\mathbf{C} = \mathbf{S} \otimes \mathbf{B}.$$

Proof. For each $i = 1, \ldots, n$ we have to find a row $\mathbf{s}_{i\cdot}$ to matrix $\mathbf{S}$, i.e., to select which rows from $\mathbf{B}$ (basis vectors) are used to cover a row $\mathbf{c}_{i\cdot}$ in $\mathbf{C}$. To do that, it is enough to check all basis vectors and select only those that do not cover any 0s in $\mathbf{C}$. This can be done in $O(dk)$ time, and, thus, selecting all rows to $\mathbf{S}$ can be done in $O(dkn) \leq O(|\mathbf{C}|^2)$ time.

To check that all 1s are covered, we compute the Boolean product of $\mathbf{S}$ and $\mathbf{B}$ and compare it to the $\mathbf{C}$. This can be done in $O(dkn) \leq O(|\mathbf{C}|^2)$ time, which also concludes the proof. ■

To study the quality of approximation, we need some sort of measure. A commonly used measure is the *approximation factor*. Recall, that an optimization problem $\mathscr{P}$ is characterized by the quadruple $(I_{\mathscr{P}}, \mathrm{SOL}_{\mathscr{P}}, \ell_{\mathscr{P}}, \mathrm{goal}_{\mathscr{P}})$, where $\ell_{\mathscr{P}}$ is the loss function providing the value of solution $y \in \mathrm{SOL}_{\mathscr{P}}$. The *optimal solution* $y^* \in \mathrm{SOL}_{\mathscr{P}}(x)$ of optimization problem $\mathscr{P}$ with input instance x is such a solution that, for any other solution $y \in \mathrm{SOL}_{\mathscr{P}}(x)$, $\ell_{\mathscr{P}}(x, y^*) \leq \ell_{\mathscr{P}}(x, y)$ for minimization problems, or $\ell_{\mathscr{P}}(x, y^*) \geq \ell_{\mathscr{P}}(x, y)$ for maximization problems. We denote the value $\ell_{\mathscr{P}}(x, y^*)$ by

$\ell^*_{\mathscr{P}}(x)$. In the definition of the approximation factor we use the following convention: if a is a non-negative integer, then

$$\frac{a}{0} = \begin{cases} 1 & \text{if } a = 0, \\ \infty & \text{otherwise.} \end{cases}$$

This convention is needed for the case when the optimal solution for the optimization problem is 0. The approximation factor is defined as follows.

Definition 2.13 (approximation factor [ACG⁺03, p. 90]). Given an optimization problem $\mathscr{P}$, for any instance x of $\mathscr{P}$ and for any feasible solution y of x, the *approximation factor* of y with respect to x is defined as

$$r_{\mathscr{P}}(x, y) = \max\left\{ \frac{\ell_{\mathscr{P}}(x, y)}{\ell^*_{\mathscr{P}}(x)}, \frac{\ell^*_{\mathscr{P}}(x)}{\ell_{\mathscr{P}}(x, y)} \right\}.$$

Theorem 2.14. *The Discrete Basis Problem cannot be approximated in polynomial time within any constant factor unless $\mathcal{P}$=$\mathcal{NP}$.*

Proof. The key idea of the proof is to show that if the DBP could be solved in polynomial time within some approximation factor, then the Set Basis Problem (Problem 5) could be solved exactly in polynomial time. Let $\mathscr{A}$ be some approximation algorithm for Problem 3, $\langle U, \mathcal{C}, k\rangle$ be the input for it and $\mathcal{B} = \mathscr{A}(U, \mathcal{C}, k)$ be the answer of $\mathscr{A}$ with that input. Now let $R_{\mathscr{A}}$ denote the approximation factor $r_{\text{DBP}}(U, \mathcal{C}, k, \mathcal{B})$ for algorithm $\mathscr{A}$ with given input. For contradiction, let us assume that $R_{\mathscr{A}} \neq \infty$, i.e., $R_{\mathscr{A}} \in \mathbb{Q}$.

If the answer to the Set Basis Problem with input $\langle U, \mathcal{C}, k\rangle$ is "yes", then in the optimal solution for the DBP the loss function $\ell_{\triangle}$ is equal to 0. By the definition of the approximation factor (and the convention used within), $R_{\mathscr{A}} \neq \infty$ if and only if $\ell_{\triangle}(\mathcal{C}, \mathscr{A}(U, \mathcal{C}, k)) = 0$. On the other hand, if the answer for the Set Basis Problem with that input is "no", then the approximation algorithm $\mathscr{A}$ will clearly return a basis that makes nonzero error. Thus the approximation algorithm $\mathscr{A}$ will give an answer that has zero error if and only if the answer to the Set Basis Problem is "yes". Lemma 2.12 shows that the answer to the Set Basis Problem can be solved from the $\mathscr{A}$'s answer within a polynomial time. However, if $\mathcal{P}$≠$\mathcal{NP}$, this is a contradiction and, thus, no $R_{\mathscr{A}} \in \mathbb{Q}$ can exists. ■

Theorems 2.11 and 2.14 indicate that the Discrete Basis Problem is somewhat a hard problem. But what about the complexity of the Basis Usage Problem? It clearly is in class $\mathcal{NP}$, and in the special case when we want exact cover it is polynomially solvable (Lemma 2.12). However, it is not know whether it is $\mathcal{NP}$-hard. Should the problem be $\mathcal{NP}$-hard, it still is possible to solve it in feasible time in many cases. This is due to the fact that it can be solved with brute force method by just enumerating over all 2^k different ways to use basis vectors for each vector in input. Therefore, for each fixed k, the Basis Usage Problem is polynomially solvable and, thus, it can be solved within feasible time if k is small enough. In general, problems that can be solved in polynomial time with respect to the input size if the parameter is fixed are said to be *fixed-parameter tractable* problems, as defined by Downey and Fellows [DF99, p. 25]. In that sense, the Basis Usage Problem is at least fixed-parameter tractable problem.

2.3 The disjoint version

We will conclude this section by presenting important modifications of the DBP, namely the *Discrete Basis Partition Problem* (DBPP) and the *Disjoint Discrete Basis Problem* (DDBP). The Disjoint Discrete Basis Problem is the DBP with added requirement that the basis sets (in notation of Problem 3) must be mutually disjoint. In the notation of Problem 1 this is analogous to the requirement, that each column in input set can belong in at most one basis vector. In the Discrete Basis Partition Problem the basis sets must be the partition of universe U, i.e., the basis sets must be disjoint and the union over them must be the universe U. Analogously, for the DBPP the columns in the input matrix must belong in exactly one basis vector.

The formal definitions of the Disjoint Discrete Basis Problem and Discrete Basis Partition Problem are as follows.

Problem 6 (Disjoint Discrete Basis Problem). Given a finite set U (a universe) and a collection $\mathcal{C}$ of subsets of U and a positive integer $k < \min\{|\mathcal{C}|, |U|\}$, find a collection $\mathcal{B}$ of k disjoint subsets of U such that $\mathcal{B}$ minimizes

$$\ell_{\triangle}(\mathcal{C}, \mathcal{B}) = \sum_{C \in \mathcal{C}} \min_{\mathcal{S} \subseteq \mathcal{B}} |C \triangle \cup \mathcal{S}|.$$

Problem 7 (Discrete Basis Partition Problem). Given a finite set U (the universe) and a collection $\mathcal{C}$ of subsets of U and a positive integer $k < \min\{|\mathcal{C}|, |U|\}$, find a collection $\mathcal{B}$ of k subsets of U such that $\mathcal{B}$ is a partition of U and it minimizes

$$\ell_{\triangle}(\mathcal{C},\mathcal{B}) = \sum_{C\in\mathcal{C}} \min_{\mathcal{S}\subseteq\mathcal{B}} |C \triangle \cup\mathcal{S}|.$$

Similarly with the Discrete Basis Problem, the above problems do not ask for finding the collections $\mathcal{S}$ in order to minimize the loss function. However, the Disjoint Basis Usage Problem, i.e., the Basis Usage Problem where all basis sets are disjoint, can be solved in polynomial time, as proved in the following lemma.

Lemma 2.15. *The Basis Usage Problem where all basis sets are disjoint can be solved optimally in polynomial time.*

Proof. Since basis sets are disjoint, at most one set can be used to cover certain point in some input set. Thus, for each input set and for each basis set, we can check if at least half of the points in the selected basis set also belong to the selected input set. If this is the case, then we use this basis set to cover this input set and continue with the next basis set.

It is straightforward to see that after we have iterated over all input sets and all basis sets, we have the optimal collections $\mathcal{S}$ for each input set. The time we need is also clearly polynomial. ■

However, it is not evident whether DDBP or DBPP is any easier than DBP. Adding new requirements to the problem may make it even harder. In this case, the DBPP is easier problem in some sense. But before going further with that subject, we must consider a couple of other problems. Fist of them is the *Metric k-median Problem.*

Problem 8 (Metric k-median Problem). Given a metric space (X, d), a finite set $C \subseteq X$ and a positive integer $k < |C|$, find a set $M = \{\mu_1, \ldots, \mu_k\} \subseteq C$ and a partition $\mathcal{D} = \{D_1, \ldots, D_k\}$ of C such that M and $\mathcal{D}$ minimize the loss function

$$\ell_d(M, \mathcal{D}) = \sum_{j=1}^{k} \sum_{x\in D_j} d(x, \mu_j). \tag{2.16}$$

For the Metric k-median Problem, the objective is to minimize the sum of distances to the corresponding point μ (point μ_i is sometimes referred as a *median* of cluster D_i). Papadimitriou [Pap81] proved that the decision version of the Metric

k-median Problem on Euclidean plane is $\mathcal{NP}$-complete.[1] It is also know that the Metric k-median Problem can be approximated within a constant factor [AGK+04].

The another problem we need to consider here is a variation of the Metric k-median Problem, the *Geometric k-median Problem.*

Problem 9 (Geometric k-median Problem). Given a metric space (X, d), a finite set $C \subseteq X$ and a positive integer $k < |C|$, find a set $M = \{\mu_1, \ldots, \mu_k\} \subseteq X$ and a partition $\mathcal{D} = \{D_1, \ldots, D_k\}$ of C such that M and $\mathcal{D}$ minimize the loss function

$$\ell_d(M, \mathcal{D}) = \sum_{j=1}^{k} \sum_{x \in D_j} d(x, \mu_j).$$

The only difference between the Metric k-median Problem and Geometric k-median Problem is that in the former the set M must be a subset of input points C, while in the latter the set M can be arbitrary subset of the space X. The decision version of the Geometric k-median Problem is known to be $\mathcal{NP}$-hard for L_1- and L_2-metrics in a real plane $\mathbb{R}^2$ [MS84].

Theorem 2.17. *The decision version of the Geometric k-median Problem in Boolean space $\{0,1\}^d$ with L_1-metric is $\mathcal{NP}$-complete.*

Proof (sketch). The problem is trivially in $\mathcal{NP}$: the set M and partition $\mathcal{D}$ together form the required polynomial witness.

Megiddo and Supowit [MS84] proved that the Geometric k-median Problem is $\mathcal{NP}$-hard in $\mathbb{R}^2$ with L_1-metric by reducing the well-known 3-satisfiability problem to it. The construction of the reduction by Megiddo and Supowit can be altered such that all points are in the positive integer plane $\mathbb{Z}_+^2$, and that the maximum distance between any two points is at most polynomial with respect to the input size (number of variables, that is). Furthermore, we can transform the points in $\mathbb{Z}_+^2$ such that the smallest coordinate of any point is 0 and, thus, the largest coordinate is polynomially bounded. Let us denote the largest coordinate of any of these points by N.

Rest of this proof uses *embeddings.* Embeddings are mappings from one metric space to another such that the distances between points are preserved. There is a simple and well-known embedding from the space $(\mathbb{Z}_+^2, \|\cdot\|_1)$, with largest coordinate being

[1]Earlier, Kariv and Hakimi [KH79] had proved that with metric not Euclidean, but induced by a graph, this problem is also $\mathcal{NP}$-complete.

N, to the space $(\{0,1\}^{2N}, \|\cdot\|_1)$: wrote the coordinates in unary. Because N was polynomially bounded, this embedding is polynomially computable.

By at first reducing the 3-satisfiability problem to the Geometric k-median Problem in space $(\mathbb{Z}_+^2, \|\cdot\|_1)$ using the altered reduction, and then using the above embedding, we can reduce the 3-satisfiability problem to the Geometric k-median Problem in space $(\{0,1\}^d, \|\cdot\|_1)$, thus proving that it is $\mathcal{NP}$-hard. ■

The following lemma shows that not only the definitions, but also the answers of the Metric and Geometric k-median Problems are near to each other.

Lemma 2.18. *With same input the optimal solution of the Metric k-median Problem is at most twice as large as is the optimal solution for the Geometric k-median Problem.*

Proof. Let (X, d) be an arbitrary metric space and let x be an arbitrary point in the set of input points $C \subseteq X$. Let $\mu_G \in X$ and $\mu_M \in C$ be the points in the set M that minimize the loss function for geometric and metric versions, respectively. Now $d(\mu_M, \mu_G) \leq d(x, \mu_G)$ by definition (if it does not hold, then the point x should be in the set M instead of the μ_M). It follows that

$$d(x, \mu_M) \leq d(x, \mu_G) + d(\mu_G, \mu_M) \leq 2d(x, \mu_G),$$

where the first inequality is due to the triangle inequality. The lemma follows directly from above inequality. ■

A clear consequence of the above lemma is that we can approximate the Geometric k-median Problem with approximation factor that is twice the approximation factor of the Metric k-median Problem. From now on, we only consider the Geometric k-median Problem in Boolean space $(\{0,1\}^n, \|\cdot\|_1)$, where $\|\cdot\|_1$ is a standard n-dimensional L_1-metric. We call this problem as *Boolean Geometric k-median Problem.*

We would not gain any additional advantage if we would allow the points in the set M to belong to $\mathbb{R}^n$ instead of $\{0,1\}^n$ if we just require that the input set C is a subset of $\{0,1\}^n$. The following lemma explains why.

Lemma 2.19. *Given a set D of points in $\{0,1\}^d$, selecting a point μ that minimizes the sum*

$$\sum_{x \in D} \|x - \mu\|_1$$

can be done by selecting all coordinate values in μ to be the coordinate-wise majority of the points in D.

Proof. Without the loss of generality, we can assume that $d = 1$. Let $n = |D|$, and $o = \sum_n x$, i.e., o is the number of 1s. Consider the case, when $o > n/2$ and thus $\mu = 1$ (the opposite direction is symmetric). For a contradiction, we will assume that for some $r \in R$ we have

$$\sum_{i=1}^{n} |x_i - r| < \sum_{i=1}^{n} |x_i - \mu|.$$

It is clear that any r with $r > 1$ or $r \leq 0$ does not satisfy above inequality. If $r = 1$, the sums are equal. If $0 < r < 1$, the sum is

$$\begin{aligned}\sum_{i=1}^{n} |x_i - r| &= o(1-r) + (n-o)r = o + r\underbrace{(n-2o)}_{<0} = o - r(2o-n) \\ &\geq n - o = \sum_{i=1}^{n} |x_i - \mu|,\end{aligned}$$

which is a contradiction and thus completes the proof. ■

We can now turn our considerations back to the previously stated claim, that the DBPP is, in some sense, easier problem than the general Discrete Basis Problem. In what follows, we prove that the DBPP, unlike the DBP, can be approximated with constant approximation factor. Furthermore, we show that the DBPP and the Geometric k-median Problem are very much the same problem.

Definition 2.20. Given two optimization problems $\mathscr{A} = (I_{\mathscr{A}}, \mathrm{SOL}_{\mathscr{A}}, \ell_{\mathscr{A}}, \mathrm{goal}_{\mathscr{A}})$ and $\mathscr{B} = (I_{\mathscr{B}}, \mathrm{SOL}_{\mathscr{B}}, \ell_{problemB}, \mathrm{goal}_{\mathscr{B}})$, we say that $\mathscr{A}$ and $\mathscr{B}$ are *equivalent with respect to the quality of answer* if

1. $\mathrm{goal}_{\mathscr{A}} = \mathrm{goal}_{\mathscr{B}}$;
2. there exists a polynomially computable one-to-one and onto mapping φ from $I_{\mathscr{A}}$ to $I_{\mathscr{B}}$;
3. for any $x \in I_{\mathscr{A}}$ there exists a polynomially computable one-to-one and onto mapping ρ from $\mathrm{SOL}_{\mathscr{A}}(x)$ to $\mathrm{SOL}_{\mathscr{B}}(\varphi(x))$; and
4. for any $x \in I_{\mathscr{A}}$ and $y \in \mathrm{SOL}_{\mathscr{A}}(x)$, $\ell_{\mathscr{A}}(x, y) = \ell_{\mathscr{B}}(\varphi(x), \rho(y))$.

Because both φ and ρ are one-to-one and onto, they have well defined total inverse functions φ^{-1} and ρ^{-1}, and also holds true that for any $x \in I_{\mathscr{B}}$ and $y \in \mathrm{SOL}_{\mathscr{B}}(x)$, $\ell_{\mathscr{B}}(x, y) = \ell_{\mathscr{A}}(\varphi^{-1}(x), \rho^{-1}(y))$. It follows, that if problems $\mathscr{A}$ and $\mathscr{B}$ are equivalent w.r.t. the quality of answer and we can solve the problem $\mathscr{A}$ with some approximation factor, we can also solve the problem $\mathscr{B}$ with the same approximation factor using mappings φ and ρ. It naturally holds true also for the other way around, i.e., if we know how to solve $\mathscr{B}$, we can solve $\mathscr{A}$.

Theorem 2.21. *The Discrete Basis Partition Problem and the Boolean Geometric k-median Problem are equivalent with respect to the quality of answer.*

Proof. The part 1 is clear: both problems are minimization problems. The instances of the DBPP are tuples of matrices $\mathbf{C}$ and positive integers k and the instances of the Boolean Geometric k-median Problem are, since the metric space is fixed, tuples of sets of input points and positive integers k. Let us define the mapping φ as follows.

$$\begin{aligned} \varphi\colon \{0,1\}^{n\times d} \times \mathbb{N} &\mapsto \{0,1\}^{d\times n} \times \mathbb{N} \\ \varphi(\mathbf{C}, k) &= \langle \mathbf{C}^T, k \rangle \end{aligned}$$

An $n \times m$ Boolean matrix can be thought as a set of m-dimensional points with cardinality of n. The φ is thus a mapping from the instances of the DBPP to the instances of the Boolean Geometric k-median and it is clearly one-to-one and onto.

For the mappings of the feasible solutions, we assume that the solution of the Discrete Basis Partition Problem also includes the matrix $\mathbf{S}$. This assumption can be made, since as Lemma 2.15 states, we can find the matrix in polynomial time. With this assumption, a feasible solution to the Discrete Basis Partition Problem is a tuple of two matrices, $\mathbf{S}$ and $\mathbf{B}$. For the Boolean Geometric k-median Problem, a feasible solution is a tuple of a set M and a partition $\mathcal{D}$ of C.

We must also assume, that the points in the input set C (for the Boolean Geometric k-median Problem) have some ordering, and we denote the ith point in set C by c_i. Let us now define the mapping ρ as follows.

$$\begin{aligned} \rho\colon \{0,1\}^{k\times d} \times \{0,1\}^{n\times k} &\mapsto \{0,1\}^{k\times n} \times 2^{\{0,1\}^{d\times n}} \\ \rho(\mathbf{S}, \mathbf{B}) &= \langle \mathbf{S}^T, \mathcal{D} \rangle, \end{aligned}$$

where point c_j belongs to the cluster $D_i \in \mathcal{D}$ if $(\mathbf{B})_{ij} = 1$. The notation 2^A means the power set of the set A. The mapping ρ thus maps the matrix $\mathbf{S}$ to the set M and the matrix $\mathbf{B}$ to the collection $\mathcal{D}$, both of which can be represented as matrices.

It is evident that ρ is one-to-one and onto. The intuition behind the given mappings is that, broadly speaking, the instances and feasible solutions of the Boolean Geometric k-median Problem are the transposes of those of the Discrete Basis Partition Problem. Now we have proved the parts 2 and 3 of the Definition 2.20.

We still need to prove the part 4 of the Definition 2.20 in order to complete our proof. For that, let $\langle \mathbf{C}, k \rangle$ be an arbitrary instance of the DBPP and let $\varphi(\mathbf{C}, k) = \langle C, k \rangle$ be its mapping to an instance of the Boolean Geometric k-median Problem. Let $\langle \mathbf{S}, \mathbf{B} \rangle$ be a feasible solution for given input and $\rho(\mathbf{S}, \mathbf{B}) = \langle M, \mathcal{D} \rangle$ be its mapping.

We want to prove that

$$\|\mathbf{C} - \mathbf{S} \otimes \mathbf{B}\|_1 = \ell_1(M, \mathcal{D}),$$

where ℓ_1 is the loss function of the Boolean Geometric k-median Problem. Consider an arbitrary element c_{ij} of matrix $\mathbf{C}$ and the corresponding element $(\mathbf{S} \otimes \mathbf{B})_{ij}$. There are four alternatives:

1. $c_{ij} = 0$ *and* $(\mathbf{S} \otimes \mathbf{B})_{ij} = 0$: In this case the loss function for the DBPP is not increased. However, since $(\mathbf{S} \otimes \mathbf{B})_{ij} = 0$, for each l at least one of elements s_{il} and b_{lj} must be 0. If $b_{lj} = 0$, then in the Boolean Geometric k-median Problem the input point in question is not included in cluster D_l, and thus the function ℓ_1 is not increased. If, for some l, $b_{lj} = 1$, then $s_{il} = 0$, i.e., the ith position in vector $\boldsymbol{\mu}_l$ (where we identify points as vectors) is 0. Since also the ith position in jth input point is 0, ℓ_1 is not increased in this case either.

2. $c_{ij} = 1$ *and* $(\mathbf{S} \otimes \mathbf{B})_{ij} = 1$: In this case, by definition of the DBPP, there is exactly one l for which $b_{lj} = s_{il} = 1$. That is, the corresponding input point is in cluster l, but the ith position of the vector $\boldsymbol{\mu}_l$ is equal with the ith position of the point and thus nothing is added to the loss function ℓ_1. If, on the other hand, $b_{lj} = 0$, the case is similar with the previous part.

3. $c_{ij} = 1$ *and* $(\mathbf{S} \otimes \mathbf{B})_{ij} = 0$: Now the loss function of the DBPP is increased by 1. However, again by definition of the DBPP, there must be exactly one l such that $b_{lj} = 1$. Thus, for that l, $s_{il} = 0$, i.e., ℓ_1 is increased by 1. Other values of l can be neglected as in previous cases.

4. $c_{ij} = 0$ *and* $(\mathbf{S} \otimes \mathbf{B})_{ij} = 1$: This case is similar with the previous one.

Because the values of $\|\mathbf{C} - \mathbf{S} \otimes \mathbf{B}\|_1$ and $\ell_1(M, \mathcal{D})$ are equal in an arbitrary point of an arbitrary input, they are always equal. Thus we have also proved the final part of the Definition 2.20 and the proof is complete. ■

The above theorem gives us two straightforward corollaries. The first corollary justifies our claim that the DBPP is, in some sense, easier problem than the DBP, i.e., that the DBPP can be approximated with constant factor.

Corollary 2.22. *The Discrete Basis Partition Problem can be approximated with approximation factor $2R$, where R is the approximation factor for the Metric k-median Problem.*

Finally, Theorems 2.17 and 2.21 yield the following corollary.

Corollary 2.23. *The decision version of the Discrete Basis Partition Problem is $\mathcal{NP}$-complete.*

Example 2.24 (the DBPP and Boolean Geometric k-median Problem). An example of the mappings φ and ρ can be seen in Figure 2. When $k = 3$, the optimal basis vectors are the top three rows in matrix $\mathbf{C}$, which are also the rows of the matrix $\mathbf{B}$. The basis vectors clearly fulfills the requirements of the Discrete Basis Partition Problem. It is also easy to see, that the error is $\|\mathbf{C} - \mathbf{S} \otimes \mathbf{B}\|_1 = 1$. Below the matrices are the corresponding mappings using φ for $\mathbf{C}$ and ρ for $\mathbf{S}$ and $\mathbf{B}$. The points in set M are first, third and fourth points in set C. The clusters in collection $\mathcal{D}$ are such that the first and the last point make up two singleton clusters and the middle points are in one cluster. Finally, it is easy to verify that $\ell_1(M, \mathcal{D}) = 1$. □

While at least something is known about the computational complexity and approximability of the Discrete Basis Partition Problem, virtually nothing is known about the complexity or approximability of the Disjoint Discrete Basis Problem (Problem 6). Its computational complexity is an open question, and the same is true also for its approximability. Intuitively, the DDBP lies somewhere in between the DBP and DBPP, but, as always, the intuition may be wrong.

3 Related work

Despite the fact that the Discrete Basis Problem is a new problem formulation, lots of related work has been done in the field of computer science. Boolean matrices are probably among the most frequently used data structures and lots of research about them has already be done. The DBP is also closely related to the problem of matrix decomposition, which is an another well studied problem. This relation

$$\underset{\mathbf{C}}{\begin{pmatrix}1&0&0&0\\0&1&1&0\\0&0&0&1\\1&1&0&1\\0&1&1&1\end{pmatrix}} \qquad \underset{\mathbf{S}}{\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\\1&0&1\\0&1&1\end{pmatrix}} \qquad \underset{\mathbf{B}}{\begin{pmatrix}1&0&0&0\\0&1&1&0\\0&0&0&1\end{pmatrix}}$$

$$\underset{C}{\begin{Bmatrix}(1,0,0,1,0),\\(0,1,0,1,1),\\(0,1,0,0,1),\\(0,0,1,1,1)\end{Bmatrix}} \qquad \underset{M}{\begin{Bmatrix}(1,0,0,1,0),\\(0,1,0,0,1),\\(0,0,1,1,1)\end{Bmatrix}} \qquad \underset{\mathcal{D}}{\begin{Bmatrix}\{(1,0,0,1,0)\}\\\{(0,1,0,1,1),\\(0,1,0,0,1)\}\\\{(0,0,1,1,1)\}\end{Bmatrix}}$$

Figure 2: Above are example input matrix $\mathbf{C}$ and optimal matrices (with $k = 3$) $\mathbf{S}$ and $\mathbf{B}$ for it. Below are the corresponding sets C and M and collection $\mathcal{D}$.

is addressed in Sections 3.1 and 3.2. The vast number of related work stresses the importance of the Discrete Basis Problem and its variations.

This section is organized as follows. Section 3.1 briefly explains the idea of one well-known matrix decomposition method called *singular value decomposition*. Then, in Section 3.2, we study the methods that are jointly referred as discrete Principal Component Analysis and see why these methods—despite the apparent similarity of their goals—might not fit well to solve the DBP. In Section 3.3, we redefine the DBP using bipartite graphs and bicliques. Then we make a sort sketch about problems and results relating to the bicliques and DBP. Finally, Section 3.4 explains the relations between the DBP and some problems in the field of data mining.

3.1 Singular Value Decomposition

Singular value decomposition (SVD) is one of the best-known methods to decompose a matrix. In matrix decomposition, a given matrix $\mathbf{A}$ is (approximately) represented using a matrix product of many matrices. In SVD, a matrix $\mathbf{A}$ is decomposed to three matrices $\mathbf{U}$, $\mathbf{D}$ and $\mathbf{V}$ such that [GVL96, p. 70]

1. $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$,
2. Matrices $\mathbf{U}$ and $\mathbf{V}$ have *orthogonal* columns and $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$, and
3. $\mathbf{D}$ is a diagonal matrix whose values are the *singular values* of the original matrix ordered so that the uppermost value is largest.

The representation $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ does not, in general, save any space, but using this representation we can form approximate representations by removing all but k uppermost rows from matrix $\mathbf{D}$ (and thus also all but k leftmost columns from matrices $\mathbf{U}$ and $\mathbf{V}$). Furthermore, this representation minimizes the *squared reconstruction error* over all rank k matrices[2] [GVL96, p. 72], that is, if $\mathbf{A}_k$ is the product of modified matrices $\mathbf{U}_k$, $\mathbf{D}_k$ and $\mathbf{V}_k^T$, then

$$\min_{\operatorname{rank}(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_2^2 = \|\mathbf{A} - \mathbf{A}_k\|_2^2.$$

There are immediate similarities between the SVD and the versions of the DBP. The product $\mathbf{U}_k\mathbf{D}_k$ corresponds to the matrix $\mathbf{S}$ and the matrix $\mathbf{V}_k^T$ corresponds to the matrix $\mathbf{B}$. The rows of the matrix $\mathbf{V}_k^T$ must be orthogonal, and this is also the case in the Disjoint Discrete Basis Problem. Indeed, the DDBP can be viewed as an analogue of the SVD for the commutative monoid $(\{0,1\}, \vee)$.

3.2 Discrete PCA techniques

We now turn our considerations on matrix decomposition methods that are based on statistical analysis. A *generative model* is an important part of these methods. Broadly speaking, a generative model of data explains how the data is supposed to be generated. These methods make a heavy use of this a priori assumption. To ease the comparison between the Discrete Basis Problem and the problem statement of these methods, we now shortly define the generative model of the Discrete Basis Problem and its variants as follows.

1. Generate k random Boolean basis vectors $\mathbf{b}_i$. In the Disjoint Discrete Basis Problem and Discrete Basis Partition Problem, generate $\mathbf{b}_i$s such that they obey the conditions given in the problem definition.

2. For each data vector $\mathbf{c}_i$, select randomly a set of basis vectors to construct the data vector.

3. Construct the data using the selections made above.

4. Add some noise, that is, flip the values of some data elements c_{ij}.

[2]For the definition of the *rank* of the matrix, c.f. for example Golub & Van Loan [GVL96, p. 49]

In our generative model, we do not fix the random distributions used. This makes it unsuitable for statistical analysis, but it is only intended to be descriptive. This model is used to generate synthetic test data in Section 5, where also the distributions are fixed. It is easy to see, that in step 1, the generative model generates the matrix $\mathbf{B}$ and in step 2 the matrix $\mathbf{S}$. In step 3, it calculates the product $\mathbf{S} \otimes \mathbf{B}$ to get the data matrix $\mathbf{C}$. In this point, the data matrix $\mathbf{C}$ can be exactly described by matrices $\mathbf{B}$ and $\mathbf{S}$. In the final step, some possible noise is added.

We now turn our considerations on methods known jointly as *discrete principal component analysis*. Principal Component Analysis (PCA) is a well-known and very popular technique for dimensionality reduction [TB99]. Despite their name, discrete PCA methods are not variations of classical PCA, but of *probabilistic principal component analysis* (PPCA) by Tipping and Bishop [TB99]. The PPCA assumes that the data has been affected by Gaussian noise and takes this into account. Like conventional PCA, PPCA can also be used to reduce the dimensionality of the data and, again like PCA, it minimizes the squared reconstruction error of the reduction [TB99]. Since the PPCA assumes Gaussian noise, i.e., continuous variables, it does not work well with discrete data having discrete noise. To overcome this, a variety of methods has been proposed. These methods include multinomial PCA (mPCA) [Bun02], probabilistic Latent Semantic Indexing (pLSI) [Hof99], latent Dirichlet allocation (LDA) [BNJ03] and non-negative matrix factorization (NMF) [LS99], to name a few. All these methods have different backgrounds, so it is rather surprising that, according to Buntine et al., they all are equivalent, ignoring statistical methodology and notation [Bun02, BJ04]. From this on, we will consider only the multinomial PCA as an representative of all these methods, which Buntine and Jakulin refer jointly as *discrete PCA* [BJ04].

The multinomial PCA is more clearly described in the terms of words and documents. That is, given a corpus of documents one is supposed to found a collection of *topics* from those documents. As a simple and naïve example, the corpus could be a collection of scientific articles and the topics could be e.g., different fields of science considered in those articles. We use this notation of words, documents and topics in order to shortly describe the generative model for multinomial PCA [Bun02].

1. In dictionary, there is d words and k topics in total.

2. For each document i, let L_i be the number of words in document.

3. For each document i, let $\mathbf{m}_{i\cdot} = (m_{i1}, \dots, m_{ik})$ be the proportion of topics in

document,

$$\mathbf{m}_{i\cdot} \sim \text{Dirichlet}(\boldsymbol{\alpha}),$$

where $\boldsymbol{\alpha}$ is the parameter vector for the Dirichlet distribution. The Dirichlet distribution is selected here mostly due to its analytical properties.

4. For each document i, let the number of words in document per topic $\mathbf{c}_{i\cdot} = (c_{i1}, \ldots, c_{ik})$, $\sum_{j=1}^{k} c_{ij} = L_i$, be sampled using $\mathbf{m}_{i\cdot}$,

$$\mathbf{c}_{i\cdot} \sim \text{Multinomial}(L_i;\ \mathbf{m}_{i\cdot}).$$

5. Let $\boldsymbol{\Omega} = (\Omega_{ji})$ be the $k \times d$ matrix, where $\boldsymbol{\Omega}_{j\cdot}$ is the proportion of words in topic j.

6. For each document i, let $\mathbf{w}_{j\cdot} = (w_{j1}, \ldots, w_{jd})$ be the number of each word in document from topic j,

$$\mathbf{w}_{j\cdot} \sim \text{Multinomial}(c_{ij};\ \boldsymbol{\Omega}_{j\cdot}).$$

7. For each document i, the number of words in document (the observed data) is thus $\mathbf{r}_{i\cdot} = (r_{i1}, \ldots, r_{id})$, $r_{il} = \sum_{j=1}^{k} w_{jl}$. Now, the $n \times d$ matrix $\mathbf{R} = (r_{il})$ is the document corpus.

With careful interpretation of the above generative model, one can find some similarities with that of the DBP. Indeed, if $\mathbf{R}$ is the input for the DBP, then $\boldsymbol{\Omega}$ is the basis. The DBP of course restricts $\mathbf{R}$ to be a Boolean matrix. Even more, it restricts the $\boldsymbol{\Omega}$ to be a Boolean matrix. These restrictions seem to make discrete PCA methods unsuitable for solving the DBP. Even though the Boolean matrix $\mathbf{R}$ is completely suitable input for mPCA, there is no way to restrict the $\boldsymbol{\Omega}$ to be a Boolean matrix. Actually, in the DBP one word can came from one topic exactly once, i.e.,

$$\max_{\substack{j\in\{1,\ldots,k\}\\ l\in\{1,\ldots,d\}}} \{w_{jl}\} = 1.$$

Because of the multinomial nature of the vector $\mathbf{w}_{j\cdot}$, this is extremely improbable. Also the summation in step 7 should be changed to logical element-wise *or*. Thus it seems that the discrete PCA methods do not fit well for solving the Discrete Basis Problem.

To overcome the problems with multinomial distribution, one could use variations of Bernoulli distributions. An example of this kind of models is the Aspect Bernoulli model by Kabán, Bingham and Hirsimäki [KBH04]. While the Aspect Bernoulli model overcomes the problems with multinomial distribution, it suffers from another problem. In the DBP, a data element c_{ij} is equal to 1 if, ignoring the effects of noise, there exists at least one basis vector $\mathbf{b}_i$ such that $b_{ij} = 1$ and $\mathbf{b}_i$ is used to construct row i in input data. Unfortunately, this is not the case in the Aspect Bernoulli model. In the Aspect Bernoulli model, a data element c_{ij} is equal to 1 if such a basis vector $\mathbf{b}_i$ (called *causes* by Kabán et al.) is selected among all possible k basis vectors, that this basis vector $\mathbf{b}_i$ causes the c_{ij} to be 1 [KBH04]. In other words, for each position j in data vector $\mathbf{c}_i$, a new random choice over all k basis vectors is done, and only the selected basis vector $\mathbf{b}_i$ affects the value of the data point c_{ij}. This is clearly different from what is done in the DBP, and thus it seems that the Aspect Bernoulli model is not suitable for solving the DBP either.

3.3 Analogues to bicliques

Bipartite graphs give an interesting way of defining the DBP. As a reminder, a *bipartite graph* $G = (V, E)$ is such a graph, that the set V of vertices can be partitioned into two distinct sets V_1 and V_2, and all edges in E are between these two sets, i.e., $E \subseteq V_1 \times V_2$.

Definition 3.1. A *biclique* B in a bipartite graph $G = (V_1, V_2, E)$ is a *complete bipartite subgraph* of G induced by a subset of edges $E' \subseteq E$. Thus, if $B = (V_1', V_2', E')$, then

$$\forall v \in V_1', u \in V_2' \colon (v, u) \in E'.$$

Bicliques induced by $E' \subseteq E$ and $E'' \subseteq E$ are *disjoint* if $E' \cap E'' = \emptyset$.

A Boolean $n \times d$ matrix $\mathbf{C}$ defines a bipartite graph G such that $|V_1| = n$, $|V_2| = d$ and $(i, j) \in E$ if and only if $c_{ij} = 1$. Using this mapping, we can formalize the DBP as follows.

Problem 10. Given a bipartite graph $G = (V_1, V_2, E)$ and a positive integer $k < \min\{|V_1|, |V_2|\}$, create a collection of k bicliques $\mathcal{B} = \{B_1, \ldots, B_k\}$, $B_i = (V_1, V_2, E_i)$, minimizing

$$\left| E \mathbin{\triangle} \left(\bigcup_{i=1}^{k} E_i \right) \right| .$$

In the above definition, a set of edges in biclique, E_i, defines a basis vector. The original bipartite graph is approximated using the union of bicliques and the objective is naturally to minimize the difference.

In the literature, the biclique problems are usually studied as a problems of finding *one* biclique from a bipartite or general graph. In a *Bipartite Edge Biclique* problem (BEB) [Hoc98], the objective is to remove as little edges as possible from a bipartite graph such that the remaining graph is a biclique. This problem corresponds to the version of the DBP where $k = 1$ and no 0s can be covered. Even this kind of special version of the DBP is $\mathcal{NP}$-complete [Hoc98]. However, Hochbaum gives a 2-approximation algorithm for it [Hoc98].

A slightly different point of view to this problem is given by the *Maximum Edge Biclique* Problem (MBP) [Pee03]. The MBP asks whether a bipartite graph G has a biclique that has at least K edges. This corresponds to the problem of largest basis vector that does not cover any 0s in the DBP. It is not very surprising that also this problem is $\mathcal{NP}$-complete [Pee03].

As the above problems are only about finding one biclique, they are of little use when trying to solve the DBP. Indeed, they merely prove that even such special cases of the DBP are $\mathcal{NP}$-complete. A much better approach to finding bicliques is given by Mishra, Ron and Swaminathan [MRS03] in their work about conjunctive clustering. The first improvement is that they try to find large subgraphs that are only *almost* bicliques, i.e., there might not be an edge between all vertex pairs. Another improvement is that they also consider the problem of finding a collection of these almost bicliques, and that the bicliques in this collection even may overlap each other. A collection of possibly overlapping almost bicliques is definitely an answer for the Problem 10.

Unfortunately, Mishra, Ron and Swaminathan are using somewhat different approach to the problem and it is not clear, whether or not their definitions characterize a collection of almost bicliques that minimizes the loss function of Problem 10. They are more interested about ensuring that the almost bicliques do not overlap too much and that they are not too much away from being a biclique. Also the overall idea is to find as large almost bicliques as possible [MRS03]. Once again, there is no proof that a big biclique is a good biclique for the DBP. However, large conjunctive clusters do provide an interesting characterization of bicliques that essentially capture the idea of basis vectors in the DBP.

3.4 Analogues to the data mining methods

Data mining is gaining more and more importance in the field of computer science. Numerous methods have been developed in order to find new information from large databases. Many of these methods try to summarize the information that lies in the databases. The Discrete Basis Problem can indeed be seen as a summarization method for binary databases: basis vectors do give some sort of characterization about the data. *Tiling* is a term very closely related to the DBP. A *tile* is actually a biclique, but for the sake of clarity, we will define tiles and tilings using terms more closely related to data mining.

Definition 3.2 (tiles and tilings and their areas). A *tile* of Boolean matrix $\mathbf{D}$ is a bi-set $T = (R, C)$ such that for each $i \in R$ and $j \in C$ we have $c_{ij} = 1$. A tile is, thus, a submatrix full of 1s. A *tiling* is a collection of tiles, $\mathcal{T} = \{T_1, \ldots, T_n\}$. The set of all tilings for a matrix $\mathbf{D}$ is $t(\mathbf{D})$. The *area* of a tile $T = (R, C)$ is $area(T) = |R||C| = |\{(r, c) \mid r \in R, c \in C\}|$. The area of tiling $\mathcal{T} = \{T_1, \ldots, T_n\}$ is

$$area(\mathcal{T}) = \left| \bigcup_{i=1}^{n} \left\{(r, c) \mid r \in R_i, c \in C_i, (R_i, C_i) = T_i\right\} \right| .$$

Even though the concept of tiling is nowadays used in data mining, formerly it has been used in many other connections [GGM04]. The above definition, once again, can be seen as a version of the DBP. A tile (R, C) is a basis vector, where C gives the indexes of 1s in vector and R gives the rows where this basis vector is to be used, i.e., a column in matrix $\mathbf{S}$. The requirement, that the tile must be full of 1s corresponds to the requirement that no 0s can be covered by the basis vector. So the problem of finding the largest tile is $\mathcal{NP}$-complete, as it is same problem as the MBP in previous section.

Geerts, Goethals and Mielikäinen [GGM04] propose a couple of tiling problems and show that they all are $\mathcal{NP}$-hard. These problems include *Maximum k-tiling* Problem and *Top-k Tiles* Problem. In Maximum k-tiling Problem, the objective is to find a tiling $\mathcal{T}$ that consists at most k tiles such that the $area(\mathcal{T})$ is maximum. In Top-k Tiles Problem the objective is to find the k tiles that have the largest area. From these two problems, the Top-k Tiles Problem is somewhat easier. Geerts, Goethals and Mielikäinen propose a branch and bound algorithm to solve it exactly and report that it works efficiently [GGM04]. Unfortunately, in the DBP the size of the basis vector, i.e., the number of 1s in it, does not tell anything about its usefulness. Thus, the answer for the Top k-tiles Problem can hardly be thought as a good candidate answer for the DBP.

The Maximum k-tiling Problem, on the other hand, is a variation of the DBP, where 0s are not allowed to get covered. As such, it is interesting to note the following theorem from Geerts, Goethals and Mielikäinen [GGM04].

Theorem 3.3. *The Maximum k-tiling Problem can be approximated within the factor* $\mathrm{e}/(\mathrm{e}-1)$ *for all values of k simultaneously, given an oracle that finds for any matrix* $\mathbf{D}$ *and tiling $\mathcal{T}$ the tile T such that*

$$area(\mathcal{T} \cup \{T\}) = \max_{T' \in t(\mathbf{D})} area(\mathcal{T} \cup \{T'\}).$$

Despite the need of an oracle, Geerts, Goethals and Mielikäinen report that the algorithm can be implemented reasonably efficiently [GGM04]. It is notable, that this approximation result is not a contradiction with the Theorem 2.14 since the objective functions are different.

The restriction of not covering 0s of course reduce the usefulness of tiling when trying to solve the DBP. Recently, however, Besson, Robardet and Boulicaut [BRB05] proposed a definition of α/β concepts, that also allows a tile to include 0s. An α/β *concept* can be thought as a tile $T = (R, C)$ of matrix $\mathbf{D}$ such that

1. for each row $i \in R$ we have

$$|\{j \in C \text{ such that } d_{ij} = 0\}| \leq \min\{\beta, |C| - 1\};$$

2. for each column $c \in C$ we have

$$|\{i \in R \text{ such that } d_{ij} = 0\}| \leq \min\{\alpha, |R| - 1\};$$

3. for each row $i \in R$, set R includes all identical (w.r.t. the columns in C) rows j of matrix $\mathbf{D}$ and for each column $c \in C$, set C includes all identical (w.r.t. the rows in R) columns j of matrix $\mathbf{D}$; and

4. T is *maximal*, i.e., nothing can be added to the R or C without breaking above requirements.

The above definition is a simplification from the definition of Besson, Robardet and Boulicaut [BRB05], but it will suit well for explaining α/β concepts and their relations to the DBP. The α/β concepts are somewhat an improvement for tiling methods as they allow zeros within tiles. Unfortunately, the user has to make an a priori selection for the values of α and β. In some cases this may be good, but in most cases it is more desirable that the user does not have to make such decisions. The work of Besson, Robardet and Boulicaut is also motivated by issues of data mining and it is highly practically oriented. Much of the possible usefulness of the α/β concepts lie on the algorithm provided by them. This algorithm finds all α/β concepts from a given data [BRB05] and it might be used also to find possible basis vectors.

As a conclusion, tiling is a promising way of formalizing the Discrete Basis Problem. Alas, the theoretical knowledge of it is not very good, and so it will not help much with the DBP. Also the objective of tiling is usually different from the objective of the DBP, e.g., founding one big tile versus founding a good basis.

4 Algorithms

After studying the complexity of the DBP and DBPP and pointing out some related work, we may now turn our attention on solving the problems. The results at Section 2 revealed that both DBP and DBPP are $\mathcal{NP}$-complete problems. Thus we are concentrating on approximation algorithms.

Section 4.1 presents an approximation algorithm for both the Disjoint Discrete Basis Problem and the Discrete Basis Partition Problem. The algorithm is based on the ideas introduced in Section 2.3, i.e., the similarity between the Boolean Geometric k-median Problem and the DBPP. A heuristic algorithm for the general DBP is presented in Section 4.2. It is based on idea about pairwise association rules. Unfortunately no guarantees can be given for the quality of the solution. In fact, we will see an example where the algorithm does not work well.

The heuristic algorithm uses an a priori assumption about the data: the data matrix $\mathbf{C}$ is supposed to be created as described in Section 3.2. Thus the algorithm expects that some basis vectors do exist in the data. This assumption is necessary. If it fails, the heuristic algorithm presented in section 4.2 will certainly not return any meaningful results, as we will see. The algorithm for the DDBP and DBPP is not as dependent upon this kind of assumptions. However, the failure of this assumption do affect the quality of the algorithm's results. Unfortunately, there are no known methods to verify in advance whether this assumption holds.

4.1 An algorithm for the disjoint version

The proofs of Lemma 2.18 and Theorem 2.21 give an idea how to approximate the Discrete Basis Partition Problem by approximating the Metric k-median Problem. Since the Metric k-median Problem is well studied, good approximation algorithms are created for it. Indeed, the first algorithm presented is actually an approximation algorithm for the Metric k-median Problem. For that problem, Jain and Vazirani [JV99] gave an approximation algorithm with approximation factor of 6. Charikar and Guha [CG99] then improved the algorithm to have approximation factor of 4. Both of these algorithms are based on linear programming. Recently, Arya et al. [AGK$^+$04] gave a local search approximation algorithm for the Metric k-median Problem. This algorithm has approximation factor of $3 + 2/p$, where p is the number of simultaneous local swaps.

As described in the proof of Theorem 2.21, the approximation algorithms for the Metric k-median Problem can be used directly to approximate the Discrete Basis Partition Problem. Only the approximation factor is doubled. Unfortunately, this is not the case with the Disjoint Discrete Basis Problem, since there may be some columns that do not belong to any basis vector. For the DDBP, we search for $k + 1$ clusters instead of k. If some columns do not belong in any basis vectors, they should be clustered in one cluster that is located around the origin, i.e., around the vector full of 0s. If none of the found clusters is such located, the algorithm is executed again. This time it searches only for k clusters.

The first implemented algorithm is Arya et al.'s local search algorithm with one simultaneous local swap, thus yielding to approximation factor of 10 for the DBPP. It is not the best known ratio, but the algorithm is straightforward to implement. For the sake of completeness, the algorithm is presented as Algorithm 1. The function Cluster, which is called first time in line 4 creates k clusters by placing each point to cluster whose corresponding point μ is closest to it. The local search condition at line 5 ensures that the algorithm will make local changes as long as it can improve the solution. In line 9 the algorithm selects new medians for the clusters. These medians are not restricted to the input points, and they may make the result better while it is guaranteed—by Lemma 2.19—to be at least the same. Finally, in line 10, the inverse of the one-to-one and onto mapping from the Theorem 2.21 is used to convert the answer of the Geometric k-median Problem to the answer for the DBPP. In the case of the DDBP, the method described above is used to make the solution feasible.

Algorithm 1 Local search algorithm for the DDBP.

```
Input: Matrix $\mathbf{C} \in \{0,1\}^{n \times d}$ for data and positive integer $k < \min\{n, d\}$.
Output: Matrix $\mathbf{B} \in \{0,1\}^{k \times d}$ for basis and matrix $\mathbf{S} \in \{0,1\}^{n \times k}$.
 1: function LocalSearch($\mathbf{C}$, $k$)
 2:     Consider each column vector $\mathbf{c}_{\cdot i}$ in matrix $\mathbf{C}$ as a point $c \in C$
 3:     Select random $M = \{\mu_1, \dots, \mu_k\} \subseteq C$
 4:     $\mathcal{D} \leftarrow$ Cluster$(C, M)$
 5:     while $\exists c \notin M : \ell_1\big((M \backslash \mu_i) \cup c,$ Cluster$((C, (M \backslash \mu_i) \cup c))\big) < \ell_1(M, \mathcal{D})$ do
 6:         $M \leftarrow (M \backslash \mu_i) \cup c$
 7:         $\mathcal{D} \leftarrow$ Cluster$(C, M)$
 8:     end while
 9:     Select new medians $M$ as described in lemma 2.19
10:     return $\rho^{-1}(M, \mathcal{D})$
11: end function
```

Arya et al. [AGK+04] give quite a cumbersome time complexity for this algorithm. We can achieve a much clearer, if not better, running time for our case with a straightforward analysis of Algorithm 1.

Theorem 4.1. *The time complexity of Algorithm 1 is $O(nd^3k)$, where n is the number of rows and d is the number of columns in input matrix $\mathbf{C}$ and k is the number of searched basis vectors with $k < \min\{n, d\}$.*

Proof. It is clear, that the **while** loop from line 5 to line 8 dominates the time complexity of the algorithm. Even more, searching for new local improvements is the most complex part of the algorithm. For each proposed $c \notin M$ the algorithm must calculate the $\ell_{\|\cdot\|_1}$ distance. For that, the algorithm has to calculate the Cluster function. This can be done in $O(dk)$ time. The rest of the $\ell_{\|\cdot\|_1}$ function can also be calculated within $O(dk)$ time. Since $|C| = d$, calculating the condition in line 5 can take at most $O(d^2k)$ time.

To conclude this proof, we must now prove that the Algorithm 1 takes only polynomially many local search steps. For any initial solution, the maximum error is at most $O(nd)$. Since the algorithm improves this solution by at least one in every local search step, it cannot take more than $O(nd)$ steps. Thus the overall time complexity of the algorithm is $O(nd(d^2k) = O(nd^3k)$. ■

The time complexity in Theorem 4.1 is an upper bound. In practice the algorithm runs much faster. The re-selection of μs in line 9 can improve the result, as mentioned. Unfortunately, it is not proven that it also removes the additional factor of

2 from the approximation ratio. Thus the approximation factor of the Algorithm 1 for the DBPP is 10.

4.2 An algorithm for the general case

The general Discrete Basis Problem is a much harder problem than the partition version of it. Due to Theorem 2.14, no approximation results can be given for the DBP. In situations like this, some sort of heuristic algorithm is often a good choice. We made the the preliminary assumption about the model how the data is generated, and we will use this assumption to devise a heuristic algorithm for the DBP.

The idea behind the algorithm is as follows. If some fixed basis vectors were used to create the data, then there should be a strong association between the bits that were in the same basis vector. Thus we will try to found all strong pairwise associations from the data and then we will use these associations to create the basis vectors.

The algorithm for calculating the association rules—the first part of the whole algorithm—is presented as Algorithm 2. It is quite straightforward: The *accuracy* of an association rule $i \Rightarrow j$, $c(i \Rightarrow j)$, for columns i and j is the proportion of the rows where column $i = 1$ and column $j = 1$ of all rows where column $i = 1$, i.e., if column $i = 1$, what is the empirical probability that column $j = 1$. A threshold τ is used to transform the accuracies to Boolean matrix. If the accuracy for $i \Rightarrow j$ is greater than or equal to τ, then element a_{ij} in matrix $\mathbf{A}$ is 1 and otherwise a_{ij} is 0. The matrix $\mathbf{A}$ is called as the *association matrix.* We should note that the accuracies of the rules are not symmetric, i.e., it is possible that $c(i \Rightarrow j) \neq c(j \Rightarrow i)$.

Algorithm 2 An algorithm for the DBP using association rules. Part 1.

```
Input: Matrix $\mathbf{C} \in \{0,1\}^{n \times d}$ for data, positive integer $k < \min\{n, d\}$ and threshold
    value $\tau \in ]0,1]$.
Output: Matrix $\mathbf{B} \in \{0,1\}^{k \times d}$.
1: function ASSOCIATION($\mathbf{C}, k, \tau$)
2:     for all pairs of columns $(i, j)$ in $\mathbf{C}$ do
3:         Calculate association accuracies $c(i \Rightarrow j)$
4:         if $c(i \Rightarrow j) \geq \tau$ then
5:             $a_{ij} \leftarrow 1$                    ▷ $\mathbf{A}$ is the association matrix.
6:         else
7:             $a_{ij} \leftarrow 0$
8:         end if
9:     end for                                          ▷ Continues in part 2.
```

How should these accuracies reveal the basis vectors? Consider a basis vector $\mathbf{b}$ that is used to create the input. Suppose that there is a column i that belongs only in a basis vector $\mathbf{b}$. Now, each time when column i is 1, all other columns that belong to $\mathbf{b}$ are also 1 and thus for all columns j that belong to the vector $\mathbf{b}$, $c(i \Rightarrow j) = 1$. Due to the possible noise, the accuracy is actually $c(i \Rightarrow j) = 1 - \varepsilon$, where ε is the noise ratio. Selecting $\tau = 1 - \varepsilon$, the result is a matrix $\mathbf{A}$ that has the basis vector $\mathbf{b}$ in its row i. If column i belongs to many basis vectors, then row i in matrix $\mathbf{A}$ cannot be used for finding basis vectors.

Example 4.2 (association accuracies). We can see an input matrix $\mathbf{C}$ and corresponding association matrix $\mathbf{A}$ in Figure 3. In association accuracy matrix, an accuracy $c(i \Rightarrow j)$ is an element a_{ij}. Because the input matrix $\mathbf{C}$ can be represented completely with two basis vectors, the threshold τ should be 1. Correct basis vectors are in rows 1 and 2 in matrix $\mathbf{A}$. The third row of the accuracy matrix looks different because the third column belongs to both basis vectors. □

Suppose that the correct basis vectors are among the rows of the association matrix $\mathbf{A}$. There is still the problem of how to find these rows. If there is no noise, this is easy. However, it is unrealistic to expect such a situation. Thus, selecting the correct rows deserves great attention. A greedy heuristic was implemented. It tries to find rows that cover most of the yet uncovered 1s in $\mathbf{C}$. For that, define the *cover* function for covering $\mathbf{c}_{n\cdot}$, the nth row in data, with vector $\mathbf{a}_{i\cdot}$ when $\mathbf{u}_{n\cdot}$ is the vector marking already covered elements, to be

$$\begin{aligned} &\mathit{cover}\colon \{0,1\}^d \times \{0,1\}^d \times \{0,1\}^d \mapsto \mathbb{N} \\ &\mathit{cover}(\mathbf{c}_{n\cdot}, \mathbf{u}_{n\cdot}, \mathbf{a}_{i\cdot}) = \max\left\{ \sum_{j=1}^{d} \Big(\underbrace{a_{ij}c_{nj}(1-u_{nj})}_{\#\text{ of covered 1s}} - \underbrace{(a_{ij} - c_{nj}a_{ij})}_{\#\text{ of covered 0s}} \Big), 0 \right\}. \end{aligned} \tag{4.3}$$

So if $\mathbf{a}_{i\cdot}$ covers more 0s than 1s, then it is not used for covering row $\mathbf{c}_{n\cdot}$. The result of *cover* function is referred to as *cover-value.*

$\begin{pmatrix}1&0&1\\0&1&1\\1&1&1\end{pmatrix}$	$\begin{pmatrix}1&1/2&1\\1/2&1&1\\2/3&2/3&1\end{pmatrix}$	$\begin{pmatrix}1&0&1\\0&1&1\\0&0&1\end{pmatrix}$
$\mathbf{C}$	association accuracies	$\mathbf{A}$ with $\tau = 1$

Figure 3: An example of accuracies.

The greedy heuristic for selecting the basis vectors (and thus the rest of the Algorithm 2) is presented as Algorithm 3. In line 12, the greedy heuristic selects the row of $\mathbf{A}$ that has the highest sum of cover-values over all rows in $\mathbf{C}$. Then, for each row in $\mathbf{C}$, if the cover-value of the selected row in $\mathbf{A}$ is positive for this row in $\mathbf{C}$, it updates the matrix of already covered 1s (line 15). It then iterates this k times and returns the basis matrix $\mathbf{B}$.

We can easily see that the algorithm presented in Algorithms 2 and 3 has polynomial running time. Once again, this proof is not strict, but an upper bound.

Theorem 4.4. *The algorithm presented in Algorithms 2 and 3 can be executed in $O(nd^2k)$ time, where n and d are the number of rows and columns in the input matrix, respectively, and k is the number of basis vectors searched with $k < \min\{n, d\}$.*

Proof. The first part of the algorithm, the Algorithm 2, can be done in $O(nd^2)$ time. The *cover* function from equation (4.3) can be calculated in $O(d)$ time. Updating the row in matrix $\mathbf{U}$ in line 15 can also be done in $O(d)$ time. Thus the **for** loop from line 13 to line 17 can be done in $O(nd)$ time. But since the line 12 takes $O(nd^2)$ time, it dominates the time complexity in Algorithm 3. Line 12 is executed k times and thus the overall time complexity of the Algorithm 3 is $O(nd^2k)$, which is also the asymptotic time complexity of the whole algorithm. ■

There are a couple of drawbacks of this method. The most severe, a real Achilles heel, is the restriction that each basis vector must have some column which belongs only to that basis vector, i.e., there is no other basis vector that has 1 in that column. This is a requirement that cannot be guaranteed for the arbitrary data, nor can it be

Algorithm 3 An algorithm for the DBP using association rules. Part 2.

```
10:     $\mathbf{U} \leftarrow \mathbf{0}$                    ▷ If element $c_{ij}$ is covered, then $u_{ij} = 1$.
11:     for $l \in \{1, \ldots, k\}$ do
12:         Select row $\mathbf{a}_{i\cdot}$ s.t. $\forall j \neq i \sum_n cover(\mathbf{c}_{n\cdot}, \mathbf{u}_{n\cdot}, \mathbf{a}_{i\cdot}) \geq \sum_n cover(\mathbf{c}_{n\cdot}, \mathbf{u}_{n\cdot}, \mathbf{a}_{j\cdot})$
13:         for all rows $n$ in $\mathbf{C}$ do
14:             if $cover(\mathbf{c}_{n\cdot}, \mathbf{u}_{n\cdot}, \mathbf{a}_{i\cdot}) > 0$ then
15:                 $\mathbf{u}_{n\cdot} \leftarrow \mathbf{u}_{n\cdot} \vee \mathbf{a}_{i\cdot}$
16:             end if
17:         end for
18:         $\mathbf{b}_{l\cdot} \leftarrow \mathbf{a}_{i\cdot}$
19:     end for
20:     return $\mathbf{B}$
21: end function
```

checked from the given data. If some basis vector does not meet this requirement, it is highly improbable that some row in association matrix $\mathbf{A}$ corresponds with that basis vector.

Selecting a correct threshold value τ is of course a problem. Usually, the noise ratio is not known a priori. Then a user of the algorithm must rely on their intuition or make an exhaustive search through all meaningful values of τ. Unfortunately, there are even cases when no value of τ yields to best answer, as described in example 4.5.

Example 4.5 (a counterexample for Algorithm 2). Figure 4 presents a counterexample for Algorithm 2. The input matrix $\mathbf{C}$ is the same with that in Figure 1 at page 5. The best basis for matrix $\mathbf{C}$ with $k = 2$ is also shown in Figure 1. As shown in Example 2.2, the optimal basis makes error of 1. But, as it can be seen from the accuracies at Figure 4, no value for τ yields to good solution. If τ is selected as proposed, i.e., $\tau = 1 - \varepsilon$, where ε is the percentage of noise, then $\tau = 1 - 1/9 = 8/9$. In this case, the matrix $\mathbf{A}$ will be 3-dimensional identity matrix, $\mathbf{I}_3$. No matter, which two rows of it will be selected, the minimal error is always 2. This clearly holds for all values $\tau \geq 1/2$. If, on the other hand, $\tau < 1/2$, the matrix $\mathbf{A}$ will be full of 1s, and thus the minimal error is 3. $\square$

As mentioned, usually the only way to select the correct value for the τ is to exhaustively test all meaningful values. Alas, it is not enough to just re-run algorithm with different values of τ. Having just the basis does not reveal, whether or not the current basis is good or whether or not current basis is better than some other basis. To solve the value of the loss function $\ell_{\otimes}$, we must also solve the Basis Usage Problem. Unfortunately, no efficient algorithm is known for that. Thus, to decide which value of τ is correct, we have to perform a brute force search in order to calculate the matrix $\mathbf{S}$. This can be done in feasible time only if number of basis vectors, k, is small enough.

$$\underset{\mathbf{C}}{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}} \qquad \underset{\text{association accuracies}}{\begin{pmatrix} 1 & 1/2 & 1/2 \\ 1/2 & 1 & 1/2 \\ 1/2 & 1/2 & 1 \end{pmatrix}}$$

Figure 4: A counterexample, where accuracies do not work.

5 Experimentation

While the theoretical results form the basis of every good algorithm, pure theory can rarely convince the practitioners. For that, extensive experimentations are needed. The need of experiments is even higher, if theory can say only little about the algorithm's properties—as is the case with the Association algorithm.

The algorithms were tested with both synthetic and real-world data to get better understanding about them. Section 5.1 introduces the test settings and data used. Then Sections 5.2 and 5.3 give the results of tests and interpretation of the results for the LocalSearch and Association algorithms, respectively.

5.1 Test settings

Experiments were done with both synthetic and real-world data. With synthetic data a complete control over the data generation process was achieved and the data properties were well known. The real-world data was used to check if algorithms could produce some sensible results for certain kinds of data.

Synthetic data. For synthetic data, an important part is of course the generative model, i.e., how the data is created. The data generation model had to follow our descriptive version given in Section 3.2. All random selections made were made uniformly at random. The synthetic data were created by first generating the basis vectors. Basis vectors were naturally random, but certain properties were fixed. In order to test the LocalSearch algorithm, a collection of sets of mutually disjoint basis vectors were created. These bases were also used to test the Association algorithm, but in addition another collection of not necessarily disjoint random basis vectors were generated.

The mutually disjoint basis vectors were not forced to fulfill the other requirement of the definition of the Discrete Basis Partition Problem, i.e., that all columns belong to exactly one basis vector. This was done, because we were more interested in studying the properties of the LocalSearch algorithm when applied to the Disjoint Discrete Basis Problem. Recall, that the LocalSearch algorithm has a constant approximation factor for the DBPP but not for the DDBP.

These bases were then used to create matrices that did not have any noise at all. With disjoint bases, two random sets were created using same properties. With arbitrary bases, only one random set per parameter combination was created. In total, 96 noiseless sets with disjoint bases and 32 noiseless sets with arbitrary bases were created. Noiseless matrices were used as references as described later. Finally, a fixed amount of random noise was added to the matrices. There were three types of noise, namely *left*, *right* and *mixed* type, where noise only changed 0s to 1s, 1s to 0s or randomly changed the values, respectively. The amount of noise is reported as percentage of 1s in noiseless matrix, not as percentage of the size of the matrix. This corresponds to the idea that the matrices are collections of sets and that the amount of noise is reported as a percentage of the cardinality of the noiseless set. In total, 1152 sets with disjoint bases and 192 sets with arbitrary bases were created. All fixed properties and values used for creating the synthetic data can be seen in Table 1.

Matrices created from disjoint basis vectors were used as an input for the both LocalSearch and Association algorithms (with one exception, see footnote on Table 1). Matrices created from arbitrary basis vectors were used as an input only for the Association algorithm. The number of basis vectors to search was set to be the number of used basis vectors for the Association algorithm and one more of that for the LocalSearch algorithm (cf. Sections 2.3 and 4.1). The threshold τ for the Association algorithm was set to be $1 - \varepsilon - 0.03$, where ε is again the noise ratio.

To analyze the quality of the results, an approximation of the input matrix was created. With the LocalSearch algorithm, the approximation was created by using the two matrices, $\mathbf{B}$ and $\mathbf{S}$, returned by the algorithm. Then a Boolean matrix product of matrices $\mathbf{B}$ and $\mathbf{S}$ was taken. With the Association algorithm, a brute force method was used to find the best possible matrix $\mathbf{S}$ in order to create the approximation. These approximations were then compared with the original, noiseless matrices and the error, i.e., the L_1-distance, was calculated. To make the errors of different matrices comparable, the calculated error was divided by the number of 1s in the original, noiseless matrix. We call this ratio as *error ratio*. The error ratio may be well over 1. A corresponding term is *percentage of error* which is an error ratio multiplied by 100. The percentage of error is directly comparable with the percentage of noise in input matrix.

Real data. In addition to the experiments made with synthetic data, two real-world databases were also used. The requirement for these databases was, that they must be Boolean by nature, i.e., numbers should not give any more information. This requirement ruled out the usual corpus databases. The information of how many

	Value	
Property	disjoint bases	arbitrary bases
Dimension	100, 1000	100, 1000
Number of basis vectors	10, 15[a]	10, 15
Expected basis vectors density (%)[b]	5, 10	10, 20
Number of rows in data matrix	100, 1000	100, 1000
Expected number of used basis vectors per row in data matrix	3, 5, 7	5, 7
Types of noise used	left, right, mixed	left, right, mixed
Percentage of noise added (%)	5, 10, 15, 20	10, 20

[a]Only 10 basis vectors were used for the ASSOCIATION algorithm.
[b]Reported as percentage of dimensionality.

Table 1: Properties and their values for generating the synthetic data. All different combinations of values were used.

times a certain word is used in document is important since same word may appear many times in same document with different meanings. The two selected databases were *Finnish Bird Atlas* and *Course completion database*.

In Bird Atlas, Finland was partitioned to 3813 squares of side 10km, so called *atlas squares*. Then nesting of different birds in these atlas squares was examined. The survey was conducted during the years 1974–79 and 1986–89. Väisänen, Lammi and Koskimies report the results of survey exhaustively [VLK98]. The data is very reliable, and possible noise in it is, by nature, of type *right*. In total, 248 different bird species nested in these squares. Originally, the probability of a particular species nesting in some atlas square was expressed by four-level confidence value, where the lowest (0) means probably no nesting and the highest (3) almost certain nesting. These values are mapped to Boolean values by mapping the two lowest values (0 and 1) to 0 and the two highest values (2 and 3) to 1. Thus the input matrix for Bird Atlas is 3813×248 Boolean matrix, where each row corresponds to one atlas square and each column to one bird nesting in Finland.

The course completion database consists of information about students at the University of Helsinki (rows) and the courses they have passed (columns). Students included have been accepted to study the Computer Science as a major between the years 1990 and 2004 and have passed at least one course in Computer Science. The student's current status, i.e., current major subject or possible graduation, has not been taken into account. This leads to a Boolean matrix with 2405 rows and 5021 columns. Most of these courses were passed only by few students and thus the matrix is very sparse. For additional examinations, the submatrices of those courses that more than 10 or 30 students have passed were also studied. The sizes of those matrices were 2405×615 and 2405×223, respectively. Some basic parameters for all these matrices are given in Table 2.

The algorithms were written on C in Linux 2.6 environment. All tests were run on normal PC with 3GHz hyper-threading Intel Pentium 4 processor and 1GB of main memory. All tests were made within a day and a night. Longest single run was the Association with the course completion database with all courses and took approximately 15 minutes, while most of the synthetic data tests were ready in approximately one second.

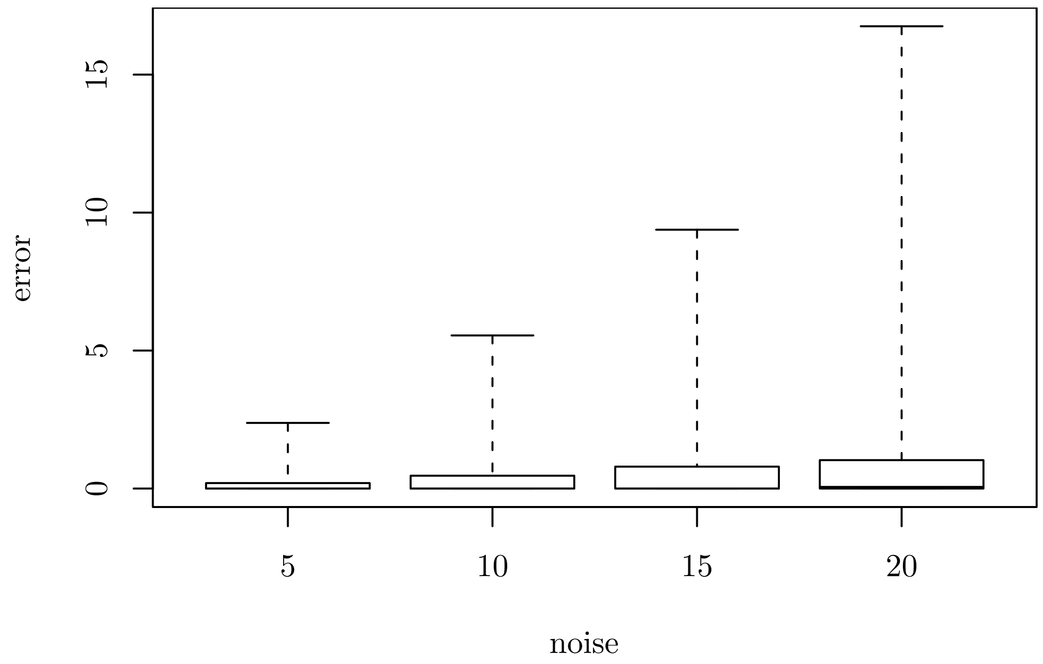
5.2 Results for the LocalSearch algorithm

Experiments made with synthetic data were the most important ones. If the algorithms had failed, it would have been clear, that they do not work well. Synthetic data also reveals some aspects of the algorithms that are very hard, if not impossible, to see from the real-world data experiments.

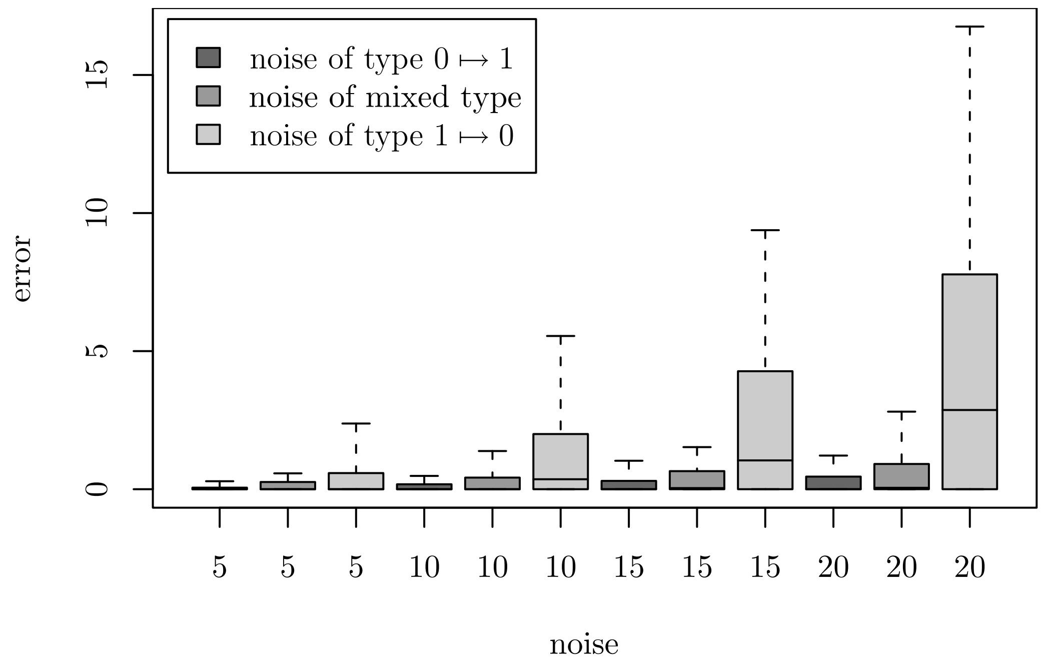
The convergence of the LocalSearch algorithm was tested first. Since the LocalSearch algorithm is a randomized algorithm, it is possible that it gives highly different results for the same data with different executions. Should that happen, all tests made with the LocalSearch algorithm should be repeated many times in order to reduce the effects of randomness. To test the convergence, the LocalSearch algorithm was executed 100 times with the same input and the results were examined. The results were good—all answers gave exactly the same error. Because of good convergence, the LocalSearch algorithm was executed only once per different input data in actual tests.

Synthetic data. For the actual synthetic test data, the results were collected as a statistical data, and no input matrix was examined by itself. When studying the results of tests, we will concentrate on the relation between the noise and the error. As the noise percentage and error percentage are comparable, we will use them as a primary values to examine in the results. We can see some boxplots about the effects of noise to the error in Figure 5.

As we can see from Figure 5(a), the error is always smaller than the noise. Adding noise does increase the error, though. From Figure 5(b), we can see that noise of type *right* has drastic effect on the error, while other types of noise have only minimal



(a) All types of noise together.



(b) Different types of noise separated.

Figure 5: Boxplots showing the correlation between noise percentage and error percentage for the LocalSearch algorithm.

	Data matrix			
Parameter	Bird Atlas	all courses	courses over 10	courses over 30
rows	3813	2405	2045	2045
columns	248	5021	615	223
number of 1s	246551	62667	52739	46602
density	0.261	0.005	0.036	0.087
average number of 1s per row	65	26	22	19
variance of number of 1s per row	1071	365	223	172
average number of 1s per column	994	12	86	209
variance of number of 1s per column	990377	5359	37663	80181

Table 2: Some parameters for the real-world data matrices.

effects. It is unclear why the noise of type *right* has so big effect while the noise of type *left* does not have almost any effect at all; based on the L_1-metrics, both types of noise should have equal effects.

It is not very surprising that the results for the noise of type *mixed* are close to the results for the noise of type *left*. In sparse matrices as our synthetic dataset, random noise mostly flips 0s to 1s, i.e., it creates noise of type *left*. Thus, noise of *mixed* type is more or less noise of *left* type with some additional noise of *right* type.

The results for the LocalSearch algorithm with synthetic data were promising: the effects of noise were limited, and in the case with noise of type *left*, the results were almost as good as possible. The noise of type *right* did have greater influence on results, but the error per cent was still kept under the noise per cent.

Real data. The results for the Bird Atlas database were not as good as it was hoped. The most notable characteristic for the results was one really big basis vector, i.e., really many species belonging to one basis vector. Intuitively the best result was achieved when 11 basis vectors were searched. The total error for this was 96922, leading to a error per cent of 39.31. Results were improved when the number of searched basis vectors were increased. The resulting basis vectors, however, did not seem to be very intuitive.

For 11 basis vectors, the largest basis vector had 124 birds in it, i.e., exactly half of all species, while the smallest basis vector had only one species in it. The number of species in other basis vectors was between 4 and 28. However, the columns in the largest basis vector had only approximately 10% of 1s in the data and the corresponding column in matrix **S** was full of 0s. Thus, the species in that basis vector could be considered as “noise”. This idea has some support from the data, as the largest basis vector included e.g., a Snow Goose and an Arctic Redpoll, which are extreme rare in Finland [VLK98, p. 487 and 512]. On the other hand, the largest basis vector also included Rock Dove, a very popular bird in towns of Finland. The appearance of Rock Dove as a “noise” is explained by the fact that it does not live outside the urban environment [VLK98, p. 248]. As Finland is by most forest and countryside, the number of atlas squares where Rock Dove lives is rather limited.

Another extreme is given by Rustic Bunting, which made up one basis vector by itself. Unfortunately, there seems not to be any good reason for that. The rest of the basis vectors did seem to have some intuition behind them. As an example, the second smallest basis vector consisted of the following species: Greenshank, Wood Sandpiper, Meadow Pipit and Brambling. From those birds, only the last is not a bird living in swamps in northern Finland [VLK98, p. 204, 208, 312 and 444]. However, Brambling is a northern bird [VLK98, p. 444], and thus that basis vector is representing some of the birds nesting in north.

When the number of searched basis vectors was increased, the results were not what was expected. Instead of splitting the “noise” vector, the more meaningful basis vectors were split. As an example, when 50 basis vectors were searched, the largest basis vector still had 111 species in it. The second largest basis vector had only 13 species in it and most of the basis vectors were singletons. The singletons are not very informative as a basis vectors and thus we may consider these results inferior to the ones with 11 basis vectors although the error was decreased to 24.54%.

All bird species in Bird Atlas are classified to 11 classes based on their main nesting environment. The results were expected to follow these classes. This did not happen. A possible explanation on the failure of the expectation is the fact that within 10 kilometer square the type of environment may change dramatically, and thus species nesting in different environments may be find in the same square. The situation may have been completely different if the size of the atlas squares would have been smaller, e.g., one square kilometer. In practice, squares of this size would be highly impractical to survey and thus they are not used. As a conclusion, it seems that the Bird Atlas database does not fit well with the assumption of the initial basis vectors.

The results for the course completion database were quite interesting, albeit not very intuitive. Once again, there was one large basis vector that was never used, and the rest of the basis vectors included only few courses. With one exception: one basis vector always included more courses than others and—most notably—many of these courses were not about Computer Science. As an example, when all 5021 courses were examined, that one basis vector included 29 courses while the average number of courses per basis vector (excluding the "noise" vector) was 5. Those courses included, but were not restricted to, Approbatur in Physics II, Theory of Macroeconomics, Introduction Course to Political Science, Introduction to Psychology and Advanced Course in German 1. The only thing in common between most of these courses seemed to be, that they are courses likely to be find from a curriculum of a student in Faculty of Law; some of the courses were offered by Faculty of Law while many others, e.g., economics, psychology and languages, can be considered as a potential minors for law students. Mielikäinen [Mie05, p. 95] reported the same kind of phenomenon. His experiments were based on tiling, which is very close with the method used, as described in Section 3.4. It seems that this basis vector is due to some students that have started with the Computer Science as a major but have later on changed to study Law as their major (as it is very rare to read law as a minor due to the opposition of Faculty of Law). The basis vectors was used only 33 times which also supports the conclusion we made.

The rest of the basis vectors for all courses were not so surprising. They were made up mostly from courses in Computer Science and Mathematics accompanied with some mandatory courses for students having major in Computer Science (e.g., language courses and Maturity Test in Finnish). Probably the most surprising fact was that the basis vectors were mostly mixtures of courses in Computer Science and Mathematics and that the logical pairs of courses, e.g., course in Data Structures and Data Structures Project, did not appeared together in the same basis vector. The only thing in common with courses in the same basis vector seemed to be their popularity. This was not very surprising, as the L_1-distance between two points is by definition large if there is a big difference in numbers of 1s. The 15 basis vectors discovered by the algorithm did not cover all courses very well: the error was as high as 65.11%.

With courses that at least 10 students have passed, the error (with 15 basis vectors) was slightly smaller, 58.54%. However, this was only due to the decreasing number of 1s in data matrix as the basis vectors (excluding the "noise" vector) were exactly the same as in all courses. This was probably one of the most surprising facts about these results. Even more surprising was the fact that, when considering only courses that over 30 students have passed, the basis vectors were still almost the same. The only exception was the large basis vector with courses in law etc.: some courses were removed from it because not enough students have passed those courses. The error was a little bit smaller decreasing to 53.48%.

When the number of basis vectors was increased to 25 (with courses that over 30 students have passed), the results were slightly different: some basis vectors stayed intact, some were split to many parts and some new courses appeared in new basis vectors. The basis vector with courses in law etc. was still almost as it was before. The error was slightly reduced, as it was 43.62%.

It seems that there are not any logical "basis vectors" among the course completion database. Many of the popular courses made up a basis vector by themselves. When a pair of courses, or even a 3-tuple, appeared as a basis vector, no strong logical connection between them was found. Once again, it seems that the data did not fit well with the idea of basis vectors.

Unfortunately the results for real-world data were not very good. The bases did not cover the data well, nor were the results intuitive. Since the LocalSearch algorithm worked well with the synthetic data, we may assume that the real data used was not very suitable for this kind of data analysis. It is possible that more in-depth study of the results will reveal some aspects that were not recognized by us, though. The numerical results for the experiments can be find at Table 3.

Database	Basis vectors	Error	Error (%)	Not covered 1s	Covered 0s
Birds	11	96922	39.31	64384	32538
Birds	15	91782	37.23	61042	30740
Birds	50	60499	24.54	42985	17514
Courses all	15	40802	65.11	37565	3237
Courses 10	15	30874	58.54	27637	3237
Courses 30	15	24923	53.48	21753	3170
Courses 30	25	20328	43.62	18245	2038

Table 3: Numerical results for the LocalSearch algorithm with the real-world data.

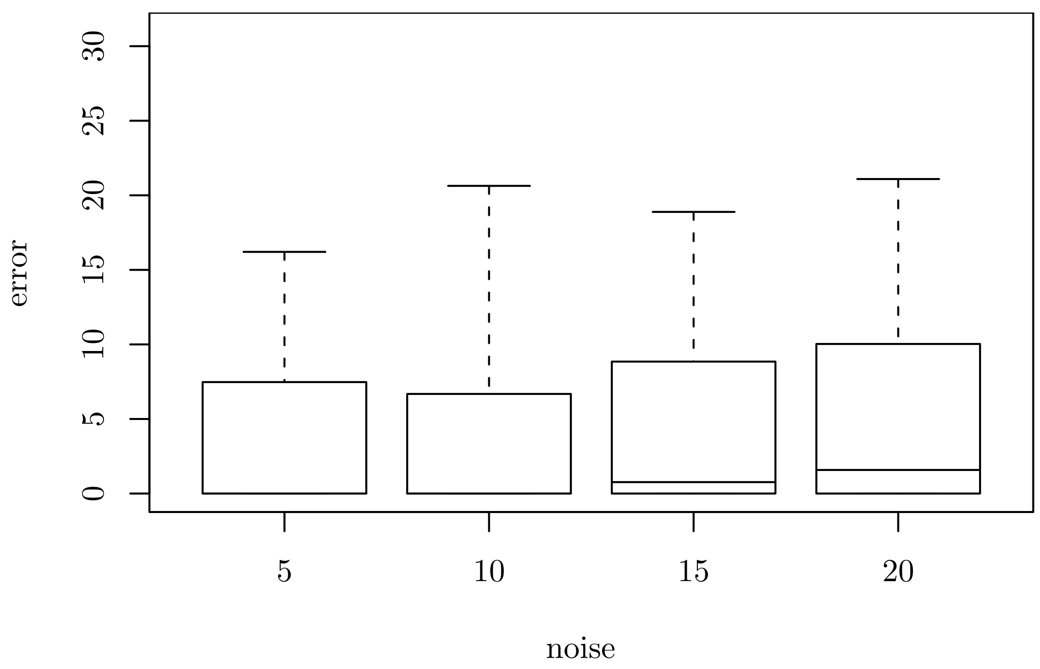
5.3 Results for the Association algorithm

For the Association algorithm, experiments were made with two different sets of synthetic data: one with disjoint bases and other with arbitrary bases. The data with disjoint bases were the same that were used with the LocalSearch algorithm, but only matrices created from 10 basis vectors were used. This was done in order to reduce computational time since an exponential search was made for each resulting basis in order to find out the best possible matrix **S**. As with the LocalSearch algorithm, the results were taken only as statistical data, and no input matrix was examined by itself.

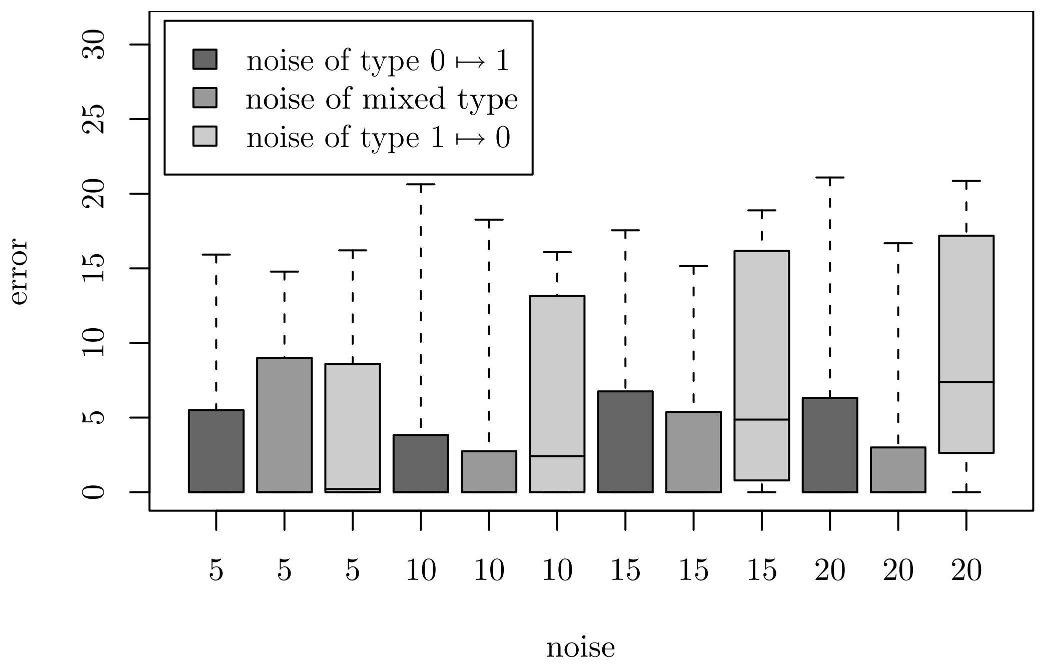
Synthetic data. We can see the results for the Association algorithm with disjoint bases in Figure 6. In Figure 6(a), different types of noise are not separated, whereas in Figure 6(b), different types of noise are separated in their own boxes. The first notable thing from Figure 6(a) is, that the amount of noise does not seem to have effect on the maximum errors. In fact, the maximum error with 15% of noise is smaller than it is with 10% of noise. However, the 3rd quartiles are rising linearly, with the exception of the first box. Also, after the first box (5% of noise), the 3rd quartiles are below the corresponding amount of noise, i.e., three-quarters of results are better than the noise. Even more important is to note that the medians are so close to the 0, that they are not drawn until the box with 15% of noise. And with all boxes, the medians stay under the 5% of error. Although the maxima are quite high for all levels of noise, most of the results are still rather good.

When concentrating on Figure 6(b), the noise level of 5% is an exception. Rather surprisingly, the error per cents for *left* and *mixed* types of noise are greater with 5% of noise than they are with 10% of noise. In fact, for the *mixed* type of noise, the error per cent is greatest with 5% of noise. And for the rest of the boxes, the *mixed* type of noise has the lowest maximum value and 3rd quartile. The only type of noise that increases the error per cent linearly is the type *right*. It also has always the highest 3rd quartile (with the exception of 5% of noise) and median, i.e., the trend is similar with Figure 5(b).

The noise of type *right* has a significant effect on the results, as was also the case with the LocalSearch algorithm. The reasons are probably similar: loosing information has more effects than having some false information. The results for matrices created from disjoint bases cannot be considered as bad results. The medians are always below the corresponding error percentage and even 3rd quartiles are usually below it. Thus most of the results can be considered as good results. Also, for



(a) All types of noise together.



(b) Different types of noise separated.

Figure 6: Boxplots showing the correlation between noise percentage and error percentage for the Association algorithm with disjoint bases.

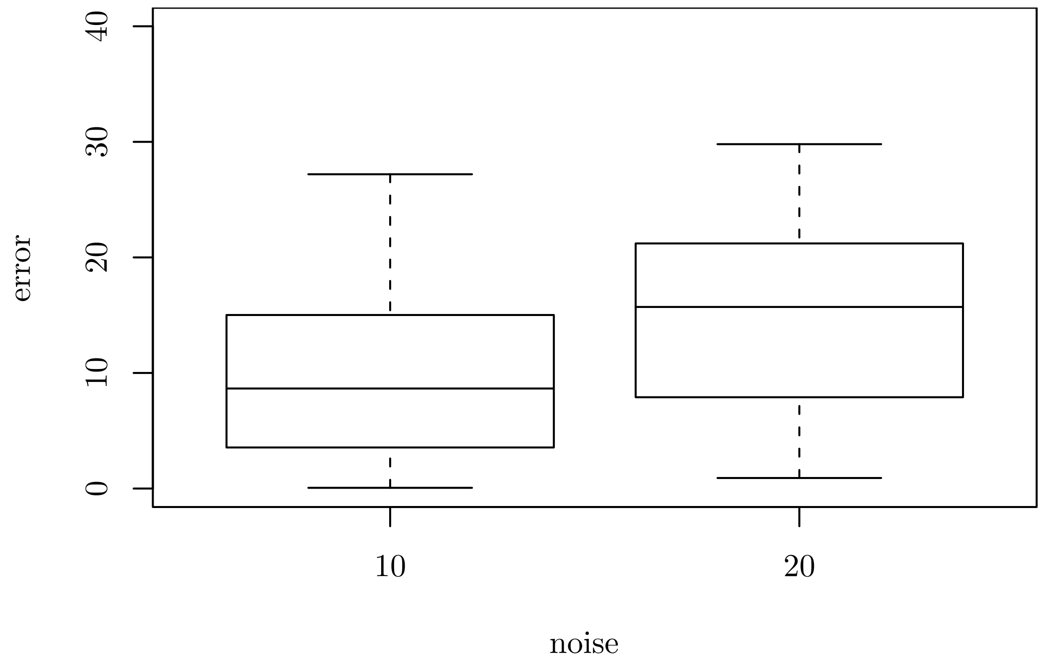
almost all types and levels of noise, the minimums and 1st quartiles are in 0% of error.

The results for tests made with matrices created from arbitrary bases can be seen in Figure 7. With arbitrary bases, only two levels of noise were used, namely 10 and 20 per cent. Once again, this was due to the computational complexity of solving the value of loss function. These results were not as good as were the results with the disjoint basis vectors. We can see this easily from Figure 7(a): the medians are close to the corresponding level of error and 1st quartiles are more or less far from 0. The effect of noise level was more clear than it was with the disjoint basis vectors.

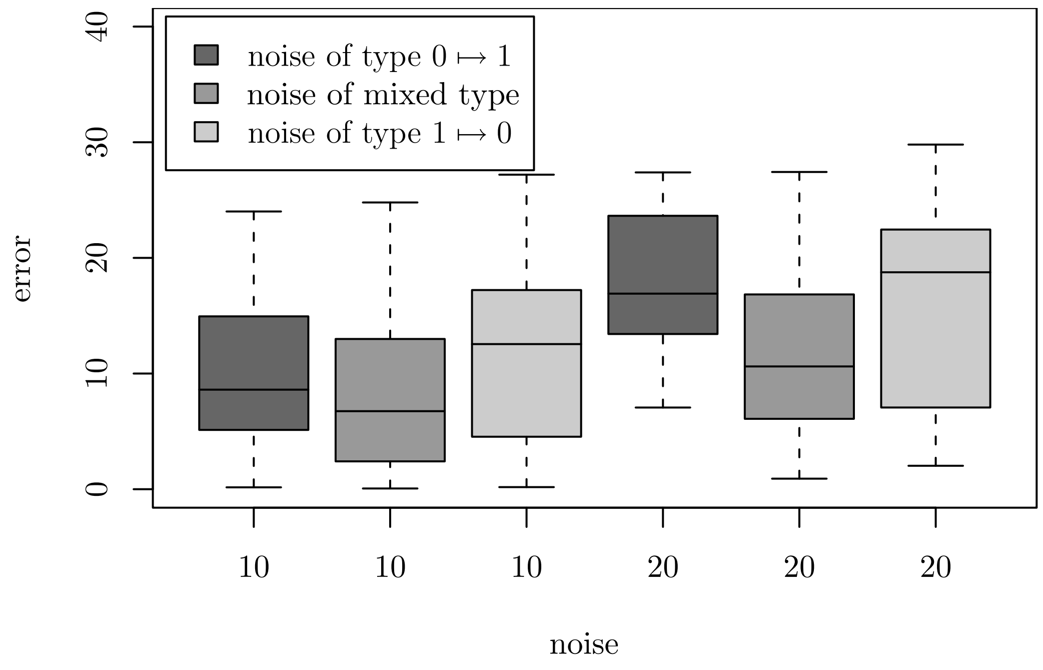
Once again, the *mixed* type of noise had the best results (Figure 7(b)). With arbitrary basis vectors, however, the results for the noise of type *left* can be considered worse than the results for the type *right*. This interpretation is based on the fact that the 1st quartile and the minimum for the noise of type *right* are at least as low as they are with the noise of type *left*. The interpretation is, however, questionable as the median, 3rd quartile and the maximum are lower with the noise of type *left*. As a whole, the results with the noise of type *left* are more concentrated around the corresponding error level.

The fact that the results with matrices created from arbitrary basis vectors are worse than they are with matrices created from disjoint basis vectors is not surprising. The arbitrary basis vectors are harder to locate and, as described in Section 4.2, there are even situations when the ASSOCIATION algorithm is unable to find correct basis vectors. The difference on effects of different types of noise is not so strong with arbitrary basis vectors. As the error level mostly follows the corresponding noise level, the results may be considered to be neutral—the effects of noise are not too strong, but they are still notable.

Real data. The Bird Atlas results were not good for the LOCALSEARCH algorithm, and they were not good either for the ASSOCIATION algorithm—for different reasons though. The behavior of the ASSOCIATION algorithm differs from the behavior of the LOCALSEARCH algorithm. As the algorithm does not enforce every column to belong to some basis vector, most of the rare bird species are not usually included at all. On the other hand, because the algorithm allows one column to belong in multiple basis vectors, the most common bird species tend to belong to almost every basis vector. The real problem is, however, the way of how the basis vectors are constructed. For almost every basis vector, there was one rare species and the accuracies were good due to this rare species: the other bird species nested in most



(a) All types of noise together.



(b) Different types of noise separated.

Figure 7: Boxplots showing the correlation between noise percentage and error percentage for the ASSOCIATION algorithm with arbitrary bases.

of the squares with this rare bird. The rest of the species were usually quite common, and thus that row in the association matrix had good cover-value. That one rare bird species does not decrease the value too much, but the interpretation of this kind of basis vector is hard—that one rare bird species does not live together with other, more common species in anywhere else but in those few squares where that rare bird species nests. The meaning of the basis vector is merely that some more common species do nest in same squares as that one rare species of bird. Another way of saying that is that when the rare bird species nests somewhere, there is τ probability that some of the other, more common, species is nesting within that 10km square. Neither of these interpretations is very enlightening, though.

As an example, when 11 basis vectors were searched with $\tau = 0.90$, both Redwing and Willow Warbler appeared in 10 of basis vectors together. Willow Warbler nests in most atlas squares of all species, and also Redwing is very common all over the Finland, albeit neither of them is nesting in archipelago [VLK98, p. 350 and 388]. Another very popular bird species in basis vectors was White Wagtail, which was in 8 of 11 vectors. White Wagtail is another very common bird, having second most atlas squares of all birds [VLK98, p. 320]. The only basis vector that did not have either Willow Warbler nor Redwing was made up from sea birds, with few exceptions. Most of those birds were nesting in southwest and west archipelago of Finland. The exceptions were White Wagtail, Wheatear and Hooded Crow—all common birds nesting all over Finland. The basis vector included Caspian Tern, which is quite a rare bird. It has only 275 nesting squares in Bird Atlas and all of them are in Finnish coastal [VLK98, p. 232]. Albeit the rest of the species in the basis vector do nest in same squares as Caspian Tern the opposite does not hold—Caspian Tern does not nest in eastern Finland, as White Wagtail do.

With $\tau = 0.90$, the error was somewhat high, 42.51%. When the threshold was lowered, at first to 0.80, the error decreased to 39.12%. When the threshold was still lowered, the error decreased even more. With $\tau = 0.70$ the error was 38.81%. However, with so low threshold, the size of basis vectors increased even more. Much of this increment happened, because more popular bird species appeared to almost every basis vector. This decreased the usability of basis vectors, and thus the decreased error did not help as much after all.

The results for the course completion database with all courses and threshold of 0.80 were surprisingly similar with the results given by the LocalSearch algorithm. Most notable, a basis vector with courses about law, psychology etc. was found. It included lots of same courses than the corresponding basis vectors found by LocalSearch, although some new courses were introduced. It was again by far the larges basis vector, including 35 courses in total, while the second largest basis vector had only 22 courses and the mean was 7 courses. It was also used only 33 times, while on average a basis vector was used 434 times. Many of the basis vectors had the phenomenon noted in Bird Atlas experiments: one rare course was the reason for the other courses to appear together. For example, the first basis vector was mostly made up from the basic, mandatory courses for Computer Science and Mathematics, including Orientation Studies, Reading Comprehension in English, Programming Project, Computer Organization[3] and many other frequent courses in database. But that basis vector also included a course named Course or Literature on Ethics and Social Philosophy, which only 10 students have passed.

As with the Bird Atlas dataset, most frequent courses appeared in multiple basis vectors. Reading Comprehension in English, the second frequent course in database, appeared in 9 of 15 basis vectors. However, it always appeared with other courses, i.e., it did not make up any basis vectors solely by itself. Two of the 15 basis vectors were singletons, and they were made up from courses Orientation Studies and Introduction to Statistics. The former course is the most frequent course in the database, while the latter is not in the top 20 of frequent courses.

Based on the numbers, the cover was not very good: the error was as high as 68.47%. Only a slightly better results were obtained, when the database was changed to have only courses that at least 10 students have passed. The error for it was still 62.01%. The results were very similar in other ways, too. Most of the basis vectors were same and they were used approximately as often. Only one basis vector was almost completely different, and the largest basis vector (with courses about law etc.) had lost some of its less frequent courses. Thus it seems that courses having less than 10 passed students did not have notable influence on the results.

No notable differences were obtained when only courses that more than 30 students have passed were considered. The error was kept high, in 57.69%. Many of the basis vectors were still same, although changes were bigger than between all courses and courses with more than 10 passed students. Also the basis vectors were used approximately as often as in previous experiments and once again the largest basis vector had lost some courses.

[3]These three courses also made up the top three courses on database with respect to the frequency.

The overall results for the course completion database were more intuitive than with the LocalSearch algorithm. The basis vectors were made up from frequent courses likely to be found from student's curriculum. The only exception was the appearance of the rare courses in basis vectors. Naturally, most of them disappeared from basis vectors when only more popular courses were considered. Probably one of the most surprising results was that the basis vectors did not change very much when some of the rarest courses were eliminated. Unfortunately, also the Association algorithm was somewhat unable to give good results for course completion database. We can see the numerical results for all real-world data tests in Table 4.

In summary, the results for the Association algorithm were twofold. On one hand, the algorithm failed to create bases with reasonable small error level. On the other hand, many of the results were more intuitive and natural than the results by LocalSearch. Probably the only major interpretation problem with basis vectors was the appearance of rare birds/courses, which was due to the method used to calculate the association matrix. However, the results with synthetic data do propose that the algorithm should be usable also with real-world data.

6 Conclusions and future work

In this thesis we have introduced the Discrete Basis Problem. We also introduced a related problem, the Basis Usage Problem, and two variations of the Discrete Basis Problem, namely the Disjoint Discrete Basis Problem and the Discrete Basis Partition Problem. We proved that the Discrete Basis Problem is $\mathcal{NP}$-complete problem and that it cannot be approximated within any finite approximation ratio. We also proved that the Boolean Geometric k-median Problem is $\mathcal{NP}$-complete and that it is, in a sense, an equivalent problem with the Discrete Basis Partition

Database	Basis vectors	τ	Error	Error (%)	Not covered 1s	Covered 0s
Birds	11	0.7	95685	38.81	60521	35164
Birds	11	0.8	96463	39.12	64276	32187
Birds	11	0.9	104799	42.51	71592	33207
Courses all	15	0.8	42908	68.47	38351	4557
Courses 10	15	0.8	32705	62.01	28660	4045
Courses 30	15	0.8	26883	57.69	23383	3500

Table 4: Numerical results for the Association algorithm with the real-world data.

Problem. It yields from these results, that the DBPP is $\mathcal{NP}$-complete and it can be approximated within constant approximation factor.

The Boolean Geometric k-median Problem is not the only problem that is related to the Discrete Basis Problem. From the vast number of related problems, we concentrated on bicliques and tiling techniques. We saw that a problem of tiling databases is very similar with the Discrete Basis Problem. We also saw that the algorithms for the discrete PCA, despite the apparent similarity of the problems, did not fit well to solve the Discrete Basis Problem.

Two algorithms were presented to solve the Discrete Basis Problem, the Association algorithm and the LocalSearch algorithm. The LocalSearch algorithm was based on solving the Metric k-median Problem, and thus it was only applicable with the variations. The Association algorithm does not have this kind of restrictions. The LocalSearch algorithm has constant approximation factor for the DBPP, but for the DDBP the case is similar with the Association algorithm for the general DBP: no approximation guarantees were not given.

To test the properties of these algorithms, some experiments with both synthetic and real-world data were made. The reported results were good for the synthetic data, but not as good for the real-world data. With the synthetic data, the LocalSearch algorithm was almost immune to the effects of noise, but with the Association algorithm and disjoint data, the results were not quite as good. For both algorithms, removing 1s in data was more problematic type of noise than adding extra 1s. This may be problematic, as in many real-world databases the lack of information, i.e., the absence of 1s, is more usual than the false information, i.e., the presence of false 1s. The Association algorithm was not able to remove the effects of noise in data generated from the arbitrary basis vectors. However, on average the level of error did not exceed the level of noise.

Neither of the algorithms worked well with the real data. There are several possible reasons for this: either the algorithms, the problem definition or the data is incorrect. The good results with the synthetic data suggest that the algorithms are working well and the definition of the DBP is very natural. Thus it seems that the data does not fit well with the idea of the basis vectors. It is an open question to investigate other datasets that might give better results with these algorithms and problem definition.

A natural goal for the future work is to find out new and better algorithms especially for the general DBP. For that, solving the computational complexity of the

Basis Usage Problem may be helpful. Reducing the approximation ratio of the LocalSearch algorithm should also be a subject of more in-depth studies.

The relationships between the Discrete Basis Problem and database tiling problems opens many new and interesting possibilities for future work with the Discrete Basis Problem. The relations between the Discrete Basis Problem and the discrete PCA methods should also be a subject for more in-depth studies.

The Disjoint Discrete Basis Problem and Discrete Basis Partition Problem are only some possible variations of the Discrete Basis Problem. Other variations are also a worth of studying. For instance, some loss functions may permit finite approximation ratios and non-metric loss functions may work better with many kinds of data.

References

ACG$^+$03 G. Ausiello, P. Crescenzi, G. Gambosi, et al. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer-Verlag Berlin Heidelberg, second edition, 2003.

AGK$^+$04 V. Arya, N. Garg, R. Kjandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

BJ04 W. Buntine and A. Jakulin. Applying discrete PCA in data analysis. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, ACM International Conference Proceedings Series, pages 59–66, 2004.

BNJ03 D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

BRB05 J. Besson, C. Robardet, and J.-F. Boulicaut. Mining formal concepts with a bounded number of exceptions from transactional data. In B. Goethals and A. Siebes, editors, *Knowledge Discovery in Inductive Databases: Third International Workshop, KDID 2004, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers*, volume 3377 of *Lecture Notes in Computer Science*, pages 33–45. Springer, 2005.

Bun02 W. Buntine. Variational extensions to EM and multinomial PCA. In *Proceedings of the ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 23–34. Springer, 2002.

CG99 M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science* [IEE99], pages 378–388.

DF99 R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in computer science. Springer-Verlag New York, 1999.

GGM04 F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In E. Suzuki and S. Arikawa, editors, *Discovery Science: 7th International Conference, Proceedings*, volume 3245 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2004.

GJ79 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, 1979.

GVL96 G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

Hoc98 D. S. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29(1):174–200, 1998.

Hof99 T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, August 1999.

IEE99 IEEE Computer Society. *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999.

JV99 K. Jain and V. V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science* [IEE99], pages 2–13.

KBH04 A. Kabán, E. Bingham, and T. Hirsimäki. Learning to read between the lines: The aspect bernoulli model. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 462–466, 2004.

KH79 O. Kariv and L. Hakimi. An algorithmic approach to network location problems. II: the p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.

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

Mie05 T. Mielikäinen. *Summarization Techniques for Pattern Collections in Data Mining.* PhD thesis, Report A-2005-1, Department of Computer Science, University of Helsinki, 2005.

MRS03 N. Mishra, D. Ron, and R. Swaminathan. On finding large conjunctive clusters. In B. Schölkopf and M. K. Warmuth, editors, *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2003.

MS84 N. Megiddo and K Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984.

Pap81 C. Papadimitriou. Worst-case and probabilistic analysis of a geometric location problem. *SIAM Journal on Computing*, 10(3):542–557, 1981.

Pap95 C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1995.

Pee03 R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

TB99 M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

VLK98 R. A. Väisänen, E. Lammi, and P. Koskimies. *Muuttuva Pesimälinnusto.* Otavan Kirjapaino, Keuruu, 1998.