algorithms

MDPI

Article

Co-Clustering under the Maximum Norm†

Laurent Bulteau [1], Vincent Froese [2,*], Sepp Hartung [2] and Rolf Niedermeier [2]

1 IGM-LabInfo, CNRS UMR 8049, Université Paris-Est Marne-la-Vallée, 77454 Marne-la-Vallée, France; laurent.bulteau@u-pem.fr

2 Institut für Softwaretechnik und Theoretische Informatik, 10587 TU Berlin, Germany; sepp.hartung@gmx.de (S.H.); rolf.niedermeier@tu-berlin.de (R.N.)

* Correspondence: vincent.froese@tu-berlin.de; Tel.: +49-30-314-73510; Fax: +49-30-314-23516

† This paper is an extended version of our paper published in Co-Clustering Under the Maximum Norm. In Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC' 14), LNCS 8889, Jeonju, Korea, 15–17 December 2014; pp. 298–309.

Academic Editor: Javier Del Ser Lorente

Received: 7 December 2015; Accepted: 16 February 2016; Published: 25 February 2016

Abstract: Co-clustering, that is partitioning a numerical matrix into "homogeneous" submatrices, has many applications ranging from bioinformatics to election analysis. Many interesting variants of co-clustering are NP-hard. We focus on the basic variant of co-clustering where the homogeneity of a submatrix is defined in terms of minimizing the maximum distance between two entries. In this context, we spot several NP-hard, as well as a number of relevant polynomial-time solvable special cases, thus charting the border of tractability for this challenging data clustering problem. For instance, we provide polynomial-time solvability when having to partition the rows and columns into two subsets each (meaning that one obtains four submatrices). When partitioning rows and columns into three subsets each, however, we encounter NP-hardness, even for input matrices containing only values from $\{0, 1, 2\}$.

Keywords: bi-clustering; matrix partitioning; NP-hardness; SAT solving; fixed-parameter tractability

1. Introduction

Co-clustering, also known as *bi-clustering* [1], performs a simultaneous clustering of the rows and columns of a data matrix. Roughly speaking, the problem is, given a numerical input matrix $\mathcal{A}$, to partition the rows and columns of $\mathcal{A}$ into subsets minimizing a given *cost* function (measuring "homogeneity"). For a given subset I of rows and a subset J of columns, the corresponding *cluster* consists of all entries a_{ij} with $i \in I$ and $j \in J$. The cost function usually defines homogeneity in terms of distances (measured in some norm) between the entries of each cluster. Note that the variant where clusters are allowed to "overlap", meaning that some rows and columns are contained in multiple clusters, has also been studied [1]. We focus on the non-overlapping variant, which can be stated as follows.

CO-CLUSTERING$_{\mathcal{L}}$
Input: A matrix $\mathcal{A} \in \mathbb{R}^{m \times n}$ and two positive integers $k, \ell \in \mathbb{N}$.
Task: Find a partition of $\mathcal{A}$'s rows into k subsets and a partition of $\mathcal{A}$'s columns into ℓ subsets, such that a given cost function (defined with respect to some norm $\mathcal{L}$) is minimized for the corresponding clustering.

Co-clustering is a fundamental paradigm for unsupervised data analysis. Its applications range from microarrays and bioinformatics over recommender systems to election analysis [1–4].

Due to its enormous practical significance, there is a vast amount of literature discussing various variants; however, due to the observed NP-hardness of "almost all interesting variants" [1], most of the literature deals with heuristic, typically empirically-validated algorithms. Indeed, there has been very active research on co-clustering in terms of heuristic algorithms, while there is little substantial theoretical work for this important clustering problem. Motivated by an effort towards a deeper theoretical analysis, as started by Anagnostopoulos *et al.* [2], we further refine and strengthen the theoretical investigations on the computational complexity of a natural special case of Co-Clustering$_{\mathcal{L}}$, namely we study the case of $\mathcal{L}$ being the maximum norm L_∞, where the problem comes down to minimizing the maximum distance between entries of a cluster. This cost function might be a reasonable choice in practice due to its outlier sensitivity. In network security, for example, there often exists a vast amount of "normal" data points, whereas there are only very few "malicious" data points, which are outliers with respect to certain attributes. The maximum norm does not allow one to put entries with large differences into the same cluster, which is crucial for detecting possible attacks. The maximum norm can also be applied in a discretized setting, where input values are grouped (for example, replaced by integer values) according to their level of deviation from the mean of the respective attribute. It is then not allowed to put values of different ranges of the standard deviation into the same cluster. Last, but not least, we study an even more restricted clustering version, where the partitions of the rows and columns have to contain consecutive subsets. This version subsumes the problem of feature discretization, which is used as a preprocessing technique in data mining applications [5–7]. See Section 3.3 for this version.

Anagnostopoulos *et al.* [2] provided a thorough analysis of the polynomial-time approximability of Co-Clustering$_{\mathcal{L}}$ (with respect to L_p-norms), presenting several constant-factor approximation algorithms. While their algorithms are almost straightforward, relying on one-dimensionally clustering first the rows and then the columns, their main contribution lies in the sophisticated mathematical analysis of the corresponding approximation factors. Note that Jegelka *et al.* [8] further generalized this approach to higher dimensions, then called *tensor clustering*. In this work, we study (efficient) *exact* instead of approximate solvability. To this end, by focusing on Co-Clustering$_{\infty}$, we investigate a scenario that is combinatorially easier to grasp. In particular, our exact and combinatorial polynomial-time algorithms exploit structural properties of the input matrix and do not solely depend on one-dimensional approaches.

1.1. Related Work

Our main point of reference is the work of Anagnostopoulos *et al.* [2]. Their focus is on polynomial-time approximation algorithms, but they also provide computational hardness results. In particular, they point to challenging open questions concerning the cases $k = \ell = 2$, $k = 1$, or binary input matrices. Within our more restricted setting using the maximum norm, we can resolve parts of these questions. The survey of Madeira and Oliveira [1] (according to Google Scholar, accessed December 2015, cited more than 1500 times) provides an excellent overview on the many variations of Co-Clustering$_{\mathcal{L}}$, there called bi-clustering, and discusses many applications in bioinformatics and beyond. In particular, they also discuss Hartigan's [9] special case where the goal is to partition into uniform clusters (that is, each cluster has only one entry value). Our studies indeed generalize this very puristic scenario by not demanding completely uniform clusters (which would correspond to clusters with maximum entry difference zero), but allowing some variation between maximum and minimum cluster entries. Califano *et al.* [10] aim at clusterings where in each submatrix, the distance between entries within each row and within each column is upper-bounded. Recent work by Wulff *et al.* [11] considers a so-called "monochromatic" bi-clustering where the cost for each submatrix is defined as the number of minority entries. For binary data, this clustering task coincides with the L_1-norm version of co-clustering, as defined by Anagnostopoulos *et al.* [2]. Wulff *et al.* [11] show the NP-hardness of monochromatic bi-clustering for binary data with an additional third value denoting missing entries (which are not considered in their cost function) and give a randomized

polynomial-time approximation scheme (PTAS). Except for the work of Anagnostopoulos *et al.* [2] and Wulff *et al.* [11], all other investigations mentioned above are empirical in nature.

1.2. Our Contributions

In terms of defining "cluster homogeneity", we focus on minimizing the maximum distance between two entries within a cluster (maximum norm). Table 1 surveys most of our results. Our main conceptual contribution is to provide a seemingly first study on the exact complexity of a natural special case of Co-Clustering$_{\mathcal{L}}$, thus potentially stimulating a promising field of research.

Table 1. Overview of results for (k, ℓ)-Co-Clustering$_{\infty}$ with respect to various parameter constellations (m: number of rows; $|\Sigma|$: alphabet size; k/ℓ: size of row/column partition; c: cost). A $\circledast$ indicates that the corresponding value is considered as a parameter, where FPT (fixed-parameter tractable (FPT)) means that there is an algorithm solving the problem where the super-polynomial part in the running time is a function depending solely on the parameter. Multiple $\circledast$'s indicate a combined parameterization. Other non-constant values may be unbounded.

m	$\lvert\Sigma\rvert$	k	ℓ	c	Complexity
-	-	-	-	0	P [Observation 1]
-	2	-	-	-	P [Observation 1]
-	-	1	-	-	P [Theorem 4]
-	-	2	2	-	P [Theorem 5]
-	3	2	-	-	P [Theorem 6]
-	-	2	$\circledast$	1	FPT [Corollary 2]
-	$\circledast$	2	-	1	FPT [Corollary 2]
$\circledast$	-	$\circledast$	$\circledast$	$\circledast$	FPT [Lemma 2]
-	3	3	3	1	NP-hard [Theorem 1]
2	-	2	-	2	NP-hard [Theorem 2]

Our main technical contributions are as follows. Concerning the computational intractability results with respect to even strongly-restricted cases, we put much effort into finding the "right" problems to reduce from in order to make the reductions as natural and expressive as possible, thus making non-obvious connections to fields, such as geometric set covering. Moreover, seemingly for the first time in the context of co-clustering, we demonstrate that the inherent NP-hardness does not stem from the permutation combinatorics behind: the problem remains NP-hard when all clusters must consist of consecutive rows or columns. This is a strong constraint (the search space size is tremendously reduced, basically from $k^m \cdot \ell^n$ to $\binom{m}{k} \cdot \binom{n}{\ell}$), which directly gives a polynomial-time algorithm for k and ℓ being constants. Note that in the general case, we have NP-hardness for constant k and ℓ. Concerning the algorithmic results, we develop a novel reduction to SAT solving (instead of the standard reductions to integer linear programming). Notably, however, as opposed to previous work on polynomial-time approximation algorithms [2,8], our methods seem to be tailored for the two-dimensional case (co-clustering), and the higher dimensional case (tensor clustering) appears to be out of reach.

2. Formal Definitions and Preliminaries

We use standard terminology for matrices. A matrix $\mathcal{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$ consists of m rows and n columns where a_{ij} denotes the entry in row i and column j. We define $[n] := \{1, 2, \ldots, n\}$ and $[i, j] := \{i, i+1, \ldots, j\}$ for $n, i, j \in \mathbb{N}$. For simplicity, we neglect the running times of arithmetical operations throughout this paper. Since we can assume that the input values of $\mathcal{A}$ are upper-bounded polynomially in the size mn of $\mathcal{A}$ (Observation 2), the blow-up in the running times is at most polynomial.

2.1. Problem Definition

We follow the terminology of Anagnostopoulos *et al.* [2]. For a matrix $\mathcal{A} \in \mathbb{R}^{m \times n}$, a (k, ℓ)-*co-clustering* is a pair $(\mathcal{I}, \mathcal{J})$ consisting of a *k-partition* $\mathcal{I} = \{I_1, \ldots, I_k\}$ of the row indices $[m]$ of $\mathcal{A}$ (that is, $I_i \subseteq [m]$ for all $1 \leq i \leq k$, $I_i \cap I_j = \emptyset$ for all $1 \leq i < j \leq k$ and $\bigcup_{i=1}^{k} I_i = [m]$) and an ℓ-partition $\mathcal{J} = \{J_1, \ldots, J_\ell\}$ of the column indices $[n]$ of $\mathcal{A}$. We call the elements of $\mathcal{I}$ (resp., $\mathcal{J}$) row blocks (column blocks, resp.). Additionally, we require $\mathcal{I}$ and $\mathcal{J}$ to not contain empty sets. For $(r, s) \in [k] \times [\ell]$, the set $\mathcal{A}_{rs} := \{a_{ij} \in \mathcal{A} \mid (i, j) \in I_r \times J_s\}$ is called a *cluster*.

The cost of a co-clustering (under maximum norm, which is the only norm we consider here) is defined as the maximum difference between any two entries in any cluster, formally $\text{cost}_\infty(\mathcal{I}, \mathcal{J}) := \max_{(r,s) \in [k] \times [\ell]} (\max \mathcal{A}_{rs} - \min \mathcal{A}_{rs})$. Herein, $\max \mathcal{A}_{rs}$ ($\min \mathcal{A}_{rs}$) denotes the maximum (minimum, resp.) entry in $\mathcal{A}_{rs}$.

The decision variant of Co-Clustering$_\mathcal{L}$ with maximum norm is as follows.

Co-Clustering$_\infty$
Input: A matrix $\mathcal{A} \in \mathbb{R}^{m \times n}$, integers $k, \ell \in \mathbb{N}$ and a cost $c \geq 0$.
Question: Is there a (k, ℓ)-co-clustering $(\mathcal{I}, \mathcal{J})$ of $\mathcal{A}$ with $\text{cost}_\infty(\mathcal{I}, \mathcal{J}) \leq c$?

See Figure 1 for an introductory example. We define $\Sigma := \{a_{ij} \in \mathcal{A} \mid (i, j) \in [m] \times [n]\}$ to be the *alphabet* of the input matrix $\mathcal{A}$ (consisting of the numerical values that occur in $\mathcal{A}$). Note that $|\Sigma| \leq mn$. We use the abbreviation (k, ℓ)-Co-Clustering$_\infty$ to refer to Co-Clustering$_\infty$ with constants $k, \ell \in \mathbb{N}$, and by $(k, *)$-Co-Clustering$_\infty$, we refer to the case where only k is constant and ℓ is part of the input. Clearly, Co-Clustering$_\infty$ is symmetric with respect to k and ℓ in the sense that any (k, ℓ)-co-clustering of a matrix $\mathcal{A}$ is equivalent to an (ℓ, k)-co-clustering of the transposed matrix $\mathcal{A}^T$. Hence, we always assume that $k \leq \ell$.

$$\mathcal{A} = \begin{bmatrix} 1 & 3 & 4 & 1 \\ 2 & 2 & 1 & 3 \\ 0 & 4 & 3 & 0 \end{bmatrix}$$

$$\begin{array}{c|cc|c} & J_1 & & J_2 \\ I_1 & 1 & 4 & 1 & 3 \\ \hline I_2 & 2 & 1 & 3 & 2 \\ & 0 & 3 & 0 & 4 \end{array}$$

$I_1 = \{1\}, I_2 = \{2, 3\}$
$J_1 = \{1, 3, 4\}, J_2 = \{2\}$

$$\begin{array}{c|cc|c} & J_1 & & J_2 \\ I_1 & 2 & 3 & 2 & 1 \\ \hline I_2 & 1 & 1 & 3 & 4 \\ & 0 & 0 & 4 & 3 \end{array}$$

$I_1 = \{2\}, I_2 = \{1, 3\}$
$J_1 = \{1, 4\}, J_2 = \{2, 3\}$

Figure 1. The example shows two $(2, 2)$-co-clusterings (middle and right) of the same matrix $\mathcal{A}$ (left-hand side). It demonstrates that by sorting rows and columns according to the co-clustering, the clusters can be illustrated as submatrices of this (permuted) input matrix. The cost of the $(2, 2)$-co-clustering in the middle is three (because of the two left clusters), and that of the $(2, 2)$-co-clustering on the right-hand side is one.

We next collect some simple observations. First, determining whether there is a cost-zero (perfect) co-clustering is easy. Moreover, since, for a binary alphabet, the only interesting case is a perfect co-clustering, we get the following.

Observation 1. Co-Clustering$_\infty$ *is solvable in* $O(mn)$ *time for cost zero and also for any size-two alphabet.*

Proof. Let $(\mathcal{A}, k, \ell, 0)$ be a Co-Clustering$_\infty$ input instance. For a (k, ℓ)-co-clustering with cost zero, it holds that all entries of a cluster are equal. This is only possible if there are at most k different rows and at most ℓ different columns in $\mathcal{A}$, since otherwise, there will be a cluster containing two different entries. Thus, the case $c = 0$ can be solved by lexicographically sorting the rows and columns of $\mathcal{A}$ in $O(mn)$ time (e.g., using radix sort). □

We further observe that the input matrix can, without loss of generality, be assumed to contain only integer values (by some rescaling arguments preserving the distance relations between elements).

Observation 2. *For any* Co-Clustering$_\infty$*-instance with arbitrary alphabet* $\Sigma \subset \mathbb{R}$*, one can find in* $O(|\Sigma|^2)$ *time an equivalent instance with alphabet* $\Sigma' \subset \mathbb{Z}$ *and cost value* $c' \in \mathbb{N}$.

Proof. We show that for any instance with arbitrary alphabet $\Sigma \subset \mathbb{R}$ and cost $c \geq 0$, there exists an equivalent instance with $\Sigma' \subset \mathbb{Z}$ and $c' \in \mathbb{N}$. Let σ_i be the i-th element of Σ with respect to any fixed ordering. The idea is that the cost value c determines which elements of Σ are allowed to appear together in a cluster of a cost-c co-clustering. Namely, in any cost-c co-clustering, two elements $\sigma_i \neq \sigma_j$ can occur in the same cluster if and only if $|\sigma_i - \sigma_j| \leq c$. These constraints can be encoded in an undirected graph $G_c := (\Sigma, E)$ with $E := \{\{\sigma_i, \sigma_j\} \mid \sigma_i \neq \sigma_j \in \Sigma, |\sigma_i - \sigma_j| \leq c\}$, where each vertex corresponds to an element of Σ, and there is an edge between two vertices if and only if the corresponding elements can occur in the same cluster of a cost-c co-clustering.

Now, observe that G_c is a *unit interval graph*, since each vertex σ_i can be represented by the length-c interval $[\sigma_i, \sigma_i + c]$, such that it holds $\{\sigma_i, \sigma_j\} \in E \Leftrightarrow [\sigma_i, \sigma_i + c] \cap [\sigma_j, \sigma_j + c] \neq \emptyset$ (we assume all intervals to contain real values). By properly shifting and rescaling the intervals, one can find an embedding of G_c, where the vertices σ_i are represented by length-c' intervals $[\sigma_i', \sigma_i' + c']$ of equal integer length $c' \in \mathbb{N}$ with integer starting points $\sigma_i' \in \mathbb{Z}$, such that $0 \leq \sigma_i' \leq |\Sigma|^2$, $c' \leq |\Sigma|$, and $|\sigma_i' - \sigma_j'| \leq c' \Leftrightarrow |\sigma_i - \sigma_j| \leq c$. Hence, replacing the elements σ_i by σ_i' in the input matrix yields a matrix that has a cost-c' co-clustering if and only if the original input matrix has a cost-c co-clustering. Thus, for any instance with alphabet Σ and cost c, there is an equivalent instance with alphabet $\Sigma' \subseteq \{0, \ldots, |\Sigma|^2\}$ and cost $c' \in \{0, \ldots, |\Sigma|\}$. Consequently, we can upper-bound the values in Σ' by $|\Sigma|^2 \leq (mn)^2$. □

Due to Observation 2, we henceforth assume for the rest of the paper that the input matrix contains integers.

2.2. Parameterized Algorithmics

We briefly introduce the relevant notions from parameterized algorithmics (refer to the monographs [12–14] for a detailed introduction). A *parameterized problem*, where each instance consists of the "classical" problem instance I and an integer ρ called *parameter*, is *fixed-parameter tractable* (FPT) if there is a computable function f and an algorithm solving any instance in $f(\rho) \cdot |I|^{O(1)}$ time. The corresponding algorithm is called an FPT-algorithm.

3. Intractability Results

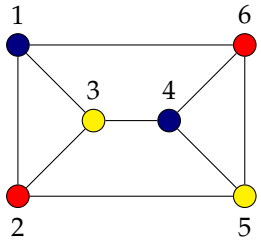
In the previous section, we observed that Co-Clustering$_\infty$ is easy to solve for binary input matrices (Observation 1). In contrast to this, we show in this section that its computational complexity significantly changes as soon as the input matrix contains at least three different entries. In fact, even for very restricted special cases, we can show NP-hardness. These special cases comprise co-clusterings with a constant number of clusters (Section 3.1) or input matrices with only two rows (Section 3.2). We also show the NP-hardness of finding co-clusterings where the row and column partitions are only allowed to contain consecutive blocks (Section 3.3).

3.1. Constant Number of Clusters

We start by showing that for input matrices containing three different entries, Co-Clustering$_\infty$ is NP-hard even if the co-clustering consists only of nine clusters.

Theorem 1. $(3,3)$-Co-Clustering$_\infty$ *is NP-hard for* $\Sigma = \{0, 1, 2\}$.

Proof. We prove NP-hardness by reducing from the NP-complete 3-Coloring [15], where the task is to partition the vertex set of a undirected graph into three independent sets. Let $G = (V, E)$ be a 3-Coloring instance with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We construct a $(3,3)$-Co-Clustering$_\infty$ instance $(\mathcal{A} \in \{0,1,2\}^{m \times n}, k := 3, \ell := 3, c := 1)$ as follows. The columns of $\mathcal{A}$ correspond to the vertices V, and the rows correspond to the edges E. For an edge $e_i = \{v_j, v_{j'}\} \in E$ with $j < j'$, we set $a_{ij} := 0$ and $a_{ij'} := 2$. All other matrix entries are set to 1. Hence, each row corresponding to an edge $\{v_j, v_{j'}\}$ consists of 1-entries except for the columns j and j', which contain 0 and 2 (see Figure 2). Thus, every co-clustering of $\mathcal{A}$ with a cost at most $c = 1$ puts column j and column j' into different column blocks. We next prove that there is a $(3,3)$-co-clustering of $\mathcal{A}$ with a cost at most $c = 1$ if and only if G admits a 3-coloring.



	2	6	1	4	3	5
{2,3}	0	1	1	1	2	1
{2,5}	0	1	1	1	1	2
{1,2}	2	1	0	1	1	1
{1,3}	1	1	0	1	2	1
{1,6}	1	2	0	1	1	1
{4,5}	1	1	1	0	1	2
{4,6}	1	2	1	0	1	1
{3,4}	1	1	1	2	0	1
{5,6}	1	2	1	1	1	0

Figure 2. An illustration of the reduction from 3-Coloring. **Left**: An undirected graph with a proper 3-coloring of the vertices, such that no two neighboring vertices have the same color. **Right**: The corresponding matrix where the columns are labeled by vertices and the rows by edges with a $(3,3)$-co-clustering of cost one. The coloring of the vertices determines the column partition into three columns blocks, whereas the row blocks are generated by the following simple scheme: edges where the vertex with a smaller index is red/blue (dark)/yellow (light) are in the first/second/third row block (e.g., the red-yellow edge $\{2,5\}$ is in the first block; the blue-red edge $\{1,6\}$ is in the second block; and the yellow-blue edge $\{3,4\}$ is in the third block).

First, assume that V_1, V_2, V_3 is a partition of the vertex set V into three independent sets. We define a $(3,3)$-co-clustering $(\mathcal{I}, \mathcal{J})$ of $\mathcal{A}$ as follows. The column partition $\mathcal{J} := \{J_1, J_2, J_3\}$ one-to-one corresponds to the three sets V_1, V_2, V_3, that is $J_s := \{i \mid v_i \in V_s\}$ for all $s \in \{1,2,3\}$. By the construction above, each row has exactly two non-1-entries being 0 and 2. We define the type of a row to be a permutation of $0,1,2$, denoting which of the column blocks J_1, J_2, J_3 contain the 0-entry and the 2-entry. For example, a row is of type $(2,0,1)$ if it has a 2 in a column of J_1 and a 0 in a column of J_2. The row partition $\mathcal{I} := \{I_1, I_2, I_3\}$ is defined as follows: All rows of type $(0,2,1)$ or $(0,1,2)$ are put into I_1. Rows of type $(2,0,1)$ or $(1,0,2)$ are contained in I_2, and the remaining rows of type $(2,1,0)$ or $(1,2,0)$ are contained in I_3. Clearly, for $(\mathcal{I}, \mathcal{J})$, it holds that the non-1-entries in any cluster are either all 0 or all 2, implying that $\text{cost}_\infty(\mathcal{I}, \mathcal{J}) \leq 1$.

Next, assume that $(\mathcal{I}, \{J_1, J_2, J_3\})$ is a $(3,3)$-co-clustering of $\mathcal{A}$ with a cost at most 1. The vertex sets V_1, V_2, V_3, where V_s contains the vertices corresponding to the columns in J_s, form three independent sets: if an edge connects two vertices in V_s, then the corresponding row would have the 0-entry and the 2-entry in the same column block J_s, yielding a cost of 2, which is a contradiction. □

Theorem 1 can even be strengthened further.

Corollary 1. Co-Clustering$_\infty$ *with* $\Sigma = \{0,1,2\}$ *is NP-hard for any* $k \geq 3$, *even when* $\ell \geq 3$ *is fixed, and the column blocks are forced to have equal sizes* $|J_1| = \ldots = |J_\ell|$.

Proof. Note that the reduction in Theorem 1 clearly holds for any $k \geq 3$. Furthermore, ℓ-Coloring with balanced partition sizes is still NP-hard for $\ell \geq 3$ [15]. □

3.2. Constant Number of Rows

The reduction in the proof of Theorem 1 outputs matrices with an unbounded number of rows and columns containing only three different values. We now show that also the "dual restriction" is NP-hard, that is the input matrix only has a constant number of rows (two), but contains an unbounded number of different values. Interestingly, this special case is closely related to a two-dimensional variant of geometric set covering.

Theorem 2. Co-Clustering$_\infty$ *is NP-hard for* $k = m = 2$ *and unbounded alphabet size* $|\Sigma|$.

Proof. We give a polynomial-time reduction from the NP-complete Box Cover problem [16]. Given a set $P \subseteq \mathbb{Z}^2$ of n points in the plane and $\ell \in \mathbb{N}$, Box Cover is the problem to decide whether there are ℓ squares $S_1, \ldots, S_\ell$, each with side length two, covering P, that is $P \subseteq \bigcup_{1 \leq s \leq \ell} S_s$.

Let $I = (P, \ell)$ be a Box Cover instance. We define the instance $I' := (\mathcal{A}, k, \ell', c)$ as follows: The matrix $\mathcal{A} \in \mathbb{Z}^{2 \times n}$ has the points $p_1, \ldots, p_n$ in P as columns. Further, we set $k := 2$, $\ell' := \ell$, $c := 2$. See Figure 3 for a small example.

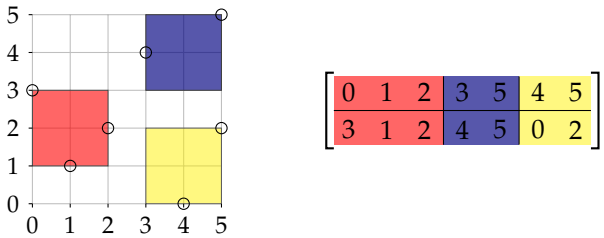


Figure 3. Example of a Box Cover instance with seven points (**left**) and the corresponding Co-Clustering$_\infty$ matrix containing the coordinates of the points as columns (**right**). Indicated is a $(2,3)$-co-clustering of cost two where the column blocks are colored according to the three squares (of side length two) that cover all points.

The correctness can be seen as follows: Assume that I is a yes-instance, that is there are ℓ squares $S_1, \ldots, S_\ell$ covering all points in P. We define $J_1 := \{j \mid p_j \in P \cap S_1\}$ and $J_s := \{j \mid p_j \in P \cap S_s \setminus (\bigcup_{1 \leq l < s} S_l)\}$ for all $2 \leq s \leq \ell$. Note that $(\mathcal{I} := \{\{1\}, \{2\}\}, \mathcal{J} := \{J_1, \ldots, J_\ell\})$ is a $(2, \ell)$-co-clustering of $\mathcal{A}$. Moreover, since all points with indices in J_s lie inside a square with side length two, it holds that each pair of entries in $\mathcal{A}_{1s}$, as well as in $\mathcal{A}_{2s}$ has a distance at most two, implying $\text{cost}_\infty(\mathcal{I}, \mathcal{J}) \leq 2$.

Conversely, if I' is a yes-instance, then let $(\{\{1\}, \{2\}\}, \mathcal{J})$ be the $(2, \ell)$-co-clustering of a cost at most two. For any $J_s \in \mathcal{J}$, it holds that all points corresponding to the columns in J_s have a pairwise distance at most two in both coordinates. Thus, there exists a square of side length two covering all of them. □

3.3. Clustering into Consecutive Clusters

One is tempted to assume that the hardness of the previous special cases of Co-Clustering$_\infty$ is rooted in the fact that we are allowed to choose arbitrary subsets for the corresponding row and column partitions since the problem remains hard even for a constant number of clusters and also with equal cluster sizes. Hence, in this section, we consider a restricted version of Co-Clustering$_\infty$, where the row and the column partition has to consist of consecutive blocks. Formally, for row indices

$R = \{r_1, \ldots, r_{k-1}\}$ with $1 < r_1 < \ldots < r_{k-1} \leq m$ and column indices $C = \{c_1, \ldots, c_{\ell-1}\}$ with $1 < c_1 < \ldots < c_{\ell-1} \leq n$, the corresponding *consecutive* (k, ℓ)-co-clustering $(\mathcal{I}_R, \mathcal{J}_C)$ is defined as:

$$\mathcal{I}_R := \{\{1, \ldots, r_1 - 1\}, \{r_1, \ldots, r_2 - 1\}, \ldots, \{r_{k-1}, \ldots, m\}\}$$
$$\mathcal{J}_C := \{\{1, \ldots, c_1 - 1\}, \{c_1, \ldots, c_2 - 1\}, \ldots, \{c_{\ell-1}, \ldots, n\}\}$$

The CONSECUTIVE CO-CLUSTERING$_\infty$ problem now is to find a consecutive (k, ℓ)-co-clustering of a given input matrix with a given cost. Again, also this restriction is not sufficient to overcome the inherent intractability of co-clustering, that is we prove it to be NP-hard. Similarly to Section 3.2, we encounter a close relation of consecutive co-clustering to a geometric problem, namely to find an optimal discretization of the plane; a preprocessing problem with applications in data mining [5–7]. The NP-hard OPTIMAL DISCRETIZATION problem [6] is the following: Given a set $S = B \cup W$ of points in the plane, where each point is either colored black (B) or white (W), and integers $k, \ell \in \mathbb{N}$, decide whether there is a consistent set of k horizontal and ℓ vertical (axis-parallel) lines. That is, the vertical and horizontal lines partition the plane into rectangular regions, such that no region contains two points of different colors (see Figure 4 for an example). Here, a vertical (horizontal) line is a simple number denoting its x- (y-) coordinate.

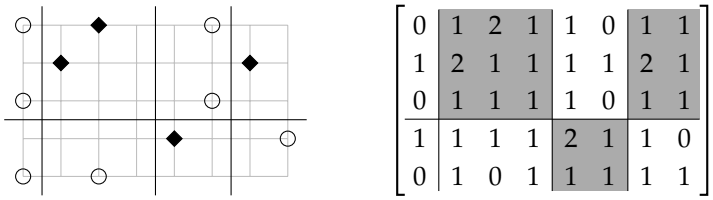


Figure 4. Example instance of OPTIMAL DISCRETIZATION (**left**) and the corresponding instance of CONSECUTIVE CO-CLUSTERING$_\infty$ (**right**). The point set consists of white (circles) and black (diamonds) points. A solution for the corresponding CONSECUTIVE CO-CLUSTERING$_\infty$ instance (shaded clusters) naturally translates into a consistent set of lines.

Theorem 3. CONSECUTIVE CO-CLUSTERING$_\infty$ *is NP-hard for* $\Sigma = \{0, 1, 2\}$.

Proof. We give a polynomial-time reduction from OPTIMAL DISCRETIZATION. Let (S, k, ℓ) be an OPTIMAL DISCRETIZATION instance; let $X := \{x_1^*, \ldots, x_n^*\}$ be the set of different x-coordinates; and let $Y := \{y_1^*, \ldots, y_m^*\}$ be the set of different y-coordinates of the points in S. Note that n and m can be smaller than $|S|$, since two points can have the same x- or y-coordinate. Furthermore, assume that $x_1^* < \ldots < x_n^*$ and $y_1^* < \ldots < y_m^*$. We now define the CONSECUTIVE CO-CLUSTERING$_\infty$ instance $(\mathcal{A}, k + 1, \ell + 1, c)$ as follows: The matrix $\mathcal{A} \in \{0, 1, 2\}^{m \times n}$ has columns labeled with $x_1^*, \ldots, x_n^*$ and rows labeled with $y_1^*, \ldots, y_m^*$. For $(x, y) \in X \times Y$, the entry a_{xy} is defined as zero if $(x, y) \in W$, two if $(x, y) \in B$ and otherwise one. The cost is set to $c := 1$. Clearly, this instance can be constructed in polynomial time.

To verify the correctness of the reduction, assume first that I is a yes-instance, that is there is a set $H = \{x_1, \ldots, x_k\}$ of k horizontal lines and a set $V = \{y_1, \ldots, y_\ell\}$ of ℓ vertical lines partitioning the plane consistently. We define row indices $R := \{r_1, \ldots, r_k\}$, $r_i := \max\{x^* \in X \mid x^* \leq x_i\}$ and column indices $C := \{c_1, \ldots, c_\ell\}$, $c_j := \max\{y^* \in Y \mid y^* \leq y_j\}$. For the corresponding $(k + 1, \ell + 1)$-co-clustering $(\mathcal{I}_R, \mathcal{J}_C)$, it holds that no cluster contains both values zero and two, since otherwise the corresponding partition of the plane defined by H and V contains a region with two points of different colors, which contradicts consistency. Thus, we have $\text{cost}_\infty(\mathcal{I}_R, \mathcal{J}_C) \leq 1$, implying that I' is a yes-instance.

Conversely, if I' is a yes-instance, then there exists a $(k+1, \ell+1)$-co-clustering $(\mathcal{I}_R, \mathcal{J}_C)$ with cost at most one, that is no cluster contains both values zero and two. Clearly, then, the k horizontal lines

$x_i := \min I_{i+1}$, $i = 1, \ldots, k$, and the ℓ vertical lines $y_j := \min J_{j+1}$, $j = 1, \ldots, \ell$ are consistent. Hence, I is a yes-instance. □

Note that even though Consecutive Co-Clustering$_\infty$ is NP-hard, there still is some difference in its computational complexity compared to the general version. In contrast to Co-Clustering$_\infty$, the consecutive version is polynomial-time solvable for constants k and ℓ by simply trying out all $O(m^k n^\ell)$ consecutive partitions of the rows and columns.

4. Tractability Results

In Section 3, we showed that Co-Clustering$_\infty$ is NP-hard for $k = \ell = 3$ and also for $k = 2$ in the case of unbounded ℓ and $|\Sigma|$. In contrast to these hardness results, we now investigate which parameter combinations yield tractable cases. It turns out (Section 4.2) that the problem is polynomial-time solvable for $k = \ell = 2$ and for $k = 1$. We can even solve the case $k = 2$ and $\ell \geq 3$ for $|\Sigma| = 3$ in polynomial time by showing that this case is in fact equivalent to the case $k = \ell = 2$. Note that these tractability results nicely complement the hardness results from Section 3. We further show fixed-parameter tractability for the parameters size of the alphabet $|\Sigma|$ and the number of column blocks ℓ (Section 4.3).

We start (Section 4.1) by describing a reduction of Co-Clustering$_\infty$ to CNF-SAT (the satisfiability problem for Boolean formulas in conjunctive normal form). Later on, it will be used in some special cases (see Theorems 5 and 7), because there, the corresponding formula, or an equivalent formula, only consists of clauses containing two literals, thus being a polynomial-time solvable 2-SAT instance.

4.1. Reduction to CNF-SAT Solving

In this section, we describe two approaches to solve Co-Clustering$_\infty$ via CNF-SAT. The first approach is based on a straightforward reduction of a Co-Clustering$_\infty$ instance to one CNF-SAT instance with clauses of size at least four. Note that this does not yield any theoretical improvements in general. Hence, we develop a second approach, which requires solving $O(|\Sigma|^{k\ell})$ many CNF-SAT instances with clauses of size at most $\max\{k, \ell, 2\}$. The theoretical advantage of this approach is that if k and ℓ are constants, then there are only polynomially many CNF-SAT instances to solve. Moreover, the formulas contain smaller clauses (for $k \leq \ell \leq 2$, we even obtain polynomial-time solvable 2-SAT instances). While the second approach leads to (theoretically) tractable special cases, it is not clear that it also performs better in practice. This is why we conducted some experiments for empirical comparison of the two approaches (in fact, it turns out that the straightforward approach allows one to solve larger instances). In the following, we describe the reductions in detail and briefly discuss the experimental results.

We start with the straightforward polynomial-time reduction from Co-Clustering$_\infty$ to CNF-SAT. We simply introduce a variable $x_{i,r}$ ($y_{j,s}$) for each pair of row index $i \in [m]$ and row block index $r \in [k]$ (respectively, column index $j \in [n]$ and column block index $s \in [\ell]$) denoting whether the respective row (column) may be put into the respective row (column) block. For each row i, we enforce that it is put into at least one row block with the clause $(x_{i,1} \vee \ldots \vee x_{i,k})$ (analogously for the columns). We encode the cost constraints by introducing $k\ell$ clauses $(\neg x_{i,r} \vee \neg x_{i',r} \vee \neg y_{j,s} \vee \neg y_{j',s})$, $(r, s) \in [k] \times [\ell]$ for each pair of entries $a_{ij}, a_{i'j'} \in \mathcal{A}$ with $|a_{ij} - a_{i'j'}| > c$. These clauses simply ensure that a_{ij} and $a_{i'j'}$ are not put into the same cluster. Note that this reduction yields a CNF-SAT instance with $km + \ell n$ variables and $O((mn)^2 k\ell)$ clauses of size up to $\max\{k, \ell, 4\}$.

Based on experiments (using the PicoSAT Solver of Biere [17]), which we conducted on randomly generated synthetic data (of size up to $m = n = 1000$), as well as on a real-world dataset (*animals with attributes* dataset [18] with $m = 50$ and $n = 85$), we found that we can solve instances up to $k = \ell = 11$ using the above CNF-SAT approach. In our experiments, we first computed an upper and a lower bound on the optimal cost value c and then created the CNF-SAT instances for

decreasing values for c, starting from the upper bound. The upper and the lower bound have been obtained as follows: Given a (k,ℓ)-Co-Clustering$_\infty$ instance on $\mathcal{A}$, solve (k,n)-Co-Clustering$_\infty$ and (m,ℓ)-Co-Clustering$_\infty$ separately for input matrix $\mathcal{A}$. Let $(\mathcal{I}_1,\mathcal{J}_1)$ and $(\mathcal{I}_2,\mathcal{J}_2)$ denote the (k,n)- and (m,ℓ)-co-clustering, respectively, and let their costs be $c_1 := \text{cost}(\mathcal{I}_1,\mathcal{J}_1)$ and $c_2 := \text{cost}(\mathcal{I}_2,\mathcal{J}_2)$. We take $\max\{c_1,c_2\}$ as a lower bound and c_1+c_2 as an upper bound on the optimal cost value for an optimal (k,ℓ)-co-clustering of $\mathcal{A}$. It is straightforward to argue the correctness of the lower bound, and we next show that c_1+c_2 is an upper bound. Consider any pair $(i,j),(i',j') \in [m]\times[n]$, such that i and i' are in the same row block of $\mathcal{I}_1$, and j and j' are in the same column block of $\mathcal{J}_2$ (that is, a_{ij} and $a_{i'j'}$ are in the same cluster). Then, it holds $|a_{ij}-a_{i'j'}| \le |a_{ij}-a_{i'j}|+|a_{i'j}-a_{i'j'}| \le c_1+c_2$. Hence, just taking the row partitions from $(\mathcal{I}_1,\mathcal{J}_1)$ and the column partitions from $(\mathcal{I}_2,\mathcal{J}_2)$ gives a combined (k,ℓ)-co-clustering of cost at most c_1+c_2.

From a theoretical perspective, the above naive approach of solving Co-Clustering$_\infty$ via CNF-SAT does not yield any improvement in terms of polynomial-time solvability. Therefore, we now describe a different approach, which leads to some polynomial-time solvable special cases. To this end, we introduce the concept of *cluster boundaries*, which are basically lower and upper bounds for the values in a cluster of a co-clustering. Formally, given two integers k,ℓ, an alphabet Σ and a cost c, we define a cluster boundary to be a matrix $\mathcal{U} = (u_{rs}) \in \Sigma^{k\times\ell}$. We say that a (k,ℓ)-co-clustering of $\mathcal{A}$ *satisfies* a cluster boundary $\mathcal{U}$ if $\mathcal{A}_{rs} \subseteq [u_{rs}, u_{rs}+c]$ for all $(r,s) \in [k]\times[\ell]$. It can easily be seen that a given (k,ℓ)-co-clustering has cost at most c if and only if it satisfies at least one cluster boundary (u_{rs}), namely the one with $u_{rs} = \min \mathcal{A}_{rs}$.

The following "subtask" of Co-Clustering$_\infty$ can be reduced to a certain CNF-SAT instance: Given a cluster boundary $\mathcal{U}$ and a Co-Clustering$_\infty$ instance I, find a co-clustering for I that satisfies $\mathcal{U}$. The polynomial-time reduction provided by the following lemma can be used to obtain exact Co-Clustering$_\infty$ solutions with the help of SAT solvers, and we use it in our subsequent algorithms.

Lemma 1. *Given a* Co-Clustering$_\infty$*-instance* $(\mathcal{A},k,\ell,c)$ *and a cluster boundary* $\mathcal{U}$*, one can construct in polynomial time a* CNF-SAT *instance* ϕ *with at most* $\max\{k,\ell,2\}$ *variables per clause, such that* ϕ *is satisfiable if and only if there is a* (k,ℓ)*-co-clustering of* $\mathcal{A}$*, which satisfies* $\mathcal{U}$*.*

Proof. Given an instance $(\mathcal{A},k,l,c)$ of Co-Clustering$_\infty$ and a cluster boundary $\mathcal{U} = (u_{rs}) \in \Sigma^{k\times\ell}$, we define the following Boolean variables: For each $(i,r) \in [m]\times[k]$, the variable $x_{i,r}$ represents the expression "row i could be put into row block I_r". Similarly, for each $(j,s) \in [n]\times[\ell]$, the variable $y_{j,s}$ represents that "column j could be put into column block J_s".

We now define a Boolean CNF formula $\phi_{\mathcal{A},\mathcal{U}}$ containing the following clauses: a clause $R_i := (x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,k})$ for each row $i \in [m]$ and a clause $C_j := (y_{j,1} \vee y_{j,2} \vee \ldots \vee y_{j,\ell})$ for each column $j \in [n]$. Additionally, for each $(i,j) \in [m]\times[n]$ and each $(r,s) \in [k]\times[\ell]$, such that element a_{ij} does not fit into the cluster boundary at coordinate (r,s), that is $a_{ij} \notin [u_{rs}, u_{rs}+c]$, there is a clause $B_{ijrs} := (\neg x_{i,r} \vee \neg y_{j,s})$. Note that the clauses R_i and C_j ensure that row i and column j are put into some row and some column block, respectively. The clause B_{ijrs} expresses that it is impossible to have both row i in block I_r and column j in block J_s if a_{ij} does not satisfy $u_{rs} \le a_{ij} \le u_{rs}+c$. Clearly, $\phi_{\mathcal{A},\mathcal{U}}$ is satisfiable if and only if there exists a (k,ℓ)-co-clustering of $\mathcal{A}$ satisfying the cluster boundary $\mathcal{U}$. Note that $\phi_{\mathcal{A},\mathcal{U}}$ consists of $km+\ell n$ variables and $O(mnk\ell)$ clauses. □

Using Lemma 1, we can solve Co-Clustering$_\infty$ by solving $O(|\Sigma|^{k\ell})$ many CNF-SAT instances (one for each possible cluster boundary) with $km+\ell n$ variables and $O(mnk\ell)$ clauses of size at most $\max\{k,\ell,2\}$. We also implemented this approach for comparison with the straightforward reduction to CNF-SAT above. The bottleneck of this approach, however, is the number of possible cluster boundaries, which grows extremely quickly. While a single CNF-SAT instance can be solved quickly, generating all possible cluster boundaries together with the corresponding CNF formulas

becomes quite expensive, such that we could only solve instances with very small values of $|\Sigma| \leq 4$ and $k \leq \ell \leq 5$.

4.2. Polynomial-Time Solvability

We first present a simple and efficient algorithm for $(1,*)$-Co-Clustering$_\infty$, that is the variant where all rows belong to one row block.

Theorem 4. $(1,*)$-Co-Clustering$_\infty$ *is solvable in* $O(n(m+\log n))$ *time.*

Proof. We show that Algorithm 1 solves $(1,*)$-Co-Clustering$_\infty$. In fact, it even computes the minimum ℓ', such that $\mathcal{A}$ has a $(1,\ell')$-co-clustering of cost c. The overall idea is that with only one row block all entries of a column j are contained in a cluster in any solution, and thus, it suffices to consider only the minimum α_j and the maximum β_j value in column j. More precisely, for a column block $J \subseteq [n]$ of a solution, it follows that $\max\{\beta_j \mid j \in J\} - \min\{\alpha_j \mid j \in J\} \leq c$. The algorithm starts with the column j_1 that contains the overall minimum value α_{j_1} of the input matrix, that is $\alpha_{j_1} = \min\{\alpha_j \mid j \in [n]\}$. Clearly, j_1 has to be contained in some column block, say J_1. The algorithm then adds all other columns j to J_1 where $\beta_j \leq \alpha_{j_1} + c$, removes the columns J_1 from the matrix and recursively proceeds with the column containing the minimum value of the remaining matrix. We continue with the correctness of the described procedure.

Algorithm 1: Algorithm for $(1,*)$-Co-Clustering$_\infty$.

Input: $\mathcal{A} \in \mathbb{R}^{m \times n}$, $\ell \geq 1$, $c \geq 0$.
Output: A partition of $[n]$ into at most ℓ blocks yielding a cost of at most c, or `no` if no such partition exists.

```
1  for j ← 1 to n do
2      α_j ← min{a_ij | 1 ≤ i ≤ m};
3      β_j ← max{a_ij | 1 ≤ i ≤ m};
4      if β_j − α_j > c then
5          return no;
6  N ← [n];
7  for s ← 1 to ℓ do
8      Let j_s ∈ N be the index such that α_{j_s} is minimal;
9      J_s ← {j ∈ N | β_j − α_{j_s} ≤ c};
10     N ← N \ J_s;
11     if N = ∅ then
12         return (J_1, ..., J_s);
13 return no;
```

If Algorithm 1 returns $(J_1, \ldots, J_{\ell'})$ at Line 12, then this is a column partition into $\ell' \leq \ell$ blocks satisfying the cost constraint. First, it is a partition by construction: the sets J_s are successively removed from $\mathcal{N}$ until it is empty. Now, let $s \in [\ell']$. Then, for all $j \in J_s$, it holds $\alpha_j \geq \alpha_{j_s}$ (by definition of j_s) and $\beta_j \leq \alpha_{j_s} + c$ (by definition of J_s). Thus, $\mathcal{A}_{1s} \subseteq [\alpha_{j_s}, \alpha_{j_s} + c]$ holds for all $s \in [\ell']$, which yields $\text{cost}_\infty(\{[m]\}, \{J_1, \ldots, J_{\ell'}\}) \leq c$.

Otherwise, if Algorithm 1 returns `no` in Line 5, then it is clearly a no-instance, since the difference between the maximum and the minimum value in a column is larger than c. If `no` is returned in Line 13, then the algorithm has computed column indices j_s and column blocks J_s for each $s \in [\ell]$, and there still exists at least one index $j_{\ell+1}$ in $\mathcal{N}$ when the algorithm terminates. We claim that the columns $j_1, \ldots, j_{\ell+1}$ all have to be in different blocks in any solution. To see this, consider any

$s, s' \in [\ell+1]$ with $s < s'$. By construction, $j_{s'} \notin J_s$. Therefore, $\beta_{j_{s'}} > \alpha_{j_s} + c$ holds, and columns j_s and $j_{s'}$ contain elements with distance more than c. Thus, in any co-clustering with cost at most c, columns $j_1, \ldots, j_{\ell+1}$ must be in different blocks, which is impossible with only ℓ blocks. Hence, we indeed have a no-instance.

The time complexity is seen as follows. The first loop examines in $O(mn)$ time all elements of the matrix. The second loop can be performed in $O(n)$ time if the α_j and the β_j are sorted beforehand, requiring $O(n \log n)$ time. Overall, the running time is in $O(n(m + \log n))$. □

From now on, we focus on the $k = 2$ case, that is we need to partition the rows into two blocks. We first consider the simplest case, where also $\ell = 2$.

Theorem 5. $(2,2)$-Co-Clustering$_\infty$ *is solvable in* $O(|\Sigma|^2 mn)$ *time.*

Proof. We use the reduction to CNF-SAT provided by Lemma 1. First, note that a cluster boundary $\mathcal{U} \in \Sigma^{2\times 2}$ can only be satisfied if it contains the elements $\min \Sigma$ and $\min\{a \in \Sigma \mid a \geq \max \Sigma - c\}$. The algorithm enumerates all $O(|\Sigma|^2)$ of these cluster boundaries. For a fixed $\mathcal{U}$, we construct the Boolean formula $\phi_{\mathcal{A}\mathcal{U}}$. Observe that this formula is in two-CNF form: The formula consists of k-clauses, ℓ-clauses and 2-clauses, and we have $k = \ell = 2$. Hence, we can determine whether it is satisfiable in linear time [19] (note that the size of the formula is in $O(mn)$). Overall, the input is a yes-instance if and only if $\phi_{\mathcal{A}\mathcal{U}}$ is satisfiable for some cluster boundary $\mathcal{U}$. □

Finally, we show that it is possible to extend the above result to any number of column blocks for size-three alphabets.

Theorem 6. $(2,*)$-Co-Clustering$_\infty$ *is* $O(mn)$*-time solvable for* $|\Sigma| = 3$.

Proof. Let $I = (\mathcal{A} \in \{\alpha, \beta, \gamma\}^{m\times n}, k = 2, \ell, c)$ be a $(2,*)$-Co-Clustering$_\infty$ instance. We assume without loss of generality that $\alpha < \beta < \gamma$. The case $\ell \leq 2$ is solvable in $O(mn)$ time by Theorem 5. Hence, it remains to consider the case $\ell \geq 3$. As $|\Sigma| = 3$, there are four potential values for a minimum-cost $(2,\ell)$-co-clustering. Namely, cost zero (all cluster entries are equal), cost $\beta - \alpha$, cost $\gamma - \beta$ and cost $\gamma - \alpha$. Since any $(2,\ell)$-co-clustering is of cost at most $\gamma - \alpha$ and because it can be checked in $O(mn)$ time whether there is a $(2,\ell)$-co-clustering of cost zero (Observation 1), it remains to check whether there is a $(2,\ell)$-co-clustering between these two extreme cases, that is for $c \in \{\beta - \alpha, \gamma - \beta\}$.

Avoiding a pair $(x,y) \in \{\alpha,\beta,\gamma\}^2$ means to find a co-clustering without a cluster containing x and y. If $c = \max\{\beta - \alpha, \gamma - \beta\}$ (Case 1), then the problem comes down to finding a $(2,\ell)$-co-clustering avoiding the pair (α,γ). Otherwise (Case 2), the problem is to find a $(2,\ell)$-co-clustering avoiding the pair (α,γ) and, additionally, either (α,β) or (β,γ).

Case 1. Finding a $(2,\ell)$-co-clustering avoiding (α,γ):

In this case, we substitute $\alpha := 0$, $\beta := 1$ and $\gamma := 2$. We describe an algorithm for finding a $(2,\ell)$-co-clustering of cost one (avoiding $(0,2)$). We assume that there is no $(2,\ell-1)$-co-clustering of cost one (iterating over all values from two to ℓ). Consider a $(2,\ell)$-co-clustering $(\mathcal{I}, \mathcal{J} = \{J_1, \ldots, J_\ell\})$ of cost one, that is for all $(r,s) \in [2] \times [\ell]$, it holds $\mathcal{A}_{rs} \subseteq \{0,1\}$ or $\mathcal{A}_{rs} \subseteq \{1,2\}$. For $s \neq t \in [\ell]$, let $(\mathcal{I}, \mathcal{J}_{st} := \mathcal{J} \setminus \{J_s, J_t\} \cup \{J_s \cup J_t\})$ denote the $(2,\ell-1)$-co-clustering where the column blocks J_s and J_t are merged. By assumption, for all $s \neq t \in [\ell]$, it holds that $\text{cost}_\infty(\mathcal{I}, \mathcal{J}_{st}) > 1$, since otherwise, we have found a $(2,\ell-1)$-co-clustering of cost one. It follows that $\{0,2\} \subseteq \mathcal{A}_{1s} \cup \mathcal{A}_{1t}$ or $\{0,2\} \subseteq \mathcal{A}_{2s} \cup \mathcal{A}_{2t}$ holds for all $s \neq t \in [\ell]$. This can only be true for $|\mathcal{J}| = 2$.

This proves that there is a $(2,\ell)$-co-clustering of cost one if and only if there is a $(2,2)$-co-clustering of cost one. Hence, Theorem 5 shows that this case is $O(mn)$-time solvable.

Case 2. Finding a $(2,\ell)$-co-clustering avoiding (α,γ) and (α,β) (or (β,γ)):

In this case, we substitute $\alpha := 0$, $\gamma := 1$ and $\beta := 1$ if (α,β) has to be avoided, or $\beta := 0$ if (β,γ) has to be avoided. It remains to determine whether there is a $(2,\ell)$-co-clustering with cost zero, which can be done in $O(mn)$ time due to Observation 1. □

4.3. Fixed-Parameter Tractability

We develop an algorithm solving $(2,*)$-Co-Clustering$_\infty$ for $c = 1$ based on our reduction to CNF-SAT (see Lemma 1). The main idea is, given matrix $\mathcal{A}$ and cluster boundary $\mathcal{U}$, to simplify the Boolean formula $\phi_{\mathcal{A}\mathcal{U}}$ into a 2-SAT formula, which can be solved efficiently. This is made possible by the constraint on the cost, which imposes a very specific structure on the cluster boundary. This approach requires to enumerate all (exponentially many) possible cluster boundaries, but yields fixed-parameter tractability for the combined parameter $(\ell,|\Sigma|)$.

Theorem 7. $(2,*)$-Co-Clustering$_\infty$ *is* $O(|\Sigma|^{3\ell}n^2m^2)$*-time solvable for* $c = 1$.

In the following, we prove Theorem 7 in several steps.

A first sub-result for the proof of Theorem 7 is the following lemma, which we use to solve the case where the number 2^m of possible row partitions is less than $|\Sigma|^\ell$.

Lemma 2. *For a fixed row partition* $\mathcal{I}$*, one can solve* Co-Clustering$_\infty$ *in* $O(|\Sigma|^{k\ell}mn\ell)$ *time. Moreover,* Co-Clustering$_\infty$ *is fixed-parameter tractable with respect to the combined parameter* (m,k,ℓ,c).

Proof. Given a fixed row partition $\mathcal{I}$, the algorithm enumerates all $|\Sigma|^{k\ell}$ different cluster boundaries $\mathcal{U} = (u_{rs})$. We say that a given column j *fits* in column block J_s if, for each $r \in [k]$ and $i \in I_r$, we have $a_{ij} \in [u_{rs}, u_{rs}+c]$ (this can be decided in $O(m)$ time for any pair (j,s)). The input is a yes-instance if and only if for some cluster boundary $\mathcal{U}$, every column fits in at least one column block.

Fixed-parameter tractability with respect to (m,k,ℓ,c) is obtained from two simple further observations. First, all possible row partitions can be enumerated in $O(k^m)$ time. Second, since each of the $k\ell$ clusters contains at most $c+1$ different values, the alphabet size $|\Sigma|$ for yes-instances is upper-bounded by $(c+1)k\ell$. □

The following lemma, also used for the proof of Theorem 7, yields that even for the most difficult instances, there is no need to consider more than two column clusters to which any column can be assigned.

Lemma 3. *Let* $I = (\mathcal{A} \in \Sigma^{m\times n}, k = 2, \ell, c = 1)$ *be an instance of* $(2,*)$-Co-Clustering$_\infty$*,* h_1 *be an integer,* $0 < h_1 < m$*, and* $\mathcal{U} = (u_{rs})$ *be a cluster boundary with pairwise different columns, such that* $|u_{1s} - u_{2s}| = 1$ *for all* $s \in [\ell]$.

Then, for any column $j \in [n]$*, two indices* $s_{j,1}$ *and* $s_{j,2}$ *can be computed in time* $O(mn)$*, such that if I has a solution* $(\{I_1,I_2\},\{J_1,\ldots,J_\ell\})$ *satisfying* $\mathcal{U}$ *with* $|I_1| = h_1$*, then it has one where each column* j *is assigned to either* $J_{s_{j,1}}$ *or* $J_{s_{j,2}}$.

Proof. We write $h_2 = m - h_1$ ($h_2 = |I_2| > 0$ for any solution with $h_1 = |I_1|$). Given a column $j \in [n]$ and any element $a \in \Sigma$, we write $\sharp_j^a$ for the number of entries with value a in column j.

Consider a column block $J_s \subseteq [n]$, $s \in [\ell]$. Write α,β,γ for the three values, such that $U_{1s} \setminus U_{2s} = \{\alpha\}$, $U_{1s} \cap U_{2s} = \{\beta\}$ and $U_{2s} \setminus U_{1s} = \{\gamma\}$. Note that $\{\alpha,\beta,\gamma\} = \{\beta-1,\beta,\beta+1\}$. We say that column j *fits* into column block J_s if the following three conditions hold:

1. $\sharp_j^x = 0$ for any $x \notin \{\alpha,\beta,\gamma\}$,
2. $\sharp_j^\alpha \leq h_1$ and
3. $\sharp_j^\gamma \leq h_2$.

Note that if Condition (1) is violated, then the column contains an element that is neither in U_{1s} nor in U_{2s}. If Condition (2) (respectively Condition (3)) is violated, then there are more than h_1 (respectively h_2) rows that have to be in row block I_1 (respectively I_2). Thus, if j does not fit into a column block J_s, then there is no solution where $j \in J_s$. We now need to find out, for each column, to which fitting column blocks it should be assigned.

Intuitively, we now prove that in most cases, a column has at most two fitting column blocks and, in the remaining cases, at most two pairs of "equivalent" column blocks.

Consider a given column $j \in [n]$. Write $a = \min\{a_{ij} \mid i \in [m]\}$ and $b = \max\{a_{ij} \mid i \in [m]\}$. If $b \geq a + 3$, then Condition (1) is always violated: j does not fit into any column block, and the instance is a no-instance. If $b = a + 2$, then, again, by Condition (1), j can only fit into a column block where $\{u_{1s}, u_{2s}\} = \{a, a + 1\}$. There are at most two such column blocks: we write $s_{j,1}$ and $s_{j,2}$ for their indices ($s_{j,1} = s_{j,2}$ if a single column block fits). The other easy case is when $b = a$, *i.e.*, all values in column j are equal to a. If j fits into column block J_s, then, with Conditions (2) and (3), $a \in U_{1s} \cap U_{2s}$, and J_s is one of the at most two column blocks having $\beta = a$: again, we write $s_{j,1}$ and $s_{j,2}$ for their indices.

Finally, consider a column j with $b = a + 1$, and let $s \in [\ell]$ be such that j fits into J_s. Then, by Condition (1), the "middle-value" for column block J_s is $\beta \in \{a, b\}$. The pair (u_{1s}, u_{2s}) must be from $\{(a-1, a), (a, a-1), (a, b), (b, a)\}$. We write $J_{s_1}, \ldots, J_{s_4}$ for the four column blocks (if they exist) corresponding to these four cases. We define $s_{j,1} = s_1$ if j fits into J_{s_1}, and $s_{j,1} = s_3$ otherwise. Similarly, we define $s_{j,2} = s_2$ if j fits into J_{s_2}, and $s_{j,2} = s_4$ otherwise.

Consider a solution assigning j to $s^* \in \{s_1, s_3\}$, with $s^* \neq s_{j,1}$. Since j must fit into J_{s^*}, the only possibility is that $s^* = s_3$ and $s_{j,1} = s_1$. Thus, j fits into both J_{s_1} and J_{s_3}, so Conditions (2) and (3) imply $\sharp_j^a \leq h_1$ and $\sharp_j^b \leq h_2$. Since $\sharp_j^a + \sharp_j^b = h_1 + h_2 = m$, we have $\sharp_j^a = h_1$ and $\sharp_j^b = h_2$. Thus, placing j in either column block yields the same row partition, namely $I_1 = \{i \mid a_{ij} = a\}$ and $I_2 = \{i \mid a_{ij} = b\}$. Hence, the solution assigning j to J_{s_3}, can assign it to $J_{s_1} = J_{s_{j,1}}$, instead, without any further need for modification.

Similarly, with s_2 and s_4, any solution assigning j to J_{s_2} or J_{s_4} can assign it to $J_{s_{j,2}}$ without any other modification. Thus, since any solution must assign j to one of $\{J_{s_1}, \ldots, J_{s_4}\}$, it can assign it to one of $\{J_{s_{j,1}}, J_{s_{j,2}}\}$ instead. □

We now give the proof of Theorem 7.

Proof. Let $I = (\mathcal{A} \in \Sigma^{m \times n}, k = 2, \ell, c = 1)$ be a $(2, *)$-Co-Clustering$_\infty$ instance. The proof is by induction on ℓ. For $\ell = 1$, the problem is solvable in $O(n(m + \log n))$ time (Theorem 4). We now consider general values of ℓ. Note that if ℓ is large compared to m (that is, $2^m < |\Sigma|^\ell$), then one can directly guess the row partition and run the algorithm of Lemma 2. Thus, for the running time bound, we now assume that $\ell < m$. By Observation 2, we can assume that $\Sigma \subset \mathbb{Z}$.

Given a $(2, \ell)$-co-clustering $(\mathcal{I} = \{\{1\}, \{2\}\}, \mathcal{J})$, a cluster boundary $\mathcal{U} = (u_{rs})$ satisfied by $(\mathcal{I}, \mathcal{J})$, and $U_{rs} = [u_{rs}, u_{rs} + c]$, each column block $J_s \in \mathcal{J}$ is said to be:

- with *equal* bounds if $U_{1s} = U_{2s}$,
- with *non-overlapping* bounds if $U_{1s} \cap U_{2s} = \emptyset$,
- with *properly overlapping* bounds otherwise.

We first show that instances implying a solution containing at least one column block with equal or non-overlapping bounds can easily be dealt with.

Claim 1. *If the solution contains a column-block with equal bounds, then it can be computed in* $O(|\Sigma|^{2\ell} n^2 m^2)$ *time.*

Proof. Assume, without loss of generality, that the last column block, J_ℓ, has equal bounds. We try all possible values of $u = u_{1\ell}$. Note that column block J_ℓ imposes no restrictions on the row

partition. Hence, it can be determined independently of the rest of the co-clustering. More precisely, any column with all values in $U_{1\ell} = U_{2\ell} = [u, u+c]$ can be put into this block, and all other columns have to end up in the $\ell - 1$ other blocks, thus forming an instance of $(2, \ell-1)$-Co-Clustering$_\infty$. By induction, each of these cases can be tested in $O(|\Sigma|^{2(\ell-1)} n^2 m(\ell-1))$ time. Since we test all values of u, this procedure finds a solution with a column block having equal bounds in $O(|\Sigma| \cdot |\Sigma|^{2(\ell-1)} n^2 m(\ell-1)) = O(|\Sigma|^{2\ell} n^2 m^2)$ time. □

Claim 2. *If the solution contains a (non-empty) column-block with non-overlapping bounds, then it can be computed in $O(|\Sigma|^{2\ell} n^2 m^2)$ time.*

Proof. Write s for the index of the column block J_s with non-overlapping bounds, and assume that, without loss of generality, $u_{1s} + c < u_{2s}$. We try all possible values of $u = u_{2s}$, and we examine each column $j \in [n]$. We remark that the row partition is entirely determined by column j if it belongs to column block J_s. That is, if $j \in J_s$, then $I_1 = \{i \mid a_{ij} < u\}$ and $I_2 = \{i \mid a_{ij} \geq u\}$. Using the algorithm described in Lemma 2, we deduce the column partition in $O(|\Sigma|^{2\ell-1} nm\ell)$ time, which is bounded by $O(|\Sigma|^{2\ell} n^2 m^2)$. □

We can now safely assume that the solution contains only column blocks with properly overlapping bounds. In a first step, we guess the values of the cluster boundary $\mathcal{U} = (u_{rs})$. Note that, for each $s \in [\ell]$, we only need to consider the cases where $0 < |u_{1s} - u_{2s}| \leq c$, that is, for $c = 1$, we have $u_{2s} = u_{1s} \pm 1$. Note also that, for any two distinct column blocks J_s and $J_{s'}$, we have $u_{1s} \neq u_{1,s'}$ or $u_{2s} \neq u_{2,s'}$. We then enumerate all possible values of $h_1 = |I_1| > 0$ (the *height* of the first row block), and we write $h_2 = m - h_1 = |I_2| > 0$. Overall, there are at most $(2|\Sigma|)^\ell m$ cases to consider.

Using Lemma 3, we compute integers $s_{j,1}, s_{j,2}$ for each column j, such that any solution satisfying the above conditions (cluster boundary $\mathcal{U}$ and $|I_1| = h_1$) can be assumed to assign each column j to one of $J_{s_{j,1}}$ or $J_{s_{j,2}}$.

We now introduce a 2-SAT formula allowing us to simultaneously assign the rows and columns to the possible blocks. Let $\phi_{\mathcal{A}\mathcal{U}}$ be the formula as provided by Lemma 1. Create a formula ϕ' from $\phi_{\mathcal{A}\mathcal{U}}$ where, for each column $j \in [n]$, the column clause C_j is replaced by the smaller clause $C'_j := (y_{j,s_{j,1}} \vee y_{j,s_{j,2}})$. Note that ϕ' is a 2-SAT formula, since all other clauses R_i or B_{ijrs} already contain at most two literals.

If ϕ' is satisfiable, then $\phi_{\mathcal{A}\mathcal{U}}$ is satisfiable, and $\mathcal{A}$ admits a $(2,\ell)$-co-clustering satisfying $\mathcal{U}$. Conversely, if $\mathcal{A}$ admits a $(2,\ell)$-co-clustering satisfying $\mathcal{U}$ with $|I_1| = h_1$, then, by the discussion above, there exists a co-clustering where each column j is in one of the column blocks $J_{s_{j,1}}$ or $J_{s_{j,2}}$. In the corresponding Boolean assignment, each clause of $\phi_{\mathcal{A}\mathcal{U}}$ is satisfied and each new column clause of ϕ' is also satisfied. Hence, ϕ' is satisfiable. Overall, for each cluster boundary $\mathcal{U}$ and each h_1, we construct and solve the formula ϕ' defined above. The matrix $\mathcal{A}$ admits a $(2,\ell)$-co-clustering of cost one if and only if ϕ' is satisfiable for some $\mathcal{U}$ and h_1.

The running time for constructing and solving the formula ϕ', for any fixed cluster boundary $\mathcal{U}$ and any height $h_1 \in [m]$, is in $O(nm)$, which gives a running time of $O((2|\Sigma|)^\ell nm^2)$ for this last part. Overall, the running time is thus $O(|\Sigma|^{2\ell} n^2 m^2 + |\Sigma|^{2\ell} n^2 m^2 + (2|\Sigma|)^\ell nm^2) = O(|\Sigma|^{2\ell} n^2 m^2)$. □

Finally, we obtain the following simple corollary.

Corollary 2. $(2,*)$-Co-Clustering$_\infty$ *with $c = 1$ is fixed-parameter tractable with respect to parameter $|\Sigma|$ and with respect to parameter ℓ.*

Proof. Theorem 7 presents an FPT-algorithm with respect to the combined parameter $(|\Sigma|, \ell)$. For $(2,*)$-Co-Clustering$_\infty$ with $c = 1$, both parameters can be polynomially upper-bounded within each other. Indeed, $\ell < |\Sigma|^2$ (otherwise, there are two column blocks with identical cluster boundaries, which could be merged) and $|\Sigma| < 2(c+1)\ell = 4\ell$ (each column block may contain two intervals, each covering at most $c+1$ elements). □

5. Conclusions

Contrasting previous theoretical work on polynomial-time approximation algorithms [2,8], we started to closely investigate the time complexity of exactly solving the NP-hard Co-Clustering$_\infty$ problem, contributing a detailed view of its computational complexity landscape. Refer to Table 1 for an overview on most of our results.

Several open questions derive from our work. Perhaps the most pressing open question is whether the case $k = 2$ and $\ell \geq 3$ is polynomial-time solvable or NP-hard in general. So far, we only know that $(2,*)$-Co-Clustering$_\infty$ is polynomial-time solvable for ternary matrices (Theorem 6). Another open question is the computational complexity of higher-dimensional co-clustering versions, e.g., on three-dimensional tensors as input (the most basic case here corresponds to (2,2,2)-Co-Clustering$_\infty$, that is partitioning each dimension into two subsets). Indeed, other than the techniques for deriving approximation algorithms [2,8], our exact methods do not seem to generalize to higher dimensions. Last, but not least, we do not know whether Consecutive Co-Clustering$_\infty$ is fixed-parameter tractable or W[1]-hard with respect to the combined parameter (k, ℓ).

We conclude with the following more abstract vision on future research: Note that for the maximum norm, the cost value c defines a "conflict relation" on the values occurring in the input matrix. That is, for any two numbers $\sigma, \sigma' \in \Sigma$ with $|\sigma - \sigma'| > c$, we know that they must end up in different clusters. These conflict pairs completely determine all constraints of a solution, since all other pairs can be grouped arbitrarily. This observation can be generalized to a graph model. Given a "conflict relation" $R \subseteq \binom{\Sigma}{2}$ determining which pairs are not allowed to be put together into a cluster, we can define the "conflict graph" (Σ, R). Studying co-clusterings in the context of such conflict graphs and their structural properties could be a promising and fruitful direction for future research.

Acknowledgments: Laurent Bulteau: Main work done while affiliated with TU Berlin, supported by the Alexander von Humboldt Foundation, Bonn, Germany. Vincent Froese: Supported by the DFG, project DAMM (NI 369/13). We thank Stéphane Vialette (Université Paris-Est Marne-la-Vallée) for stimulating discussions.

Author Contributions: Laurent Bulteau, Vincent Froese, Sepp Hartung and Rolf Niedermeier drafted, wrote and revised the paper. Laurent Bulteau, Vincent Froese, Sepp Hartung and Rolf Niedermeier conceived and designed experiments. Vincent Froese conducted experiments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Madeira, S.C.; Oliveira, A.L. Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **2004**, *1*, 24–45.
2. Anagnostopoulos, A.; Dasgupta, A.; Kumar, R. A Constant-Factor Approximation Algorithm for Co-clustering. *Theory Comput.* **2012**, *8*, 597–622.
3. Banerjee, A.; Dhillon, I.S.; Ghosh, J.; Merugu, S.; Modha, D.S. A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation. *J. Mach. Learn. Res.* **2007**, *8*, 1919–1986.
4. Tanay, A.; Sharan, R.; Shamir, R. Biclustering Algorithms: A Survey. In *Handbook of Computational Molecular Biology*; Chapman & Hall/CRC: Boca Raton, FL, USA., 2005.
5. Nguyen, S.H.; Skowron, A. Quantization Of Real Value Attributes-Rough Set and Boolean Reasoning Approach. In Proceedings of the Second Joint Annual Conference on Information Sciences, Wrightsville Beach, NC, USA, 28 September–1 October 1995; pp. 34–37.
6. Chlebus, B.S.; Nguyen, S.H. On Finding Optimal Discretizations for Two Attributes, In Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC'98), Warsaw, Poland, 22–26 June 1998; pp. 537–544.
7. Nguyen, H.S. Approximate Boolean Reasoning: Foundations and Applications in Data Mining. In *Transactions on Rough Sets V*; Springer: Berlin Heidelberg, Germany, 2006; pp. 334–506.

8. Jegelka, S.; Sra, S.; Banerjee, A. Approximation Algorithms for Tensor Clustering. In Proceedings of the 20th International Conference of Algorithmic Learning Theory (ALT'09), Porto, Portugal, 3–5 October 2009; pp. 368–383.
9. Hartigan, J.A. Direct clustering of a data matrix. *J. Am. Stat. Assoc.* **1972**, *67*, 123–129.
10. Califano, A.; Stolovitzky, G.; Tu, Y. Analysis of Gene Expression Microarrays for Phenotype Classification. In Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB'00), AAAI, San Diego, CA, USA, 16–23 August 2000; pp. 75–85.
11. Wulff, S.; Urner, R.; Ben-David, S. Monochromatic Bi-Clustering. In Proceedings of the 30th International Conference on Machine Learning (ICML'13), Atlanta, GA, USA, 16–21 June 2013; pp. 145–153.
12. Cygan, M.; Fomin, F.V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; Saurabh, S. *Parameterized Algorithms*; Springer International Publishing: Switzerland, 2015.
13. Downey, R.G.; Fellows, M.R. *Fundamentals of Parameterized Complexity*; Springer: London, UK, 2013.
14. Niedermeier, R. *Invitation to Fixed-Parameter Algorithms*; Oxford University Press: Oxford, UK, 2006.
15. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman and Company: New York, NY, USA, 1979.
16. Fowler, R.J.; Paterson, M.S.; Tanimoto, S.L. Optimal Packing and Covering in the Plane are NP-Complete. *Inf. Process. Lett.* **1981**, *12*, 133–137.
17. Biere, A. PicoSAT Essentials. *J. Satisf. Boolean Model. Comput.* **2008**, *4*, 75–97.
18. Lampert, C.H.; Nickisch, H.; Harmeling, S. Attribute-Based Classification for Zero-Shot Visual Object Categorization *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *36*, 453–465.
19. Aspvall, B.; Plass, M.F.; Tarjan, R.E. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Inf. Process. Lett.* **1979**, *8*, 121–123.