Tel Aviv University
Raymond and Beverly Sackler
Faculty of Exact Sciences
School of Mathematical Sciences

The Bicluster Graph Editing Problem

by

Noga Amit

The research work has been conducted
under the supervision of
Prof. Ron Shamir

Submitted as partial fulfillment of the requirements
towards the M.Sc. degree

September 2004

Abstract

In this thesis we study the Bicluster Graph Editing Problem. The goal is to add/remove fewest edges from a bipartite graph so that it becomes a vertex disjoint union of complete bipartite graphs. The problem arises in analysis of gene expression data in molecular biology. We show that the problem is NP-hard and provide a polynomial algorithm that guarantees an approximation factor of 11.

Contents

1 Introduction **5**

1.1 Biological background 5

1.2 Clustering 9

1.3 Biclustering 13

1.3.1 Problem formulations 14

1.3.2 Biclustering Heuristics 17

1.4 Summary of thesis results 21

1.4.1 Two formulations of our problem 21

1.4.2 Our results 23

2 NP-Completeness Proof **25**

3 An Approximation Algorithm **33**

3.1 The Linear Program 33

3.2 The algorithm 35

3.3 The approximation factor 36

3.3.1 Type I 37

3.3.2 Type II 37

3.3.3 Type III 38

3.3.4 Type IV 42

Chapter 1

Introduction

In this chapter we first review the biological background that motivated the problem studied in this thesis. We then discuss the clustering problem, its formulations, principles, use and disadvantages. Next, we describe the biclustering problem. We discuss different formulations and review some known heuristics. Finally, we introduce the problem studied in this work, and summarize the thesis results.

1.1 Biological background

DNA, or deoxyribonucleic acid, is the hereditary material in humans and almost all other organisms. Nearly every cell in a persons body has the same DNA and most DNA is located in the cell nucleus. The information in DNA is stored as a code made up of four chemical bases or nucleotides: A, T, C and G. The order, or sequence, of these bases constitutes the information

available for building and maintaining an organism, similar to the way in which letters of the alphabet appear in a certain order to form words and sentences. The DNA molecule consists of two *strands*, or sequences. The bases in the two strands are paired, so that A in one strand matches T in the other, and C matches G. This way the sequence of one strand completely determines that of the other, *complement* strand. The two long strands form a spiral called a double helix.

An important property of DNA is that it can replicate, or make copies of itself. Each strand of DNA in the double helix can serve as a pattern for duplicating the sequence of bases. This is critical when cells divide because each new cell needs to have an exact copy of the DNA of the old cell.

Most genes contain the information needed to make functional molecules called proteins (A few genes produce other molecules that help the cell assemble proteins). The journey from gene to protein is complex and tightly controlled within each cell. It consists of two major steps: transcription and translation. Together, transcription and translation are known as g*ene expression*.

During the process of *transcription*, the information stored in a gene's DNA is transferred to a similar molecule called RNA (ribonucleic acid) in the cell nucleus. Both RNA and DNA are made up of a chain of nucleotide bases, but they have slightly different chemical properties. The type of RNA that contains the information for making a protein is called messenger RNA (mRNA) because it carries the information, or message, from the DNA out of the nucleus into the cytoplasm.

Translation, the second step in getting from a gene to a protein, takes place in the cytoplasm. The mRNA interacts with a specialized complex called a ribosome, which "reads" the sequence of mRNA bases. Each sequence of three bases, called a codon, usually codes for one particular amino acid. Amino acids are the building blocks of proteins.

The flow of information from DNA to RNA to proteins is one of the cornerstones of molecular biology. It is so important that it is sometimes called *the central dogma*. For much more information on this topic see [3, 16]

Under any particular condition, each cell expresses only a fraction of its genes, while the rest of the genes are silent. The process of turning genes on and off is known as g*ene regulation*. Gene regulation makes a brain cell look and act different from a liver cell or a muscle cell. Although we know that the regulation of genes is critical for life, this complex process is far from being fully understood.

Gene regulation can occur at any point during gene expression, but most commonly occurs at the level of transcription (when the information in a genes DNA is transferred to mRNA). Signals from the environment or from other cells activate proteins called *transcription factors*. These proteins bind to regulatory regions of a gene and increase or decrease the level of transcription. By controlling the level of transcription, this process can determine the amount of protein product that is made by a gene at any given time.

Today's high throughput DNA microarrays technology and other techniques enable us to measure the mRNA level of thousands of genes simultaneously [43, 13, 2, 27]. A microarray experiment typically assesses a large

number of gene expression levels under multiple conditions.

The gene expression data are usually displayed as a matrix, in which the rows correspond to different genes, and the columns to conditions. The conditions might be different individuals, different experimental conditions of the organism, or different tissues (e.g., cancerous vs healthy) from the same individual [11, 34]. The row vector of a gene is called the *expression pattern* of that gene. A column vector is called the *expression profile* of the condition.

The expression matrix is a powerful source of information for obtaining biological insights [1]. One goal of gene expression studies is to get better understanding of gene functions [45, 36, 35]. The idea is that genes with similar expression patterns are likely to be involved in similar processes, and hence have similar functionality.

Other than deducing function of unknown genes, gene expression analysis has proven to be helpful to identify diseases profiles [19, 30], deciphering regulatory mechanisms [22, 23, 49], genotyping [48] and drug developing [23].

The number of genes whose expression level is measured under one condition is thousands or tens of thousands. Since the expression matrix consists of a vast number of measurements, computational aids are needed, and the biological challenge should be formulated as a mathematical problem. Two very central methods in this analysis are clustering and biclustering, that will be briefly reviewed in the next sections.

1.2 Clustering

The concept of grouping data into clusters arises in numerous contexts and disciplines. This subject has been extensively studied and various exact algorithms, approximations and heuristics were proposed. For basic reviews and monographs on clustering see [42, 17, 28, 29, 37]. In the context of computational biology, clustering algorithms aim to partition the expression matrix into a collection of rows (or columns) subsets. The genes (or conditions) of each subset should share similar properties or biological function. Each such subset is called a *cluster*, and the collection of clusters is called a *clustering*, or a *clustering solution*. See Figure 1.1 for an example.

For concreteness suppose from now on that the matrix has rows corresponding to genes and columns corresponding to conditions, and assume that we wish to cluster the genes. Clustering of the conditions can be done analogously.

There can be many different formulations of the problem but they all share some basic demands. First of all, a measure of similarity or distance between expression patterns needs to be defined. Using this measure, one can use the original expression data to form a new matrix of the distances between every two genes.

Two main criteria need to be satisfied in a clustering: the first one is *homogeneity*, which means that genes in the same cluster should be relatively similar to each other. The second one is *separation*, which means that genes in different clusters should have low similarity to each other.

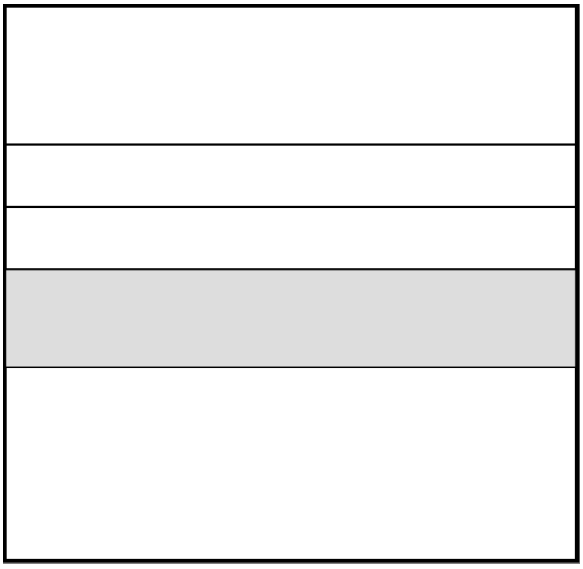

Figure 1.1: Clustering of a gene expression matrix. The rows of the matrix correspond to the genes, and the columns correspond to the conditions. A cluster is a partition of the rows. The features of each gene are all its column coordinates, and genes of the same cluster should have similar features (or, equivalently, similar row vectors). Since the clusters are disjoint, the rows of the matrix can be reordered so that each cluster is a contiguous strip.

Though the clustering problem can be formulated in various ways, most of the useful variations were proved to be NP-hard. We will describe here three formulations, though many more were proposed and might be relevant to different applications.

The Weighted Clustering Problem: *Given a complete graph with weights on the edges, find a partition of the graph nodes into sets called clusters, that maximizes the sum of weights of edges inside clusters, and minimizes the sum of edges weights between clusters.*

Note that there are two conflicting objectives here, which cannot be simultaneously optimized. One possible way around this is to fix the number of required clusters and maximize the sum of edge weights within clusters.

Minimum Disagreements Clustering (MinDisAgree): *Given a complete edge weighted graph where the edges weights are +1 and -1 only, find a partition of the vertices into clusters, that will minimize the number of errors: An error is a +1 edge connecting vertices in different clusters, or a -1 edge inside a cluster.*

Clearly, this problem is a special case of the weighted clustering problem. Bansal *et al.* [4] proved that the problem is NP-hard and gave a polynomial approximation that guarantees a (rather large) constant approximation factor. They also present a polynomial time approximation scheme for the maximization variant of the problem, in which the objective is to maximize the number of non-errors. Charikar *et al.* [6] presented an improved polynomial approximation algorithm that guarantees an approximation factor of four. This approximation algorithm will serve as the starting point to the approximation algorithm presented in this work.

An equivalent formulation originating from a graph modification viewpoint is the following. An *edit* operation in a graph is an addition or a deletion of an edge. A *cluster graph* is the disjoint union of cliques.

Cluster Graph Editing: *Given a graph G and an integer k, determine if one can make at most k edits in order to transform G into a cluster graph.*

This problem and other cluster modification problems were studied by Shamir *et al.* [38] and proved to be NP-hard.

Emanuel and Fiat [12] studied several correlation clustering problems. In particular, they studied the problem where "don't care" edges are present. Precisely, given a complete graph with weights $+1, -1$ or 0 on the edges, find a node partition that minimizes the number of disagreements. A disagreement is a $+1$ edge between clusters or a -1 edge inside a cluster. They show that the problem is NP-hard and give a polynomial-time approximation algorithm that guarantees an O(logn) approximation factor. The proof is based on showing equivalence to the multiway cut problem, and the approximation factor is the same is that for multiway cut [47].

Since most of the clustering problems are hard, most of the clustering algorithms are heuristics or approximations. In general, we can distinguish between two types of clustering methods; *agglomerative* methods, that start with small groups of genes and gradually join them to larger groups, and *divisive* methods that start the analysis with large groups and divide them into smaller clusters.

One clustering technique used for gene expression data analysis is **Hierarchic Clustering** [21, 24, 33], which attempts to place the input elements in a tree hierarchy structure, with the elements located in the leaves. Other methods include **K-means** clustering [25], **self organizing maps** [20, 44], and **CLICK** [40]. All these methods and others have become powerful tools of genetic research.

In spite of its tremendous utility, gene expression data analysis via clustering has several limitations. To start with, the traditional clustering formulations do not allow a gene to be a member of more than one group, while

in reality genes can have multiple biological functions. Moreover, clustering calls for grouping the genes according to their expression pattern across all conditions, while some conditions may be irrelevant to certain aims, especially when conditions are numerous and diverse. In other words, the requirement of finding a disjoint cover of the genes and of the expression patterns does not consistently reflect the biological motivation. A more flexible structure and formulation must be designed.

In order to try and address these shortcomings, the concept of *biclustering* was introduced to gene expression analysis. In the next section we will formulate the biclustering problem and discuss its application.

1.3 Biclustering

The concept of biclustering was first introduced in the seventies [17, 29], but its first usage in the context of computational biology was due to Cheng and Church [8]. Roughly speaking, biclustering calls for finding "significant" submatrices of the input matrix. A *bicluster* is a subset of rows and a subset of columns, defining together a submatrix that shows unique, similar expression patterns according to some scoring method. The difference between clustering and biclustering is illustrated in figures 1.1 and 1.2.

As in clustering, a good biclustering solution is one that satisfies the criterion of homogeneity: Genes inside a bicluster should have high similarity to each other. The second criterion of separation is not well defined, since genes can belong to more than one bicluster.

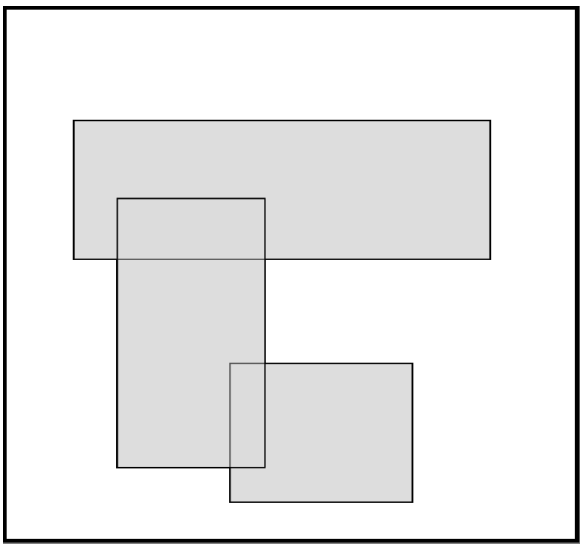

Figure 1.2: Biclustering of a gene expression matrix. Each bicluster is a subset of genes (rows) and a subset of conditions (columns). The reordering of rows and columns so that each bicluster forms a contiguous rectangle is usually impossible.

The biclustering problem also has many alternative formulations, as well as different applications and approaches. See Madeira and Oliveira [26], Liang *et al.* [7] and Tanay *et al.* [46] for recent reviews. We will describe below some of the possible mathematical formulations of biclustering problems, and present heuristics for finding biclusters.

1.3.1 Problem formulations

In this section, we will give a glimpse into the broad spectrum of possible formulations to the biclustering problem. These formulations differ in their objective function, their constraints, their complexity and the algorithms

used to solve them. Our terminology, as before, will be motivated by gene expression, although the problems and results have other applications as well.

We transform expression matrix into a bipartite graph in which the vertices in one part correspond to the genes and the vertices in the other part correspond to the conditions. Edges weights reflect expression levels. In some formulations a threshold on the (absolute) value of expression is used to obtain an unweighted graph.

A *biclique* is a complete bipartite subgraph. The biclustering problem aims to find large bicliques in the graph or, more generally, large dense subgraphs that are "close" to bicliques. In the weighted formulation, high weight subgraph are sought.

Several studies have dealt with the following biclustering problems:

Maximum Vertex Weight Biclique: *Given a vertex-weighted bipartite graph, find a biclique of maximum total vertex weight.*

Yannakakis presented a polynomial solution for this problem [50], and an alternative polynomial solution was later given by Hochbaum [18].

Exact Cardinality Biclique: *Given a bipartite graph and positive integers k and l, does there exist a biclique whose parts are of size k and l respectively?*

This problem was shown to be NP-complete by Dawande *et al.* [10]. A special case is the following:

Maximum Balanced Vertex Cardinality Biclique: *Given a bipartite graph $G = (U, V, E)$ and an integer k, determine if there exist subsets $A \subseteq$*

$U, B \subseteq V$ such that $|A| = |B| \geq k$ and $A \cup B$ induces a biclique.

The problem was shown to be NP-complete [14]. An equivalent problem was proved to be NP-complete by Dawande *et al.* [10].

The following problems are ones in which the objective function relates the edges.

Minimum Edge Deletion Biclique: *Given a bipartite graph $G = (U, V, E)$ and a positive integer k, does there exist a biclique (A, B, E') in G such that $|E \backslash E'| \leq k$?*

This problem is NP-complete, as proved by Dawande *et al.* [9]. Hochbaum [18] showed a polynomial approximation algorithm that guarantees an approximation factor of 2.

Maximum Edge Cardinality Biclique: *Given a bipartite graph G and a positive integer k, determine whether there exists a biclique (A, B, E') in G such that $|E'| \geq k$.*

Peeters [31] has shown that the problem is NP-hard.

Minimum Edge Deletion Weighted Biclique: *Given an edge-weighted bipartite graph $G = (U, V, E, w)$ and a positive integer k, does there exist a biclique (A, B, E') in G such that $\sum_{e \notin E'} w(e) \leq k$.*

Dawande *et al.* [9] proved that the problem is NP-complete. A polynomial approximation algorithm that guarantees an approximation factor of 2 was given by Hochbaum [18].

Maximum Edge Weighted Biclique: *Given an edge-weighted bipartite graph, find a biclique of maximum total edge weight.*

The hardness of this problem was proved by Dawande *et al.* [10].

Maximum ±1 Edge Weighted Biclique: *Given a complete bipartite graph with edge weights +1 and -1 only, find a biclique of maximum total edge weight.*

This formulation of the problem has strong relation to the problem studied in this work. The complexity of the problem is still open.

1.3.2 Biclustering Heuristics

The first application of biclustering in the context of gene expression data analysis was by Cheng and Church [8]. Their algorithm is greedy: it iteratively finds one bicluster on each step. This is done by finding a submatrix that has a low mean squared residue score. Roughly speaking, the residue score measures the variance across the genes in each column (see details below). After finding the best candidate bicluster, the algorithm "removes" it from the graph by replacing its submatrix entries by neutral values.

Precisely, given a data matrix E, a subset of genes I and a subset of conditions J, we define $e_{Ij} = \frac{\sum_{i \in I} e_{ij}}{|I|}$, $e_{iJ} = \frac{\sum_{j \in J} e_{ij}}{|J|}$ and $e_{IJ} = \frac{\sum_{i \in I, j \in J} e_{ij}}{|I||J|}$. The *residue score* of an element e_{ij} in the submatrix is $RS_{IJ}(i,j) = e_{ij} - e_{Ij} - e_{iJ} + e_{IJ}$, and the *mean squared residue score* of the entire submatrix is $\sum_{i \in I, j \in J} \frac{RS_{IJ}(i,j)^2}{|I||J|}$.

The goal of the algorithm is to find a bicluster of maximum size (which can be defined in several ways) among all biclusters with mean squared residue score not exceeding a threshold δ. After discovering such a bicluster, it is

removed from the matrix by modifying its entries, preventing the correlative signal in them to be beneficial for other biclusters. For applications of the Cheng and Church algorithm see [8].

Getz, Levine and Domany [15] define a generic scheme for transforming a one-dimensional clustering algorithm into a biclustering algorithm. Their **Coupled Two-way Clustering** algorithm (CTWC) relies on having one-dimensional clustering algorithm that can discover significant clusters. The CTWC algorithm recursively applies this algorithm to submatrices of the expression matrix, and at each step it selects a gene subset and a condition subset and applies the one-dimensional clustering algorithm twice. The submatrix created from pairing a significant subset of conditions and a significant subset of genes is called *stable*.

During its execution, CTWC dynamically maintains two lists of stable clusters (one for row clusters and one for column clusters) and a list of pairs of row and column subsets. Newly generated stable clusters are added to the row and column lists and a pointer that identifies the parent pair is recorded to indicate where this cluster came from. The iteration continues until no new clusters are found. The CTWC algorithm has been used in a variety of applications, e.g. [32].

Another biclustering algorithm is **SAMBA** [45] which stands for Statistical Algorithmic Method for Biclustering Analysis. The algorithm works as follows: It forms a bipartite graph according to the gene expression matrix, and calculates vertex pair weights according to a specific weighting scheme. This scheme is based on probabilistic modeling of the data. Then graph

theoretic techniques are used to derive scoring schemes for identifying significant subgraphs. Each subgraph corresponds to a bicluster, and the SAMBA algorithm aims to find the k "heaviest" biclusters.

More precisely, the expression matrix is transformed into a bipartite graph $G = (U, V, E)$, where U corresponds to the conditions and V corresponds to the genes, and $(u, v) \in E$ iff v responds to a condition u. The weight of a subgraph $H = (U', V', E')$ is the sum of its gene-condition pairs weights, edges or non-edges.

The weight of an edge (u, v) is $\log \frac{p_c}{p_{u,v}}$, where $p_{u,v}$ is the fraction of bipartite graphs with degree sequence identical to G that contains the edge (u, v), and p_c results from an alternative model that assumes that each edge in a true bicluster occurs with a constant probability. Similarly, the weight of every non-edge (u, v) is set to be $\log \frac{1-p_c}{1-p_{u,v}}$, and the weight of E is then:

$$\sum_{(u,v)\in E'} \log \frac{p_c}{p_{u,v}} + \sum_{(u,v)\in (U'\times V')\setminus E'} \log \frac{1-p_c}{1-p_{u,v}}$$

Under this scoring scheme, the weight of a subgraph is the log-likelihood ratio of that bicluster, so we need to discover the k heaviest subgraphs of G. Since this problem is NP-hard, SAMBA employs a heuristic search for such subgraphs. The implementation of the algorithm is part of the EXPANDER platform [39].

The **Plaid Model** is another statistically inspired modeling approach. Lazzeroni and Owen [22] represent the expression data as a matrix, which is a sum of "plaids" or layers: The matrix is approximated by a linear sum of signals, each having non-zero value outside its specified submatrix. The goal

is to find a small set of plaids, with a minimal difference between their sum and the observed signal.

The model uses the metaphor of colors to describe the superpositions of signal levels. It assumes the matrix entries values are a sum of a uniform background color μ_0 and k biclusters, each of them coloring a submatrix in a color θ_{ij0}. This color is composed of a background color of the bicluster, and row and column specific additive constants. Under this model, the expression matrix is represented as

$$A_{ij} = \mu_0 + \sum_{k=1}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}$$

where $\rho_{ik} = 1$ iff gene i belongs to bicluster k, and $\kappa_{jk} = 1$ iff the sample j belongs to bicluster k. The biclustering problem is formulated as finding parameter values so that the resulting matrix would fit the original data as much as possible. The objective is to minimize the function:

$$\sum_{ij}[A_{ij} - \sum_{k=0}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}]^2$$

where $\mu_0 = \theta_{ij0}$. The Plaid model algorithm was applied to yeast expression data [22].

Spectral Biclustering [19] uses techniques from linear algebra to identify bicluster structure in the input data. The algorithm assumes that, after normalization, the matrix contains a "checkerboard" structure.

Supposing the original matrix E has been normalized appropriately to form the matrix E'. The idea is to solve the eigenvalue problem $E'^T E' x = \lambda^2 x$ and examine the eigenvectors x. If the constants in an eigenvector can be

sorted to produce a step-like structure, the column clusters can be identified accordingly. The row clusters are found similarly from y satisfying $E'^T E'y = \lambda^2 y$.

More precisely, the checkerboard pattern in a matrix E is reflected in the constant structures of the pair of eigenvectors x and y that solved the coupled eigenvalue problem $E'^T E'x = \lambda^2 x$ and $E'^T E'y = \lambda^2 y$, where x and y have a common eigenvalue.

The algorithm depends critically on the normalization procedure used to transform the matrix, and Kluger *et al.* proposed three normalization methods. In a cancer context, the checkerboards structure reveals genes that are markedly up-regulated or down-regulated in patients with particular types of tumors. The Spectral Biclustering algorithm was applied in cancer research [19].

For other algorithmic approaches see [5, 49, 41, 30, 23].

1.4 Summary of thesis results

1.4.1 Two formulations of our problem

A ± 1 ***bipartite graph*** is a complete weighted bipartite graph $G = (U, V, E, w)$ where for each edge e, its weight $w(e)$ is $+1$ or -1. For convenience, we shall sometimes use $W = U \cup V$.

A ***biclustering*** of G is a partition of W into subsets $B = \{B_1, B_2 ... B_K\}$. Each B_i is called a ***bicluster***. If B_i consists of a single vertex then it is called

a ***singleton***. Given a biclustering B, we denote $U_i = B_i \cap U$ and $V_i = B_i \cap V$.

A ***negative error*** in a biclustering is a negative edge inside a bicluster, i.e., one whose end vertices reside in the same bicluster. A ***positive error*** is a positive edge between two biclusters. An ***error*** is a positive or a negative error.

The ± 1 Biclustering Problem: *Given a ± 1 bipartite graph, find a biclustering of the graph with a minimum number of errors.*

As seen in figure 1.3, the formulation we have chosen implies finding a collection of submatrices so that: (1) Each two submatrices have disjoint row and column sets.(2) The number of -1 values in these submatrices, plus the number of $+1$ outside them, is minimum. A singleton bicluster corresponds to a row or a column that does not intersect with any submatrix.

An equivalent formulation of the problem is motivated by graph modifications. In an unweighted bipartite graph, an ideal bicluster is a complete bipartite subgraph, and any missing edge inside the subgraph is considered an error. Similarly, edges that do not belong to any bicluster are errors. This gives rise to the following formulation:

A bipartite graph $G = (U, V, E)$ is called a **bicluster graph** if every connected component of G is a biclique (a complete bipartite subgraph). Define $E \triangle F = (E \setminus F) \cup (F \setminus E)$. If G is any bipartite graph and $F \subset U \times V$ is such that $G' = (U, V, E \triangle F)$ is a bicluster graph, then F is called a **bicluster editing set** for G.

Bicluster Graph Editing: *Given a bipartite graph G and an integer k, determine if G has a bicluster editing set of size at most k.*

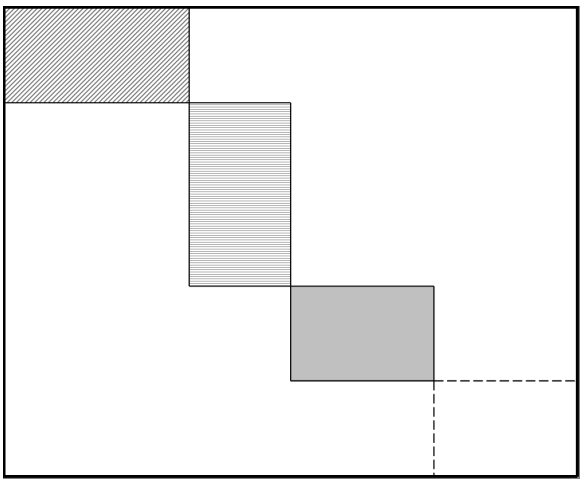

Figure 1.3: The ±1 Biclustering Problem. The goal is to find a collection of submatrices that have disjoint row and column sets with a minimum number of errors. The dashed area corresponds to singletons, i.e., rows and columns that do not take part in any bicluster submatrix.

It is easy to notice that the problem is equivalent to the ±1 Biclustering Problem. Given a ±1 bipartite graph $G = (U, V, E, w)$ for the ±1 Biclustering Problem, the corresponding input for the Bicluster Graph Editing is $G' = (U, V, E')$ where $e \in E' \Leftrightarrow w(e) = +1$. In the opposite direction, we change every edge to a positive edge and every non-edge to a negative edge. A minimum bicluster editing set corresponds to a minimum number of errors.

1.4.2 Our results

We first prove the NP-completeness of the problem. Hardness is proved by reducing the problem from the 3-Exact 3-Cover (3X3C) problem. The reduction is inspired by the work of Shamir, Sharan and Tsur [38], in proving

the hardness of the Clustering Editing Problem.

Next, we give a polynomial approximation algorithm for the problem. The algorithm is based on the algorithm of Charikar *et al.* to the MinDisAgree clustering problem [6]. The algorithm uses linear programming, relaxation and rounding, and guarantees an approximation factor of 11.

Our formulation of the biclustering problem can be transformed to the problem solved by Emanuel and Fiat [12]. This is done by constructing a complete graph by adding to the original bipartite graph all the missing edges, assigning a zero weight to them. By the results of Emanuel and Fiat [12], this implies that the problem has a polynomial-time approximation algorithm that guarantees an O(logn) approximation factor [1]. This result is improved here. The NP-hardness result that we obtain is stronger too.

[1] We thank Amos Fiat for this observation.

Chapter 2

NP-Completeness Proof

In this chapter we shall prove that Bicluster Graph Editing is NP-Complete. Our proof is inspired by the NP-hardness proof of [38] for Cluster Graph Editing.

Theorem: *Bicluster Graph Editing is NP-complete.*

Membership in NP is trivial. We prove NP-completeness by reduction from the following restriction of Exact Cover by 3-sets:

The 3-Exact 3-Cover (3X3C): *Given a collection C of triplets of elements from a set $\mathcal{U} = \{1, 2, ..., 3n\}$, such that each element of $\mathcal{U}$ is a member of at most 3 triplets, determine if there exist a sub-collection $I \subseteq C$ of size n that covers $\mathcal{U}$.*

This problem is known to be NP-complete [14].

The reduction: Let $m \equiv 36n$. Given an instance $< C, \mathcal{U} >$ of the 3X3C problem we build a bipartite graph $G = (U, V, E)$ as follows:

$$
\begin{aligned}
U &= U_1 \cup U_2 \\
V &= V_1 \cup V_2 \\
U_1 &= \{u_1, ...u_{3n}\} \\
V_1 &= \{v_1, ...v_{3n}\} \\
U_2 &= \bigcup_{S \in C} \{u_1(S), ..., u_m(S)\} \\
V_2 &= \bigcup_{S \in C} \{v_1(S), ..., v_m(S)\} \\
E &= \bigcup_{i=1}^{5} E_i \\
E_1 &= \{(u_i, v_i) : i = 1...3n\} \\
E_2 &= \{(u_i, v_j) : \exists S \in C : (i, j) \in S\} \\
E_3 &= \{(u_i(S), v_j(S)) : S \in C, i, j = 1...m\} \\
E_4 &= \{(u_i, v_j(S)) : S \in C, i \in S, j = 1...m\} \\
E_5 &= \{(v_i, u_j(S)) : S \in C, i \in S, j = 1...m\}
\end{aligned}
$$

In words: for every element of $\mathcal{U}$ we construct two connected vertices, one in each part of G. For each triplet in C we connect the corresponding six vertices into a biclique. We also construct a balanced biclique of $2m$ vertices for every triplet and connect the vertices of the biclique to the three original vertices corresponding to that triplet in the other side. The reduction is partially illustrated in Figure 2.1

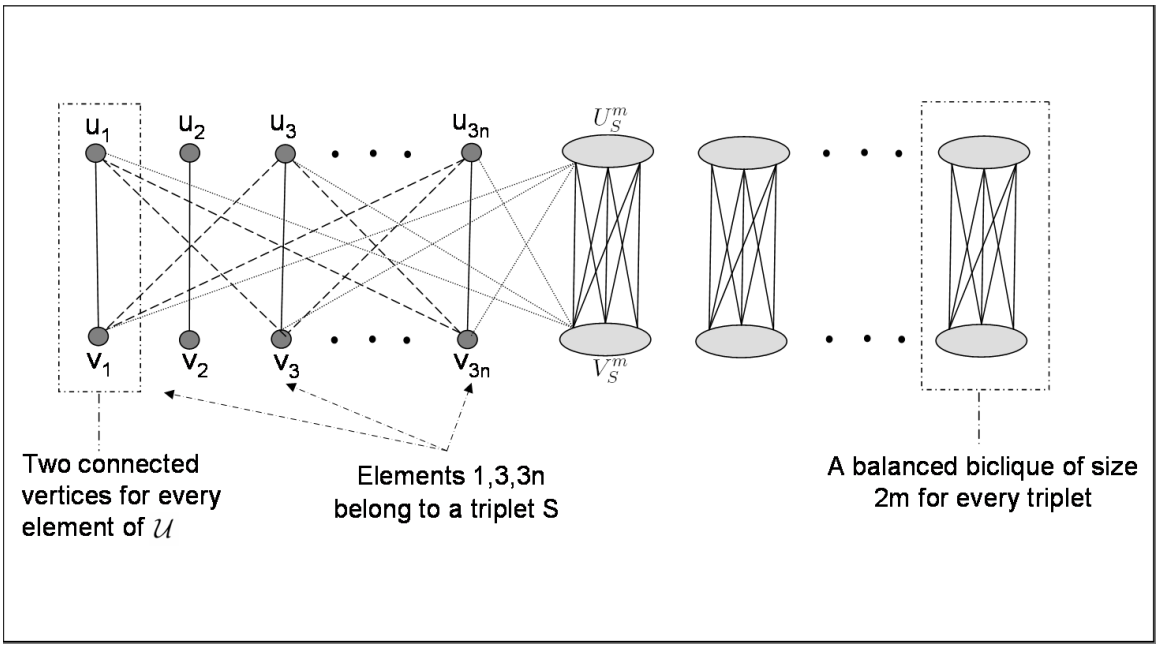

Figure 2.1: A sketch of the graph built by the reduction.

For every triplet $S \in C$ we denote:

$$\begin{aligned} U_S &= \{u_i, u_j, u_k : i, j, k \in S\} \\ V_S &= \{v_i, v_j, v_k : i, j, k \in S\} \\ W_S &= U_S \cup V_S \\ U_S^m &= \{u_1(S), ..., u_m(S)\} \\ V_S^m &= \{v_1(S), ..., v_m(S)\} \\ W_S^m &= U_S^m \cup V_S^m \end{aligned}$$

The reduction is clearly polynomial. Let $M \equiv 2m(3|C| - 3n)$ and $N \equiv |E_2| - 6n$. We prove that there is an exact cover of $\mathcal{U}$ if and only if there is a bicluster editing set for G of size $M + N$.

Proof:

$\Rightarrow$ Let $I \subseteq C$ be an exact cover of $\mathcal{U}$. Let $F_1 = \{(u_i, v_j(S)) : S \notin I, u_i \in S, j = 1...m\}$. Similarly, let $F_2 = \{(v_i, u_j(S)) : S \notin I, v_i \in S, j = 1...m\}$. Also, let $F_3 = \{(v_i, u_j) : S \notin I, i, j \in S\}$ and let $F = F_1 \cup F_2 \cup F_3$. It is easy to verify that $|F_1| = |F_2| = m(3|C| - 3n)$ and that $|F_3| = |E_2| - 6n$. Hence, $|F| = M + N$. Since I is an exact cover, if follows that $G \backslash F$ is a bicluster graph.

$\Leftarrow$ Let F' be a bicluster editing set for G with $|F'| \leq M + N$. Let F be an optimum bicluster editing set for G for which $|F| \leq |F'| \leq M + N$. We prove that $|F| = M + N$ and that we can derive from F an exact cover of $\mathcal{U}$. This implies $|F'| = |F|$, and therefore F' is an optimum bicluster editing set.

We know that each element of $\mathcal{U}$ occurs in at most 3 triplets and therefore $3|C| \leq 9n$. Hence $M \leq 2m(9n - 3n) = 12mn$. In addition, in the graph (U, V, E_2) each vertex has at most 6 neighbors (being a member of at most 3 triplets) which implies $|E_2| \leq 3n \cdot 6 = 18n$ and $N \leq 18n - 6n = 12n$. Hence:

$$|F| \leq 12mn + 12n = \frac{m^2}{3} + \frac{m}{3} = m(\frac{m+1}{3}) \tag{2.1}$$

Let $G' = (U, V, E \triangle F)$ be the bicluster graph obtained by editing G according to F. We shall prove that for every subset $S \in C$ there exists a unique biclique in G' that contains W_S^m.

We first show is that there exists a biclique B in G' such that $|B \cap W_S^m| \geq \frac{3m}{2} + 3$. Suppose that the vertices of W_S^m are partitioned among k bicliques

$B_1, ..., B_k$ in G'. Assume by contradiction that for every i, $|W_S^m \cap B_i| \leq \frac{3m}{2}+2$.

Let $x_i = |U_S^m \cap B_i|$ and $y_i = |V_S^m \cap B_i|$. So we suppose that $x_i+y_i \leq \frac{3m}{2}+2$ for every i.

Missing edges between $W_S^m \cap B_i$ and $W_S^m \setminus B_i$ are:

$$x_i(m-y_i) + y_i(m-x_i) = (x_i+y_i)m - 2x_iy_i$$

This expression is minimizes (for $x_i + y_i$ fixed) when $x_i = y_i$ and is at least:

$$(x_i+y_i)(m - \frac{x_i+y_i}{2})$$

Therefore:

$$\begin{aligned} |F| &\geq \sum_{i=1}^{k}(x_i+y_i)(m-\frac{x_i+y_i}{2}) \\ &\geq \sum_{i=1}^{k}(x_i+y_i)(m-\frac{\frac{3m}{2}+2}{2}) \\ &= 2m(\frac{m-4}{4}) \end{aligned}$$

From (2.1) we know that $|F| \leq m(\frac{m+1}{3})$. Since $m > 14$, a contradiction follows.

For a triplet S, let B_S be the biclique B_i for which $x_i + y_i$ is maximized. We next prove that $B_S \subseteq W_S^m \cup W_S$. Assume by contradiction that there exists a vertex x in $B_S \setminus (W_S^m \cup W_S)$ and w.l.o.g. assume that $x \in U$. Consider a new partition obtained from the original partition by moving x

outside B_S, and turning it into a singleton. Let $\tilde{F}$ be the new editing set and let $\tilde{G} = (U, V, E \triangle \tilde{F})$.

We have shown that $|B_S \cap W_S^m| \geq \frac{3m}{2} + 3$ which implies that $|B_S \cap V_S^m| \geq \frac{m}{2} + 3$. Since there are no edges between x and $B_S \cap V_S^m$ in G, these edges belong to F but not to $\tilde{F}$. The new edges that might have been added to $\tilde{F}$ due to the new partition are edges between x and at most $\frac{m}{2} - 3$ edges between x and $B_S \setminus (W_S^m)$. Also, the edges between x and at most 3 vertices of V_S might be in $\tilde{F}$ and not in F (if these vertices belong to B_S). So we get that:

$$|F| - |\tilde{F}| \geq (\frac{m}{2} + 3) - (\frac{m}{2} - 3 + 3) = 3$$

Since F is an optimum editing set we get a contradiction. Hence, $B_S \subseteq W_S^m \cup W_S$.

Next we show that $W_S^m \subseteq B_S$. Assume by contradiction that there exists a vertex y in $W_S^m \setminus B_S$ and w.l.o.g. assume that $y \in U$. Consider a new partition obtained from the original partition by moving y into B_S. Again, let $\tilde{F}$ be the new editing set and let $\tilde{G} = (U, V, E \triangle \tilde{F})$.

As previously, note that $|B_S \cap V_S^m| \geq \frac{m}{2} + 3$. All the edges between y and $B_S \cap V_S^m$ belong to F but not to $\tilde{F}$. The new edges that might have been added to $\tilde{F}$ due to the new partition are edges between y and at most $\frac{m}{2} - 3$ vertices of $V_S^m \setminus B_S$. Also, the edges between y and at most 3 vertices of V_S might be in $\tilde{F}$ (if these vertices don't belong to B_S). We get that:

$$|F| - |\tilde{F}| \geq (\frac{m}{2} + 3) - (\frac{m}{2} - 3 + 3) = 3$$

Again, we get a contradiction to the optimality of F. In summary, we

have shown that $W_S^m \subseteq B_S \subseteq W_S^m \cup W_S$.

To complete the proof we will show that $|F| \geq M + N$ with equality if and only if for every $S \in C$ either $B_S = W_S^m$ or $B_S = W_S^m \cup W_S$. We denote $F' = F \cap (E_4 \cup E_5)$ and $F'' = F \cap (E_1 \cup E_2)$. (Note that since we have shown that each W_S^m is contained in some bicluster, it follows that $E_3 \cap F = \emptyset$). Hence, $F \supseteq F' \cup F''$.

Examine an element $i \in \mathcal{U}$ that is a member of (at least) two subsets $S_1, S_2 \in C$. By the previous claim, $W_{S_1}^m$ and $W_{S_2}^m$ are subsets of distinct bicliques in G'. Hence either the edges between u_i and $W_{S_1}^m$ belong to F or the edges between u_i and $W_{S_2}^m$ belong to F (or both). If each u_i remains connected to precisely one W_S^m, the total contribution to F of edges incident on u_i is $m(3|C| - 3n)$ and this is the minimum possible. The same argument applies to the edges incident on v_i. Therefore $|F'| \geq M$, with equality iff each u_i is connected in G' to exactly one V_S^m and each v_i is connected in G' to exactly one U_S^m.

Since each $B_S \subseteq W_S^m \cup W_S$, it follows that in G', every u_i is a neighbor of at most three vertices from V_1. Since $|V_1| = 3n$, there are at most $9n$ edges between U_1 and V_1 in G'. Therefore:

$$|F''| \geq |(E_1 \cup E_2)| - 9n = |E_1| - 6n = N$$

Combining the previous results we get that $|F''| \geq N$ and $|F'| \geq M$. Since $M + N \leq |F'| + |F''| \leq |F| \leq M + N$, we must have $|F| = M + N$, and, more importantly, for every $S \in C$ either $B_S = W_S^m$ or $B_S = W_S^m \cup W_S$. The set $\{S \in C | B_S = W_S^m \cup W_S\}$ induces an exact cover of $\mathcal{U}$. □

Note that in both directions of the proof only edge deletions are performed. Hence the same reduction also proves the NP-hardness of the following problem:

The Bicluster Graph Deletion Problem: Given a bipartite graph $G = (U, V, E)$, find a set $F \subseteq E$ of minimum cardinality such that $(U, V, E \setminus F)$ is a bicluster graph.

Corollary: *Bicluster Graph Deleting is NP-complete.* □

Chapter 3

An Approximation Algorithm

In this chapter we will present an approximation algorithm for the ± 1 biclustering problem. The algorithm is based on the algorithm to the MinDisAgree clustering problem [6]. The algorithm uses linear programming, relaxation and rounding, and guarantees an approximation factor of 11.

3.1 The Linear Program

An equivalent formulation of the problem is as follows: Assign a binary variable x_{ij} to every edge ij, such that $x_{ij} = 0$ iff i and j are both in the same bicluster. Define the following integer programming problem (IP).

$$
\begin{array}{lll}
\text{Minimize} & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) & \\
\text{Such that} & x_{ij} \leq x_{il} + x_{kj} + x_{kl} & \text{for all } i, k \in U \text{ and } j, l \in V \\
& x_{ij} \in \{0, 1\} & \text{for all } i \in U \text{ and } j \in V
\end{array}
$$

Here and throughout $+(ij) = \{ij | w(ij) = +1\}$, and $-(ij) = \{ij | w(ij) = -1\}$.

The objective function measures the number of errors. This includes *positive errors*, i.e., positive edges between biclusters (positive edges ij for which $x_{ij} = 1$), as well as *negative errors*, i.e., negative edges inside biclusters (negative edges ij for which $x_{ij} = 0$).

It is easy to see that the inequalities $x_{ij} \leq x_{il} + x_{kj} + x_{kl}$ guarantees that if i and l are in the same bicluster, k and l are in the same bicluster, and so are k and j, then i and j must be in the same bicluster as well.

Relaxation is obtained by replacing the integer constraints of $x_{ij} \in \{0, 1\}$ with linear programming (LP) constraints $0 \leq x_{ij} \leq 1$. Under this LP formulation, we refer to x_{ij} as the *distance* between i and j. Intuitively, points (nodes) that are close should be placed in the same bicluster and points that are far should be placed in different biclusters.

We present a polynomial algorithm, that guarantees an approximation factor of 11. We shall show that the algorithm solution is at most 11 times the LP solution. Since the LP solution is not larger than the IP solution, the result will follow.

3.2 The algorithm

Let $S = V \cup U$. Repeat the following steps:

1. Pick an edge uv with a distance smaller than $\frac{1}{11}$.
   Let N_u and N_v be the set of vertices within a distance of at most $\frac{5}{11}$ from u and v respectively (not including u and v themselves).
   Similarly, let N'_u and N'_v be the set of vertices within a distance of at most $\frac{3}{11}$ from u and v respectively (Again, not including u and v).
   In addition, denote by α_u (α_v) the average distance of the vertices in N_u from u (N_v from v).
   If $N_u = \emptyset$ then define $\alpha_u = 1$

2. Let
$$B = \begin{cases} \{u,v\} & ; \text{if } \alpha_u, \alpha_v > \frac{3}{11} \\ & \text{or if } \frac{1}{11} < \alpha_u \leq \frac{3}{11} \text{ and } \alpha_v > \frac{3}{11} \\ & \text{or if } \alpha_u > \frac{3}{11} \text{ and } \frac{1}{11} < \alpha_v \leq \frac{3}{11} \\ \{u,v\} \cup N_u \cup N_v & ; \text{if } \alpha_u, \alpha_v \leq \frac{3}{11} \\ \{u,v\} \cup N_u \cup N'_v & ; \text{if } \alpha_u \leq \frac{1}{11} \text{ and } \alpha_v > \frac{3}{11} \\ \{u,v\} \cup N'_u \cup N_v & ; \text{if } \alpha_u > \frac{3}{11} \text{ and } \alpha_v \leq \frac{1}{11} \end{cases}$$
   Output B as a bicluster, let $S = S \backslash B$, and return to step 1.

3. When no edges with distance smaller than $\frac{1}{11}$ are left, output all the vertices of S as singletons.

For proving the approximation we shall use an ordering of the vertices at each iteration of the loop, so that $i < j$ if the distance of i from u or v is

smaller than the distance of j from u or v. This ordering is applied to both sides together. We call the algorithm above Algorithm A.

3.3 The approximation factor

In order to calculate the cost (number of errors) of the algorithm and compare it to the LP cost, we consider one bicluster at the time, in the order generated by the algorithm (singletons can be arbitrary ordered) and count the errors in the edges associated with these biclusters. The set of edges associated with bicluster B consists of the edges inside the bicluster, i.e., $in(B) = \{ij | i, j \in B\}$, and the set of edges connecting B to $S \backslash B$, i.e., $out(B) = \{ij | i \in B, j \in S \backslash B \text{ or } j \in B, i \in S \backslash B\}$. Note that S is a subset of $U \cup V$ whose size depends on previous iterations.

Denote by a superscript $+/-$ the subset of edges with positive/negative weight, e.g., $in^-(B) = \{ij \in in(B) | w(ij) = -1\}$. The cost of the algorithm associated with B is:

$$A(B) = |in^-(B)| + |out^+(B)|$$

The LP cost associated with B is:

$$LP(B) = \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij})$$

We shall show that for every B, $A(B) \leq 11 \cdot LP(B)$, which will establish the approximation factor. We analyze separately each of the four types of biclusters generated by the algorithm:

1. Type I: singletons (generated in step 3).

2. Type II: single edge biclusters.

3. Type III: biclusters of the form $\{u, v\} \cup N_u \cup N_v$.

4. Type VI: biclusters of the form $\{u, v\} \cup N_u \cup N'_v$ or $\{u, v\} \cup N'_u \cup N_v$.

3.3.1 Type I

The edges associated with a singleton bicluster $B = \{x\}$, are simply all edges adjacent to x in the current graph, and the positive ones are the errors of A. Since a singleton bicluster is constructed when no edges with a distance smaller than $\frac{1}{11}$ are left in S, the LP cost associated to B will be at least $\frac{1}{11}$ times the number of positive edges. Hence $A(B) \leq 11 \cdot LP(B)$.

3.3.2 Type II

Let B be a bicluster of the form of $\{u, v\}$. There are two kinds of errors in this case. The first possible error is when uv is negative, but in this case the LP cost associated with it will be 1-x_{uv} , and since $x_{uv} < \frac{1}{11}$ the LP cost will be at least $\frac{10}{11}$. Since the algorithm A pays 1 for this error, the approximation factor holds.

The second kind of errors are $out^+(B)$, i.e., positive edges between B and $S \backslash B$. We first consider edges connecting u to N_u and v to N_v. For type II biclusters both $\alpha_u, \alpha_v > \frac{1}{11}$.Therefore, we know that:

$$\sum_{i \in N_v} x_{iv} + \sum_{j \in N_u} x_{uj} \geq$$
$$\geq \frac{1}{11}|N_v| + \frac{1}{11}|N_u| =$$
$$= \frac{1}{11}(|N_v| + |N_u|)$$

.

Since for $i \in N_v$, $1 - x_{iv} \geq x_{iv}$, and for $j \in N_u$, $1 - x_{uj} \geq x_{uj}$, it follows that the LP cost of *all* edges from u to N_u and from v to N_v is at least $\frac{1}{11}(|N_v| + |N_u|)$. Since the number of positive errors is at most $|N_v| + |N_u|$, this cost is at most 11 times the LP cost for these edges.

The remaining edges connect vertices outside N_u and N_v to u or v. Each positive edge has a distance greater than $\frac{5}{11}$ and so its error cost in A is at most $\frac{11}{5}$ times the LP cost.

We have thus shown $A(B) \leq 11 \cdot LP(B)$ for biclusters of type II.

3.3.3 Type III

Let B be a bicluster of the form $\{u, v\} \cup N_u \cup N_v$, which was constructed when both α_u and α_v were at most $\frac{3}{11}$. As in the previous sections, we will separate the discussion to negative errors and positive errors. We will need the following observations, which follow from the first type of constraints of the linear programming (the quadrangle inequality):

Observation 1. *The following is a lower bound for the LP cost of the positive*

edge ij:

$$x_{ij} \geq x_{uj} - x_{uv} - x_{iv}$$

Observation 2. *The following is a lower bound for the LP cost of the negative edge ij:*

$$1 - x_{ij} \geq 1 - x_{uv} - x_{uj} - x_{iv}$$

Negative edge errors

We first calculate the cost for edges $ij \in in(B)$. If x_{uj} and x_{iv} are both smaller than $\frac{9}{22}$, then by Observation 2, we know that the LP cost of ij is at least $1 - \frac{1}{11} - \frac{9}{22} - \frac{9}{22} = \frac{1}{11}$, which satisfies the required factor.

Recall that $i < j$ iff $x_{iv} < x_{uj}$. Each remaining negative edge ij will be charged to the vertex with the greater distance from u or v, i.e., ij will be charged to j iff $i < j$ (with ties broken arbitrarily). W.l.o.g. we will assume that this vertex is $j \in V$, and therefore we know that x_{uj} lies in the range $(\frac{9}{22}, \frac{5}{11}]$. The total LP cost of edges in $in^{-}(B)$ associated with j (using both observations) is at least:

$$\Theta(j) = \sum_{i<j;+(ij)} (x_{uj} - x_{uv} - x_{iv}) + \sum_{i<j;-(ij)} (1 - x_{uv} - x_{uj} - x_{iv})$$

Denote by p_j the number of positive edges ij inside B, for which i is less than this j. Similarly n_j stands for the number of such negative edges. The sum is then:

$$\Theta(j) = p_j(x_{uj} - x_{uv}) + n_j(1 - x_{uv} - x_{uj}) - \sum_{i;i<j} x_{iv}$$

The edge uv was chosen such that $x_{uv} < \frac{1}{11}$. In addition, we know that $\sum_{i \in N_v} x_{iv} \leq \frac{3}{11}|N_v|$. Now:

$$\sum_{i \in N_v} x_{iv} = \sum_{i \in N_v, i<j} x_{iv} + \sum_{i \in N_v, i>j} x_{iv}$$

Since all the items in the second sum are greater than $\frac{9}{22}$, we can be sure that:

$$\sum_{i \in N_v, i<j} x_{iv} \leq \frac{3}{11}(p_j + n_j)$$

Therefore:

$$\begin{aligned} \Theta(j) &\geq p_j(x_{uj} - \frac{1}{11}) + n_j(\frac{10}{11} - x_{uj}) - \frac{3}{11}(p_j + n_j) = \\ &= p_j(x_{uj} - \frac{4}{11}) + n_j(\frac{7}{11} - x_{uj}) \end{aligned} \tag{3.1}$$

The LP cost is bounded below by the linear function (3.1) that lies between $\frac{1}{22}p_j + \frac{5}{22}n_j$ (when $x_{uj} = \frac{9}{22}$) and $\frac{1}{11}p_j + \frac{2}{11}n_j$ when ($x_{uj} = \frac{5}{11}$). In particular the LP cost is at least $\frac{2}{11}n_j$. Since the number of errors is n_j, the algorithm cost on these edges is at most $\frac{11}{2}$ times the LP cost.

Positive edge errors

W.l.o.g. we will consider the positive edge ij such that $x_{iv} \leq \frac{5}{11}$ and $x_{uj} > \frac{5}{11}$. If $x_{uj} \geq \frac{7}{11}$ then by Observation 1 the LP cost of ij is at least $\frac{7}{11} - \frac{1}{11} - \frac{5}{11} = \frac{1}{11}$.

Each remaining positive edge ij will be charged to a vertex outside B. W.l.o.g. we will assume that this vertex is $j \in V$, and therefore we know that x_{uj} lies in the range $(\frac{5}{11}, \frac{7}{11})$. The total LP cost of edges within B charged to j (using both observations) is at least:

$$\Theta(j) \geq \sum_{i \in N_v; +(ij)} (x_{uj} - x_{uv} - x_{iv}) + \sum_{i \in N_v; -(ij)} (1 - x_{uv} - x_{uj} - x_{iv})$$

As before, denote by p_j the number of positive edges ij for which $i \in N_v$. Similarly n_j stands for the number of such negative edges. The sum is then:

$$\Theta(j) \geq p_j(x_{uj} - x_{uv}) + n_j(1 - x_{uv} - x_{uj}) - \sum_{i \in N_v} x_{iv}$$

Again, since $x_{uv} < \frac{1}{11}$ and $\sum_{i \in N_j} x_{iv} \leq \frac{3}{11}|N_v|$, we can derive the same function as (3.1):

$$\begin{aligned} \Theta(j) &\geq p_j(x_{uj} - \frac{1}{11}) + n_j(\frac{10}{11} - x_{uj}) - \frac{3}{11}(p_j + n_j) \geq \\ &\geq p_j(x_{uj} - \frac{4}{11}) + n_j(\frac{7}{11} - x_{uj}) \end{aligned} \tag{3.2}$$

The LP cost is bounded below by the linear function (3.2) that lies between $\frac{1}{11}p_j + \frac{2}{11}n_j$ (when $x_{uj} = \frac{5}{11}$) and $\frac{3}{11}p_j$ (when $x_{uj} = \frac{7}{11}$). In particular, the LP cost is at least $\frac{1}{11}p_j$. Since the number of errors performed by the algorithm A is p_j we get the cost ratio 11 again.

3.3.4 Type IV

W.l.o.g. let B be a bicluster of the form $\{u, v\} \cup N_u \cup N'_v$, that was constructed when $\alpha_u \leq \frac{1}{11}$ and $\alpha_v > \frac{3}{11}$. Again, we will separate the discussion to negative edges inside B and positive edges between B and $S \backslash B$.

Negative edge errors

Negative errors are negative edges between N_u and N'_v. In this case, by Observation 2, we know that the LP cost of each negative edge is at least $1 - \frac{1}{11} - \frac{5}{11} - \frac{3}{11} = \frac{2}{11}$ and hence we get the required approximation factor.

Positive edge errors

There are two types of positive errors. The first is positive edges between $(V \cap S) \backslash N_u$ and N'_v. We will consider the positive edge ij such that $i \in N'_v$ and j is outside N_u. From that we know that $x_{iv} \leq \frac{3}{11}$ and $x_{uj} > \frac{5}{11}$. Hence, by Observation 1, the LP cost of ij is at least $\frac{5}{11} - \frac{1}{11} - \frac{3}{11} = \frac{1}{11}$.

The second type are positive edges between $(U \cap S) \backslash N'_v$ and N_u. Similarly to the proof in 3.3.3, if $x_{iv} \geq \frac{7}{11}$ then the LP cost of a positive edge ij is at least $\frac{7}{11} - \frac{1}{11} - \frac{5}{11} = \frac{1}{11}$.

Each remaining positive edge ij will be charged to vertex $i \in U$ outside N'_v. x_{iv} lies in the range $(\frac{3}{11}, \frac{7}{11})$. The total LP cost of edges charged to i is at least:

$$\rho(i) = \sum_{j \in N_u; +(ij)} (x_{iv} - x_{uv} - x_{uj}) + \sum_{j \in N_u; -(ij)} (1 - x_{uv} - x_{iv} - x_{uj})$$

Again, denote by p_i the number of positive edges ij for which $j \in N_u$. Similarly n_i stands for the number of such negative edges. The sum is then:

$$\rho(i) = p_i(x_{iv} - x_{uv}) + n_i(1 - x_{uv} - x_{iv}) - \sum_{j \in N_u} x_{uj}$$

From the way B was constructed we know that $x_{uv} < \frac{1}{11}$ and $\sum_{j \in N_u} x_{uj} \leq \frac{1}{11}|N_u| = \frac{1}{11}(p_i + n_i)$. Hence:

$$\rho(i) \geq p_i(x_{iv} - \frac{1}{11}) + n_i(\frac{10}{11} - x_{iv}) - \frac{1}{11}(p_i + n_i) = p_i(x_{iv} - \frac{2}{11}) + n_i(\frac{9}{11} - x_{iv}) \tag{3.3}$$

This time, the linear function (3.3) lies between $\frac{1}{11}p_i + \frac{6}{11}n_i$ (when $x_{iv} = \frac{3}{11}$) and $\frac{5}{11}p_i + \frac{2}{11}n_i$ (when $x_{iv} = \frac{7}{11}$). The LP cost is thus at least $\frac{1}{11}p_i$. Since the number of errors made by the algorithm A is p_j we get the factor 11.

□

We are now ready to state the main result of this chapter:

Theorem: *Algorithm A is a polynomial-time approximation algorithm for the Bicluster Graph Editing Problem which guarantees an approximation factor of 11.*

Bibliography

[1] The chipping forecast II. Special supplement to Nature Genetics Vol 32, 2002.

[2] A. Abbott. DNA chips intensify the sequence search. *Nature*, 379:392, 1996.

[3] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J.D. Watson. *Molecular Biology of the Cell*. Garlend Publishing Inc., New York and London, 1994.

[4] N. Bansal, S. Chawla, and A. Blum. Correlation clustering. *The 43rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 238–247, 2002.

[5] S. Bergmann, J. Ihmels, and N. Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys Rev E Stat Nonlin Soft Matter Phys*, 67(3 Pt 1):03190201–18, 2003.

[6] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *The 44th Annual IEEE Symposium on Foundations of Computer Science*, page 524, 2003.

[7] G.L. Chen, C. Yan, and Y.F. Shen. Outlier analysis for gene expression data. *J. Comput. Sci. Technol.*, 19(1):13–21, 2004.

[8] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proc. ISMB'00*, pages 93–103. AAAI Press, 2000.

[9] M. Dawande, P. Keskinocak, J. M. Swaminathan, and S. Tayur. On the biclique problem in bipartite graphs. *GSIA working paper 1996-04, Carnegie-Mellon University*, 1997.

[10] M. Dawande, P. Keskinocak, J. M. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *J. Algorithms*, 41(2):388–403, 2001.

[11] J. DeRisi, L. Penland, P.O. Brown, et al. Use of a cDNA microarray to analyse gene expression patterns in human cancer. *Nat Genet*, 14:457–460, 1996.

[12] D. Emanuel and A. Fiat. Correlation clustering – minimizing disagreements on arbitrary weighted graphs. *In Proceedings of 11th ESA, volume 2832 of LNCS, Springer*, pages 208–220, 2003.

[13] S. P. Fodor, R. P. Rava, X. C. Huang, A. C. Pease, C. P. Holmes, and C. L. Adams. Multiplexed biochemical assays with biological chips. *Nature*, pages 555–556, 1993.

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

[15] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*, 97(22):12079–84, 2000.

[16] A.J.F. Griffiths, J.H. Miller, D.T. Suzuki, R.C. Lewontin, , and W.M. Gelbart. *An Intorduction to Genetic Analysis*. New York, 7th edition, 1996.

[17] J.A. Hartigan. *Clustering Algorithms*. John Wiley and Sons, 1975.

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

[19] Y. Kluge, R. Basri, J.T. Cheng, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res.*, 13(4):703–16, 2003.

[20] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, 1997.

[21] G.N. Lance and W.T. Williams. A general theory of classification sorting strategies. 1. hierarchical systems. *The Computer Journal*, 9:373–380, 1967.

[22] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2002.

[23] J. Liu and W. Wang. Op-cluster: Clustering by tendency in high dimensional space. *In Proceedings of the 3rd IEEE International Conferance of Data Mining*, pages 187–194, 2003.

[24] P. O. Brown M. B. Eisen, P. T. Spellman and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[25] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1965.

[26] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[27] A. Marshall and J. Hodgson. DNA chips: an array of possibilities. *Nat Biotechnol*, 16:27–31, 1998.

[28] D.W. Matula. Graph theoretic techniques for cluster analysis algorithms. In J. Van Ryzin, editor, *Classification and Clustering*, pages 95–129. Academic Press, 1977.

[29] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer, 1996.

[30] T. M. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. *In Proceedings of the Pacific Symposium on Biocomputing*, 2003.

[31] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Math.*, 131(3):651–654, 2000.

[32] T. Rozovskaia, O. Ravid-Amir, S. Tillib, G. Getz, E. Feinstein, H. Agrawal, A. Nagler, E.F. Rappaport, I. Issaeva, Y. Matsuo, U.R.

Kees, T. Lapidot, F. Lo Coco, R. Foa, A. Mazo, T. Nakamura, CM. Croce, G. Cimino, E. Domany, and E. Canaani. Expression profiles of acute lymphoblastic and myeloblastic leukemias with ALL-1 rearrangements. *Proc Natl Acad Sci U S A*, 100(13):7853–8, 2003.

[33] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[34] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

[35] E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. *In Proceedings of Pacific Symposium on Biocomputing*, 8:89–100, 2003.

[36] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl 1):S243–52, 2001.

[37] R. Shamir and R. Sharan. Algorithmic approaches to clustering gene expression data. In T.Jiang, T. Smith, Y. Xu, and M. Q. Zhang, editors, *Current Topics in Computational Molecular Biology*, pages 269–300. MIT Press, 2002.

[38] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Proceedings of the 28th International Workshop on Graph The-*

oretic Concepts in Computer Science (WG '02), L. Kucera (editor), LNCS 2573, pages 379–390, 2002.

[39] R. Sharan, A. Maron-Katz, and R. Shamir. CLICK and EXPANDER: a system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–99, 2003.

[40] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Proceedings of the 8th Annual International Conference on Intelligent Systems for Molecular Biology, (ISMB '00)*, pages 307–316. AAAI Press, 2000.

[41] Q. Sheng, Y. Moreau, and B. De Moor. Biclustering gene expression data by Gibbs sampling. *Bioinformatics*, 19(Supp 2):i196–i205, 2003.

[42] R. R. Sokal. Clustering and classification: Background and current directions. In J. Van Ryzin, editor, *Classification and Clustering*, pages 1–15. Academic Press, 1977.

[43] E. M. Southern. DNA chips: analysing sequence by hybridization to oligonucleotides on a large scale. *Trends in Genetics*, 12:110–115, 1996.

[44] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, ES. Lander, and TR. Golub. Interpreting patterns of gene expression with self organizing maps: Methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Science USA*, 96:2907–2912, 1999.

[45] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl 1):S136–44, 2002.

[46] A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. *To appear in the Handbook of Bioinformatics*, 2004.

[47] E. Tardos and V. Vazirani. Improved bounds for the max-flow min-multicut ratio for planar and kr,r-free graphs. *Inf. Process. Lett.*, 47(2):77–80, 1993.

[48] C. K. Tham, C. K. Heng, and W. C. Chin. Predicting risk of coronary artery disease from dna microarray-based. citeseer.ist.psu.edu/656840.html.

[49] J. Yang, W. Wang, H. Wang, and P. S. Yu. Delta-cluster: Capturing subspace correlation in a large data set. *In Proceedings of the 18th International Conference on Data Engineering*, pages 517–528, 2002.

[50] M. Yannakakis. Edge deletion problems. *SIAM J. Computing*, 10:297–309, 1981.