Co-Clustering Under the Maximum Norm

Laurent Bulteau[⋆], Vincent Froese[⋆⋆], Sepp Hartung, and Rolf Niedermeier

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
`l.bulteau@campus.tu-berlin.de`,{`vincent.froese, sepp.hartung, rolf.niedermeier`}`@tu-berlin.de`

Abstract. Co-clustering, that is, partitioning a 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 $\{0, 1, 2\}$.

1 Introduction

Co-clustering, also known as *biclustering*, performs a simultaneous clustering of the rows and columns of a data matrix. Roughly speaking, the problem is, given a numerical input matrix A, to partition the rows and columns of A into subsets minimizing a given *cost* function (measuring "homogeneity"). For a given subset of rows I and a subset of columns J, 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 [10]. We focus on the non-overlapping variant which can be stated as follows.

CO-CLUSTERING$_{\mathcal{L}}$

Input: A matrix $A \in \mathbb{R}^{m \times n}$ and two positive integers $k, \ell \in \mathbb{N}$.

Task: Find a partition of A's rows into k subsets and a partition of 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

[⋆] Supported by the Alexander von Humboldt Foundation, Bonn, Germany.
[⋆⋆] Supported by the DFG project DAMM (NI 369/13).

To appear in *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC' 14), Jeonju, Korea, December 2014.* © *Springer.*

systems to election analysis [1, 3, 10]. 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" [10], 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. [1], we further refine and strengthen the theoretical investigations on the computational complexity of a natural special case of Co-Clustering$_{\mathcal{L}}$ for the maximum norm $\mathcal{L} = L_\infty$.

Anagnostopoulos et al. [1] 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. [9] 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, we investigate a more limited scenario, focussing on Co-Clustering$_\infty$, where the problem comes down to minimizing the maximum distance between entries of a cluster. 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.

Related work. Our main point of reference is the work of Anagnostopoulos et al. [1]. 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 [10][1] provides an excellent overview on the many variations of Co-Clustering$_{\mathcal{L}}$, there called biclustering, and discusses many applications in bioinformatics and beyond. In particular, they also discuss the special case where the goal is to partition into uniform clusters [8] (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 0) but allowing some variation between maximum and minimum cluster entries. Finally, Califano et al. [4] aimed at clusterings where in each submatrix the distance between entries within each row and within each column is upper-bounded. Except for the work of Anagnostopoulos et al. [1], all investigations mentioned above are empirical in nature.

Our contributions. In terms of defining "cluster homogeneity", we focus on minimizing the maximum distance between two entries within a cluster (maximum

[1] According to Google Scholar, accessed September 2014, cited more than 1350 times.

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), where $*$ indicates a value being part of the input and $\circledast$ indicates that the corresponding value(s) is/are the parameter.

m	$\|\Sigma\|$	k	ℓ	c	Complexity
$*$	$*$	$*$	$*$	0	P [Observation 1]
$*$	2	$*$	$*$	$*$	P [Observation 1]
$*$	$*$	1	$*$	$*$	P [Theorem 4]
$*$	$*$	2	2	$*$	P [Theorem 5]
$*$	$*$	2	$\circledast$	1	FPT [Corollary 2]
$\circledast$	$*$	$\circledast$	$\circledast$	$\circledast$	FPT [Lemma 2]
$*$	3	3	3	1	NP-h [Theorem 1]
2	$*$	2	$*$	2	NP-h [Theorem 2]

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. Our main technical contributions are as follows. Concerning the computational intractability results with respect to even strongly restricted cases, we put a lot of effort in 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 $\ell^n \cdot k^m$ to $\binom{n}{\ell} \cdot \binom{m}{k}$) 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 developed a novel reduction to SAT solving (instead of the standard reductions to integer linear programming) which may prove beneficial on the theoretical but also on the practical side. Notably, however, as opposed to previous work on approximation algorithms [1, 9], 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.

Due to the lack of space, several details are deferred to a full version.

2 Formal Definitions and Preliminaries

We use standard terminology for matrices. A matrix $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ consists of m rows and n columns where a_{ij} denotes the entry in row i and column j. We define $[n] := \{1, 2, \dots, n\}$ and $[i, j] := \{i, i+1, \dots, j\}$ for $n, i, j \in \mathbb{N}$. Throughout this paper, we assume that arithmetical operations require constant time.

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

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

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

Fig. 1: The example shows two $(2, 2)$-co-clusterings (middle and right) of the same matrix 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.

Problem definition. We follow the terminology of Anagnostopoulos et al. [1]. For a matrix $A \in \mathbb{R}^{m\times n}$, a (k, ℓ)-*co-clustering* is a pair $(\mathcal{I}, \mathcal{J})$ consisting of a k-partition $\mathcal{I} = \{I_1, \dots, I_k\}$ of the rows of A (more specifically, a partition of the row indices $[m]$) and an ℓ-partition $\mathcal{J} = \{J_1, \dots, J_\ell\}$ of the columns (the column indices $[n]$, respectively) of A. We call the elements of $\mathcal{I}$ ($\mathcal{J}$) 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 $A_{rs} := \{a_{ij} \in A \mid (i, j) \in I_r \times J_s\}$ 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 $\text{cost}(\mathcal{I}, \mathcal{J}) := \max_{(r,s)\in[k]\times[\ell]}(\max A_{rs} - \min A_{rs})$. Herein, $\max A_{rs}$ ($\min A_{rs}$) denotes the maximum (minimum, resp.) entry in A_{rs}.

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

Co-Clustering$_\infty$

Input: A matrix $A \in \mathbb{R}^{m\times n}$, 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 A with $\text{cost}(\mathcal{I}, \mathcal{J}) \leq c$?

See Figure 1 for an introductory example. We define $\Sigma := \{a_{ij} \in A \mid (i, j) \in [m] \times [n]\}$ to be the *alphabet* of the input matrix A (consisting of the values that occur in A). 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 A is equivalent to an (ℓ, k)-co-clustering of the transposed matrix A^T. Hence, we always assume that $k \leq \ell$.

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 polynomial time for cost zero and also for any size-two alphabet.*

Proof. Let $(A, k, \ell, 0)$ be a Co-Clustering$_\infty$ instance. For a (k, ℓ)-co-clustering with cost 0, 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 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 A. $\square$

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 $\Sigma \subseteq \mathbb{R}$, one can find in $O((mn)^2)$ time an equivalent instance with alphabet $\Sigma' \subseteq \mathbb{Z}$ and cost value $c' \in \mathbb{N}$.*

Parameterized algorithmics. We briefly introduce the relevant notions from parameterized algorithmics. A parameterized problem, each instance consisting of the "classical" problem instance I and an integer k, is *fixed-parameter tractable* (FPT) if there is a computable function f and an algorithm solving any instance in $f(k) \cdot |I|^{O(1)}$ time. The corresponding algorithm is called 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 or input matrices with only two rows. We also show NP-hardness of finding co-clusterings where the row and column partitions are only allowed to contain consecutive blocks.

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 reduce from the NP-complete 3-Coloring [7], where the task is to partition the vertex set of an 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 $(A \in \{0,1,2\}^{m \times n}, k := 3, \ell := 3, c := 1)$ as follows. The columns of 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 one. 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. Thus, every co-clustering of A with 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 A with cost at most $c = 1$ if and only if G admits a 3-coloring.

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 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 $\text{cost}(\mathcal{I}, \mathcal{J}) \leq 1$.

Next, assume that $(\mathcal{I}, \{J_1, J_2, J_3\})$ is a $(3, 3)$-co-clustering of A with 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. $\square$

Theorem 1 can even be strengthened further.

Corollary 1. *Co-Clustering$_\infty$ is NP-hard even when $k = m$ (that is, each row is in its own cluster), ℓ is fixed with $\ell \geq 3$, $\Sigma = \{0, 1, 2\}$, and the column blocks are forced to have equal sizes $|J_1| = \ldots = |J_\ell|$.*

Proof (Sketch). Note that the reduction in Theorem 1 can easily be adapted to the NP-hard ℓ-Coloring problem with balanced partition sizes [7]. Note also that the proof holds for any $k \geq 3$. Hence, the problem is NP-hard for $k = m$ row blocks. $\square$

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 $k = m = 2$.*

Proof. We give a polynomial-time reduction from the NP-complete Box Cover problem [6]. 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 2, covering P, that is, $P \subseteq \bigcup_{1 \leq i \leq \ell} S_i$.

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

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 $J_1 := \{i \mid p_i \in P \cap S_1\}$ and $J_j := \{i \mid p_i \in P \cap S_j \setminus (\bigcup_{1\le l<j} S_l)\}$ for all $2 \le j \le \ell$. Note that $(\mathcal{I} = \{\{1\},\{2\}\}, \mathcal{J} = \{J_1, \ldots, J_\ell\})$ is a $(2,\ell)$-co-clustering of A. Moreover, since all points with indices in J_j lie inside a square with side length 2, it holds that each pair of entries in A_{1j} as well as in A_{2j} has distance at most 2, implying $\text{cost}(\mathcal{I}, \mathcal{J}) \le 2$.

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

3.3 Clustering into Consecutive Clusters

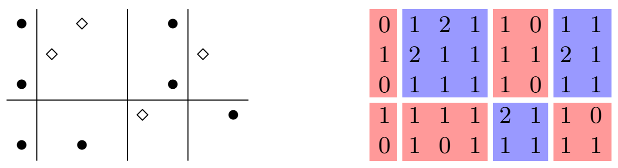
One is tempted to assume that the hardness of the previous special cases of Co-Clustering$_\infty$ is rooted in the fact that we are allowed to choose arbitrary subsets for the corresponding row and column partitions since the problem remains hard even for a constant number of clusters and also with 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 $1 < r_1 < \ldots < r_{k-1} \le m$ and column indices $C = \{c_1, \ldots, c_{\ell-1}\}$ with $1 < c_1 < \ldots < c_{\ell-1} \le n$, the corresponding *consecutive* (k,ℓ)-co-clustering $(\mathcal{I}_R, \mathcal{J}_C)$ is defined as

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

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 [5]. The NP-hard Optimal Discretization problem [5] is the following: Given a set S of points in the plane, each 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 2 for an example). Here, a vertical (horizontal) line is a simple number denoting its x-(y-)coordinate.

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

Proof (Sketch). We give a polynomial-time reduction from Optimal Discretization. Let (S, k, ℓ) be an Optimal Discretization instance and let $X :=$



0	1	2	1	1	0	1	1
1	2	1	1	1	1	2	1
0	1	1	1	1	0	1	1
1	1	1	1	2	1	1	0
0	1	0	1	1	1	1	1

Fig. 2: Example instance of Optimal Discretization (left) and the corresponding instance of Consecutive Co-Clustering$_\infty$ (right). A solution to the Consecutive Co-Clustering$_\infty$ instance (shaded clusters) naturally translates into a consistent set of lines.

$\{x_1^*, \ldots, x_n^*\}$ 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 $(A, k+1, \ell+1, c)$ as follows: The matrix $A \in \{0,1,2\}^{m \times n}$ 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 0 if $(x,y) \in B$, 2 if $(x,y) \in W$, and otherwise 1. The cost is set to $c := 1$. Clearly, this instance can be constructed in polynomial time. We prove correctness in a full version of the paper. □

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 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 case of unbounded ℓ and $|\Sigma|$. In contrast to these hardness results, we now investigate which parameter combinations yield tractable cases. It turns out 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.

We start 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 Theorem 5 and Theorem 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

To start with, 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 A *satisfies* a cluster boundary $\mathcal{U}$ if $A_{rs} \subseteq [u_{rs}, u_{rs} + c]$ 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 $u_{rs} = \min A_{rs}$.

The following "subtask" of Co-Clustering$_\infty$ can be reduced to a certain CNF-SAT instance: Given a cluster boundary U and a Co-Clustering$_\infty$-instance I, find a co-clustering for I that satisfies $\mathcal{U}$. The 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* (A, k, ℓ, c) *and a cluster boundary* $\mathcal{U}$*, one can construct in polynomial time a* CNF-SAT*-instance* ϕ *with at most* $\max\{k, \ell\}$ *variables per clause such that* ϕ *is satisfiable if and only if there is a* (k, ℓ)*-co-clustering of* A *which satisfies* $\mathcal{U}$*.*

Proof. Given an instance (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 $\phi_{A,\mathcal{U}}$ 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 $u_{rs} \leq a_{ij} \leq u_{rs} + c$. Clearly, $\phi_{A,\mathcal{U}}$ is satisfiable if and only if there exists a (k, ℓ)-co-clustering of A satisfying the cluster boundary $\mathcal{U}$. Note that $\phi_{A,\mathcal{U}}$ consists of $O(km + \ell n)$ variables and $O(mnk\ell)$ clauses. □

4.2 Polynomial-Time Solvability

We first present a fairly simple 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 A has a $(1, \ell)$-co-clustering of cost c.

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

```
Input: $A \in \mathbb{R}^{m\times n}$, $\ell \geq 1$, $c \geq 0$
Output: A partition of $[n]$ into at most $\ell$ blocks yielding a cost of at most $c$, or
        no if no such partition exists.
1  for $j \leftarrow 1$ to $n$ do
2      $\alpha_j \leftarrow \min\{a_{ij} \mid 1 \leq i \leq m\}$;
3      $\beta_j \leftarrow \max\{a_{ij} \mid 1 \leq i \leq m\}$;
4  $\mathcal{N} \leftarrow [n]$;
5  for $s \leftarrow 1$ to $\ell$ do
6      Let $x_s \in \mathcal{N}$ be the index such that $\alpha_{x_s}$ is minimal;
7      $J_s \leftarrow \{j \in \mathcal{N} \mid \beta_j - \alpha_{x_s} \leq c\}$;
8      $\mathcal{N} \leftarrow \mathcal{N} \setminus J_s$;
9      if $\mathcal{N} = \emptyset$ then
10         return $(J_1, \dots, J_s)$;
11 return no;
```

If Algorithm 1 returns $(J_1, \dots, J_{\ell'})$ at line 10, 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 $\alpha_j \geq \alpha_{x_s}$ (by definition of x_s) and $\beta_j \leq \alpha_{x_s} + c$ (by definition of J_s). Thus, $A_{1s} \subseteq [\alpha_{x_s}, \alpha_{x_s} + c]$ holds for all $s \in [\ell']$, which yields $\operatorname{cost}(\{[m]\}, \{J_1, \dots, J_{\ell'}\}) \leq c$. Otherwise, if Algorithm 1 returns no at Line 11, then it has computed indices x_s, $s \in [\ell]$, and there exists at least one element $x_{\ell+1}$ in $\mathcal{N}$ when the algorithm terminates. Consider any $1 \leq s < s' \leq \ell + 1$. By construction, $x_{s'} \notin J_s$. Therefore, $\beta_{x_{s'}} > \alpha_{x_s} + c$ holds, and columns x_s and $x_{s'}$ contain elements with distance more than c. Thus, in any co-clustering with cost at most c, columns $x_1, \dots, x_{\ell+1}$ must be in different blocks, which is impossible by the pigeon-hole principle. Hence, this is indeed a no-instance.

The time complexity is seen as follows. The first loop examines all elements of the matrix in $O(mn)$ time. 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 $\phi_{A,\mathcal{U}}$. Observe that this formula is in 2-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 [2] (note that the size of the formula is in $O(mn)$). Overall, the input is a yes-instance if and only if $\phi_{A,\mathcal{U}}$ is satisfiable for some cluster boundary $\mathcal{U}$. ☐

Finally, we claim that it is possible to extend the above result to any number of column blocks for size-three alphabets (refer the full version for a proof).

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

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 A and cluster boundary $\mathcal{U}$, to simplify the boolean formula $\phi_{A,\mathcal{U}}$ 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* $O(|\Sigma|^{3\ell} n^2 m^2)$*-time solvable for* $c = 1$.

A subresult in the proof of Theorem 7 (deferred to a full version), 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 $(c+1)k\ell$. ☐

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 $|\Sigma|$ and ℓ. For $(2,*)$-Co-Clustering$_\infty$ with $c = 1$, both parameters are equivalent. 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 Conclusion

Contrasting previous theoretical work on approximation algorithms [1, 9], we started to closely investigate the time complexity of exactly solving the NP-hard Co-Clustering$_\infty$ problem, contributing a detailed view on its computational complexity landscape. Refer to Table 1 for an overview on most of our results. From a practical perspective, both our polynomial-time algorithms and our reduction to CNF-SAT solving—notably, exact solving approaches for Co-Clustering$_\mathcal{L}$ so far mostly rely on integer linear programming—may prove useful.

Several open questions derive from our work. Perhaps the most pressing open question is whether the most basic three-dimensional case—(2,2,2)-Co-Clustering$_\infty$ on three-dimensional input matrices—is polynomial-time solvable or NP-hard. Indeed, other than the techniques for deriving approximation algorithms [1, 9] our exact methods do not seem to generalize to higher dimensions.

Acknowledgment. We thank Stéphane Vialette for stimulating discussions.

Bibliography

[1] A. Anagnostopoulos, A. Dasgupta, and R. Kumar. A constant-factor approximation algorithm for co-clustering. *Theory of Computing*, 8:597–622, 2012.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inform. Process. Lett.*, 8(3):121–123, 1979.

[3] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *J. Mach. Learn. Res.*, 8:1919–1986, 2007.

[4] A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In *Proc. Eighth ISMB*, pages 75–85. AAAI, 2000.

[5] B. S. Chlebus and S. H. Nguyen. On finding optimal discretizations for two attributes. In *Proc. First RSCTC*, volume 1424 of *LNCS*, pages 537–544. Springer, 1998.

[6] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inform. Process. Lett.*, 12(3):133–137, 1981.

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

[8] J. A. Hartigan. Direct clustering of a data matrix. *J. Amer. Stat. Assoc.*, 67(337):123–129, 1972.

[9] S. Jegelka, S. Sra, and A. Banerjee. Approximation algorithms for tensor clustering. In *Proc. 20th ALT*, volume 5809 of *LNCS*, pages 368–383. Springer, 2009.

[10] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM TCBB*, 1(1):24–45, 2004.