BI-(N-) CLUSTER EDITING AND ITS BIOMEDICAL APPLICATIONS

PENG SUN

A dissertation submitted towards the degree Doctor of Natural Sciences (Dr. rer. nat.) of the Faculty of Mathematics and Computer Science of Saarland University

2016 SAARBRÜCKEN

---- Date of the colloquium:
28.06.2017

--- Dean
Prof. Dr. Frank-Olaf Schreyer

--- Chairman
Prof. Dr. Volkhard Helms

--- Reporter:
Prof. Dr. Jan Baumbach,
Prof. Dr. Jiong Guo,
Prof. Dr. Dr. Thomas Lengauer

--- Scientific Assistant:
Dr. Markus List

Abstract

Executive Summary

The extremely fast advances in wet-lab techniques lead to an exponential growth of heterogeneous and unstructured biological data, posing a great challenge to data integration in nowadays system biology. The traditional clustering approach, although widely used to divide the data into groups sharing common features, is less powerful in the analysis of heterogeneous data from n different sources ($n \geq 2$). The co-clustering approach has been widely used for combined analyses of multiple networks to address the challenge of heterogeneity. In this thesis, novel methods for the co-clustering of large scale heterogeneous data sets are presented in the software package n-CluE: one exact algorithm and two heuristic algorithms based on the model of bi-/n-cluster editing by modeling the input as n-partite graphs and solving the clustering problem with various strategies.

In the first part of the thesis, the complexity and the fixed-parameter tractability of the extended bicluster editing model with relaxed constraints are investigated, namely the Π-bicluster editing model and its NP-hardness is proven. Based on the results of this analysis, three strategies within the n-CluE software package are then established and discussed, together with the evaluations on performances and the systematic comparisons against other algorithms of the same type in solving bi-/n-cluster editing problem.

To demonstrate the practical impact, three real-world analyses using n-CluE are performed, including (a) prediction of novel genotype-phenotype associations by clustering the data from Genome-Wide Association Studies; (b) comparison between n-CluE and eight other biclustering tools on GEO Omnibus microarray data sets; (c) drug repositioning predictions by co-clustering on drug, gene and disease networks. The outstanding performance of n-CluE in the real-world applications shows

its strength and flexibility in integrating heterogeneous data and extracting biological relevant information in bioinformatic analyses.

Kurzzusammenfassung

Die enormen Fortschritte im Bereich Labortechnik haben in jüngster Zeit zu einer exponentiell wachsenden Menge an heterogenen und unstrukturierten Daten geführt. Dies stellt eine große Herausforderung für systembiologische Forschung dar, innerhalb derer diese Datenmengen durch Datenintegration und Datamining zusammengefasst und in Kombination analysiert werden. Traditionelles Clustering ist eine vielseitig eingesetzte Methode, um Entitäten innerhalb grosser Datenmengen bezüglich ihrer Ähnlichkeit bestimmter Attribute zu gruppieren ("clustern"). Beim Clustern von heterogenen Daten aus n ($n \geq 2$) unterschiedlichen Quellen zeigen traditionelle Clusteringmethoden jedoch Schwächen. In solchen Fällen bieten Co-clusteringmethoden dadurch Vorteile, dass sie Datensätze gleichzeitig partitionieren können. In dieser Dissertation stelle ich neue Clusteringmethoden vor, die in der Software n-CluE zusammengeführt sind. Diese neuen Methoden wurden aus dem bi-/n-cluster editing heraus entwickelt und lösen durch Transformation der Eingangsdatensätze in n-partite Graphen mit verschiedenen Strategien das zugrundeliegende Clusteringproblem.

Diese Dissertation ist in zwei verschiedene Teile gegliedert. Der erste Teil befasst sich eingehend mit der Komplexitätanalyse verschiedener erweiterter bicluster editing Modelle, die sog. Π-bicluster editing Modelle und es wird der Beweis der NP-Schwere erbracht. Basierend auf diesen theoretischen Gesichtspunkten präsentiere ich im zweiten Teil drei unterschiedliche Algorithmen, einen exakten Algorithmus und zwei Heuristiken und demonstriere ihre Leistungsfähigkeit und Robustheit im Vergleich mit anderen algorithmischen Herangehensweisen. Die Stärken von n-CluE werden anhand von drei realen Anwendungsbeispielen untermauert: (a) Die Vorhersage neuartiger Genotyp-Phänotyp-Assoziationen durch Biclustering-Analyse

von Daten aus genomweiten Assoziationsstudien (GWAS);(b) Der Vergleich zwischen n-CluE und acht weiteren Softwarepaketen anhand von Bicluster-Analysen von Microarraydaten aus den Gene Expression Omnibus (GEO); (c) Die Vorhersage von Medikamenten-Repositionierung durch integrierte Analyse von Medikamenten-, Gen- und Krankeitsnetzwerken. Die Resultate zeigen eindrucksvoll die Stärken der n-CluE Software. Das Ergebnis ist eine leistungsstarke, robuste und flexibel erweiterbare Implementierung des Biclustering-Theorems zur Integration grosser heterogener Datenmengen für das Extrahieren biologisch relevanter Ergebnisse im Rahmen von bioinformatischen Studien.

Acknowledgements

First of all, I dedicate my special gratitude to both my supervisors, Prof. Dr. Jan Baumbach and Prof. Dr. Jiong Guo , for giving me the opportunity to work with them throughout my PhD studies, for providing me the wonderful research topic, for their extremely valuable advices, support and supervision, for their tireless commitments to helping me establish the correct attitude towards scientific researches, for their efforts to set up the researching environment for me to further develop my own ideas, and for their friendship and kindness outside of the office. I also owe my gratitude to Prof. Dr. Dr. Thomas Lengauer, for kindly reviewing my thesis and helping me during the writing.

I want to thank the Max Planck Institute for Informatics, International Max Planck Research School for Computer Science, the Center for Bioinformatics, the Cluster of Excellence on Multimodal Computing and Interaction and the University of the Saarland for the financial supports and the academic supports to my researches, without which I could never finish my study.

All my colleagues in the Computational Systems Biology group, including the brother-like Richard ("Richy"), who always gives me useful suggestions, my office-mates, the industrious Rashid, Josch, and other members Anne-Christin, Nic and Chris, all deserve my gratefulness for their kindness, their cooperations and their inspirations during the research.

Special thanks is dedicated to my student assistants Nora K. Speicher, Yuan Zhao and Chen Hong for their great work and dedication to the projects.

I also would like to thank the supporting staff and IT helpdesk for their help and support, namely, Sabine Nermerich, Polina Quaranta, Mona Linn, Jennifer Gerling and Dirk Raufer.

Outside my study in the institute, I owe many thanks to the priceless support coming from my parents. They have been extremely supportive, not only to my study, but also to my life.

To my parents, my supervisors and everyone who has helped me

Contents

List of Tables

List of Figures

Chapter 1

Motivation

Modern biology has entered into an era where quantitative tools play an indispensable role. Mathematical and statistical models as well as computer science techniques become widely applied and critically important in biological research, helping to store and to synthesize relevant data. A particular challenge is the integration of heterogeneous information from various sources in order to better reveal the underlying mechanisms of biological processes.

The focus of this thesis is one of the classic problems of data integration in bioinformatics which has been extensively studied by computer scientists and bioinformaticians — the clustering problem [80, 95, 96, 136]. Clustering is one of the most widely used unsupervised learning techniques for exploratory analysis in computer science, economics, social science and biological research. It is regarded as "unsupervised machine learning" because no pre-defined classes or labeled gold standard examples are provided. Clustering algorithms seek to extract the relations between the given data entities, by grouping them into different clusters based on certain key features. Such data processing methods are particularly meaningful in the context of biological studies, to discover "similar behavior" in a biological system, e.g. proteins with the same functional annotation or co-regulated genes. Compared to the

traditional clustering algorithms such as k-means [122, 123], graph-based clustering methods possess a list of remarkable advantages (see Chapter 2). In this thesis, we focus on a sub-branch of the traditional clustering problem, namely "biclustering" and "n-clustering" problems which refer to data mining techniques allowing simultaneous clustering of several data sets. The robustness of bi-/n-clustering on integrating heterogeneous data have been proven by a number of different studies (see Chapter 2 and Chapter 3). With this thesis we seek to investigate and prove important mathematical properties of the bi-/n-clustering problems and provide paradigms for practical applications of bi-/n-clustering in the context of biomedical data sets.

Before proceeding to the next chapters where details of the studies in this thesis are described, the first chapter is to introducing the background and the motivation of the work presented here.

1.1 The "-omics" Age

In the last two decades, we experienced an "explosion of information" in biological research, where an enormous amount of data has been generated through modern high-throughput wet-lab technology. The large amount of data provides on one hand unprecedented opportunities to understand the structures and functions of the cellular components on molecular level; on the other hand, challenges are induced by the sheer amount and complexity of data of heterogeneous types. To meet this challenge, a number of well-known integrated databases are established. For instance, the GenBank [159], as the largest database of genes, now stores over 197,000,000 sequences [13], including the whole genome sequences of ~ 600 eukaryotes, $\sim 6,000$ prokaryotes and $\sim 5,900$ viruses. The Genomes Online Database (GOLD) (https://gold.jgi.doe.gov) contains curated data from $26,117$ studies, $97,212$ sequencing projects, $78,579$ analysis projects, $239,100$ organisms and $15,887$ biosamples [131]. Similar trends can be

found in other databases like UniProtKb/Swiss-Prot [36] and GenExpressionOmibus (GEO) [11, 50]: UniProtKb/Swiss-Prot is a database containing more than $553,000$ annotated sequences, extracted and integrated from $205,244$ published references. Protein Data Bank (PDB) has incorporated over $124,000$ molecule structures [155]. GEO has incorporated over 1.6 million gene expression samples on over $15,000$ platforms. Fig. 1.1 illustrates the rapid growth of the three databases, which is just one glimpse of the dramatic progress resulting from the remarkable advances in wet-lab technology. One of the most outstanding developments emerged by the advent of Next Generation Sequencing (NGS) techniques, enabling scientists to process thousands of DNA sequences in a parallel manner and enhancing the throughput at exponential rates [129, 164, 215].

Nevertheless, the huge amount of raw and unstructured data inevitably leads to higher and stricter requirements regarding the data mining algorithms and more adequate mathematical models to extract interesting and meaningful information for further investigations, not only in terms of speed, but also in terms of robustness: in most cases, the heterogeneous raw data tends to be noisy and may contain a large extent of redundancy, on the one hand, but also suffer from missing values, on the other, especially regarding metadata. Great efforts have been put into mainstream thesauri like Medical Subject Heading (MeSH) [119] and OMIM [76], which have been proven to be highly valuable resources for drug target screening [106], disease similarity network building [66] and human metabolomics data integration[199]. Yet large amount of incompatible data exists in the terminology and ontology and, thus, mapping between vocabularies becomes indispensable [135].

1.2 Clustering and Biclustering

System biology is an interdiscplinary research field that focuses on building data-oriented models to extract and analyze the currently available heterogeneous data in order to gain relevant insights into the underlying complex biological processes. To this end, clustering analysis is one of the most powerful tools for system biology research. Its main aim is simple: to find a partition of biomedical objects into groups such that objects within one group are similar and objects in different groups are dissimilar to each other based on a given similarity function. Data clustering is widely used for data compression [193] and dimensionality reduction [214], which becomes increasingly important in Big Data processing. In the area of computational biology, clustering analysis plays a critical role in many research fields, e.g., taxonomy analysis, protein-complex prediction, microarray data mining, etc. In many other disciplines, clustering analysis is also widely applied: for instance, economics [24], information science [158] and computational linguistics [158].

Although clustering has been widely accepted and utilized in various studies, yet in some particular scenarios, the traditional clustering approach needs a further "refinement" to meet special requirements of heterogeneous data. For instance, microarray technology best exemplifies the astounding advances and developments in high-throughput wet-lab technology in the last two decades. One of the most common applications of microarray is to detect gene expression levels at genome scale under different conditions. Traditional clustering of microarray data, such as hierarchical clustering, aims to group similar genes (in most cases, rows in the matrix refer to genes) based on their expressions value across a set of conditions present in the matrix (columns in the matrix refer to experimental conditions), or to group similar conditions based on all genes. This procedure has been successfully applied for long and has been regarded as a standard data processing step in microarray data processing. However, clusters of genes emerging only in a subset of specific conditions,

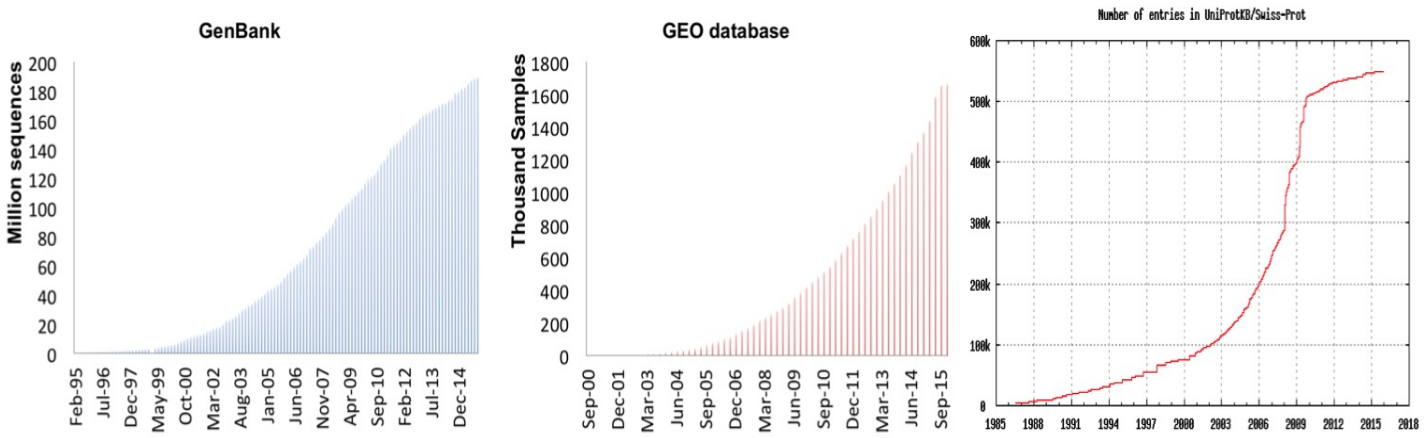

Figure 1.1: The growth of the sequences in GenBank (left), the samples in GEO (middle) and the protein records in UniProtKB/Swiss-Prot.

are simply ignored by traditional clustering. A different kind of approach, called "*biclustering*" or "*co-clustering*", offers a new direction to solve the problem, based on the algorithms that are able to cluster rows and columns simultaneously, forming rectangle blocks. Fig. 1.2 depicts the difference.

The strategy of biclustering has been proven advantageous over the traditional clustering in many ways. First, it is possible that the microarray contains a certain amount of genes or conditions that are not relevant for the experimental purpose. These irrelevant genes and conditions present noise in non-biclustering since information of all genes and conditions is considered. Second, the "*consistent behaviors*", e.g. up-regulated, down-regulated within only a subset of conditions, i.e. the "*local patterns*", would be difficult to detect. Third, biclustering approaches have been widely used to handle heterogeneous data coming from different sources [140, 153, 190]. A systematic comparison between biclustering and clustering on microarray data can be found in [148].

One strategy for solving the biclustering problem is to state biclustering in terms of graph theory by converting the biclustering of a matrix to clustering of a bipartite

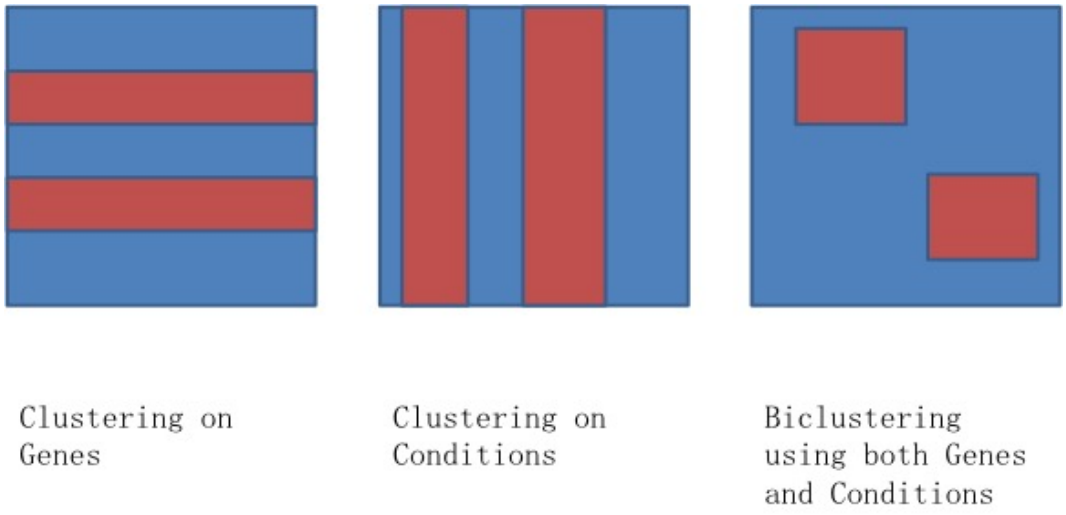

Figure 1.2: Demonstration of the difference between traditional clustering of rows(left), traditional clustering of columns (middle) and biclustering (right).

graph, i.e., the graph with two separate sets of nodes, and edges only between the two node sets (refer to Chapter 2 for details). This transformation from a matrix to a bipartite graph was first introduced by Hartigan *et al.* [77] in this context. A number of studies have followed this strategy and proven the power of biclustering by using bipartite graph models [22, 46, 121, 184].

The model of bicluster editing, expanded from the classic cluster editing problem (sometimes called "Transitivity Editing"), provides a possible solution for the bipartite clustering problem converted from the matrix biclustering. Given a bipartite graph, the aim is to convert the input to a solution graph with only disjoint bicliques by "*virtually*" inserting and deleting edges, such that the cost paid for the *changes on the input graph* (including insertions and deletions) are minimized. See Chapter 2 for more information.

Many studies on bicluster editing have been carried out, both on its theory and practical applications, forming the background of this thesis (see Chapter 2 and Chapter 3 for details). One of the major advantages of the bicluster editing model, compared to other biclustering algorithms, is that no particular prior knowledge or distri-

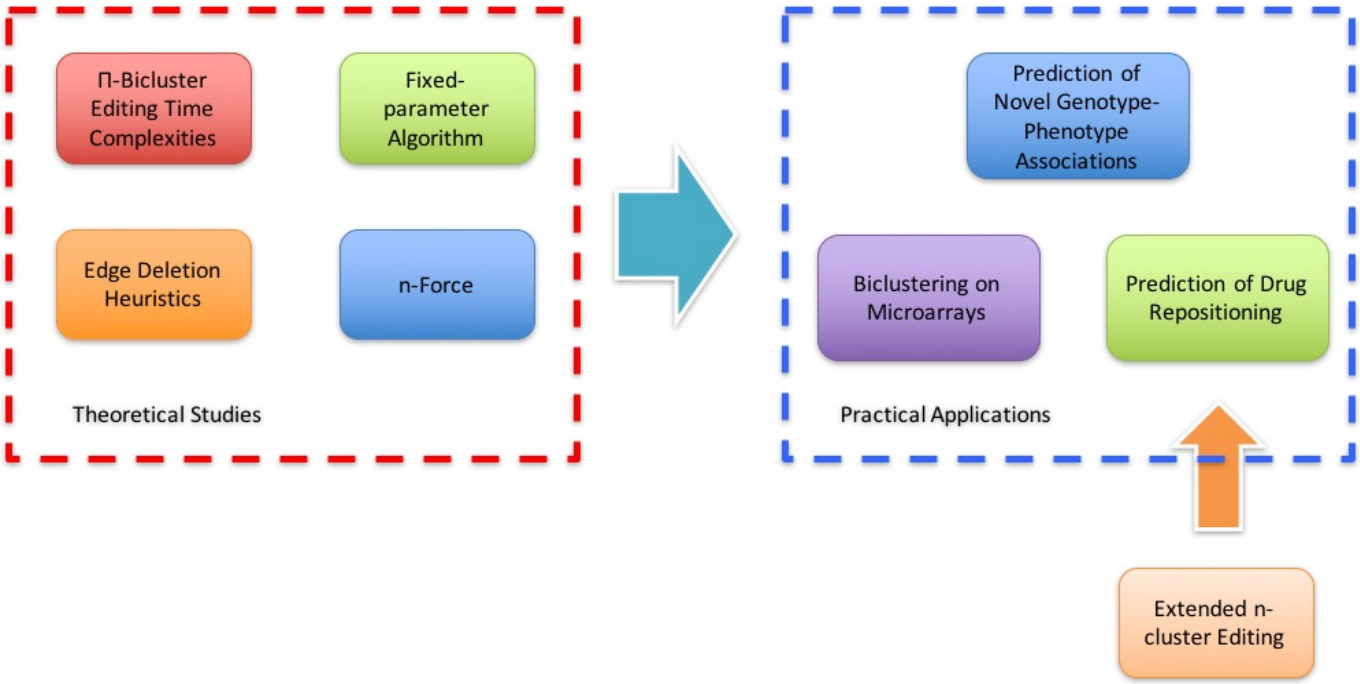

Figure 1.3: The basic rationale behind the thesis.

bution is assumed [202]. Furthermore, the bicluster editing model can be extended to n-partite graphs where the data entities are divided into n different groups, providing a general approach of integrating an arbitrary number of heterogeneous data sets.

1.3 The Thesis Structure

Chapter 2 and Chapter 3 introduce biclustering and parameterized theory. In Chapter 4, proofs regarding the time complexity of Π-bicluster editing are presented. Afterwards in Chapter 5, the main contribution of this thesis, a software package for weighted bi-/n-cluster editing, named n-CluE, is introduced. The following chapters mainly focus on the biomedical utilities of n-CluE: Chapter 6 presents the prediction of novel genotype-phenotype associations by modeling Genome-Wide Association Study (GWAS) data as bipartite graphs. Chapter 7 provides a systematic comparison between n-CluE and eight other prevailing biclustering tools on gene expression

data sets, showing that our strategy outperforms all other tools presented in the study. Chapter 8 illustrates another promising application of n-CluE by performing n-clustering on a drug – gene – disease network to integrate heterogeneous data and predict novel drug repositionings. In conclusion, we studied the model of bi-/n-cluster editing in depth and demonstrated its power in different biomedical contexts. Note that some of the content of this thesis has already been published in the following list of publications.

1.4 List of Publications

- Peng Sun, Jiong Guo, and Jan Baumbach. Integrated simultaneous analysis of different biomedical data types with exact weighted bi-cluster editing. *J Integr Bioinform*, 17, 2012

- Richard Röttger, Prabhav Kalaghatgi, Peng Sun, Siomar de Castro Soares, Vasco Azevedo, Tobias Wittkop, and Jan Baumbach. Density parameter estimation for finding clusters of homologous proteins-tracing actinobacterial pathogenicity life styles. *Bioinformatics*, pages 215–222, 2012

- Peng Sun, Jiong Guo, and Jan Baumbach. Biclue-exact and heuristic algorithms for weighted bi-cluster editing of biomedical data. In *BMC proceedings*, volume 7, page S9. BioMed Central Ltd, presented at GLBIO2013, 2013

- Peng Sun, Nora K Speicher, Richard Röttger, Jiong Guo, and Jan Baumbach. Bi-force: large-scale bicluster editing and its application to gene expression data biclustering. *Nucleic Acids Research*, page gku201, 2014

- Peng Sun, Jiong Guo, and Jan Baumbach. Complexity of dense bicluster editing problems. *Computing and Combinatorics*, pages 154–165, 2014

- Peng Sun, Jan Baumbach, and Jiong Guo. Efficient large-scale bicluster editing. In *German Bioinformatics Conference 2014*, pages 54–60

- Peng Sun, Jiong Guo, Rainer Winnenburg, and Jan Baumbach. Integrated literature mining and drug-gene-disease triangulation reveals ten thousand new purposes for existing medication. *Drug Discovery Today*, (In press), 2016

Chapter 2

Background: Biclustering and Bicluster Editing

In this chapter, we introduce the concept "*biclustering*", the algorithms developed in previous studies to solve the biclustering problem, and major applications of biclustering.

2.1 Clustering

Data clustering is an important approach in unsupervised machine learning procedures that has been applied in almost every area of system biology (two comprehensive reviews can be found here [16, 96]), for instance: (1) gene and protein clustering, including the database Clusters of Orthologous Genes (COG) [110, 126, 186], the software packages of TransClust [130, 200, 203] and non-complete TransClust [157]; (2) phylogenetic analysis on pathogens [156]; and (3) microarray data mining [124, 179].

The clustering of differentially expressed genes is usually performed as the initial step in microarray data mining to discover a rough "relation network" amongst all genes, before proceeding to the actual detection on specific pathways or regulatory processes. Gene expression data is arranged in a matrix where rows refer to differ-

ent genes and columns refer to the varying experimental conditions. The elements in the matrix represent the intensities of the expressions of the genes under certain conditions. Data clustering algorithms thus focus on the analysis of the matrix elements, usually pursuing one of the following objectives: (a) to cluster genes based on their complete expression profiles (the expression levels across all experimental conditions), and (b) to cluster the experimental conditions based on expression patterns of all involved genes. The two objectives mentioned above refer to two distinct clustering directions in gene expression data mining, referred to as "*traditional*" clustering. Many common clustering methods can be used to implement this strategy: for instance, hierarchy clustering [51], k-means clustering [79], self-organizing maps (SOM) [108] ; several other groups have developed software tools implementing traditional clustering strategies [27, 28, 91, 151, 172, 180].

However, the traditional approach is not perfect in many cases, which largely limits its applications in real world. In fact, many interesting clusters of gene expression data fall into *local rectangles* in the microarray matrix, meaning that a group of genes are co-regulated only under a subset of conditions but not all. This also concurs with our knowledge about the cellular processes where a small group of genes correlate with each other only under specific conditions but do not correlated in other cases. In this respect, using the clustering methods considering expression profiles under all conditions might introduce unnecessary noise. For instance, a microarray experiment may contain 1,000 genes extracted from 10 samples, where only 200 genes are up-regulated in 3 samples. The expression intensities of the other 800 genes and 7 samples provide no useful information. In order to detect the groups of genes sharing a common pattern only under a subset of conditions, a new methodology is required, to cluster genes and conditions simultaneously and to extract local patterns hidden in the matrix. Such simultaneous clustering on microarrays is called *biclustering*.

2.2 Biclustering

2.2.1 Introduction

The concept of biclustering was first introduced by Hartigan [77], referred to as "block clustering". However, Hartigan's research focused rather on the algorithmic perspective of biclustering than its bioinformatic applications. The first biclustering method applied on microarray data was designed by Cheng and Church [30], to identify groups of genes with significantly less variance in their expression profiles than others. Since then, various biclustering algorithms have been designed and applied in real-world data analysis. Before we proceed to the discussion on bicluster models, we need some preliminary definitions. As mentioned in the previous paragraphs, gene expression data forms a matrix with rows representing genes and columns representing conditions. Given a matrix M, denote the row set and the columns set as R and C, where $R = \{r_1, r_2, ..., r_m\}$ and $C = \{c_1, c_2, ..., c_n\}$. r_i and c_j refer to the row i and column j in M, respectively. The element $m_{ij} \in M$ ($m_{ij} \in r_i$ and $m_{ij} \in c_j$) is the expression level of gene i under condition j. A set of rows is denoted as X, $X \subseteq R$ and a set of columns is denoted as Y, $Y \subseteq C$. M_{XY} refers to a "sub-matrix" inside M defined by X and Y. A bicluster M_{XY} in the matrix M can be defined as a subgroup of rows X and columns Y where the elements belonging to M_{XY} exhibit certain "common behavior" (e.g. up-regulated or down-regulated). The aim of biclustering is: given a matrix M_{mn} where $R = \{r_1, r_2, ..., r_m\}$ and $C = \{c_1, c_2, ..., c_n\}$, try to identify a subset of rows and columns, M_{XY} with $X \subseteq R$ and $Y \subseteq C$, such that M_{XY} satisfies certain criteria.

In a matrix M_{IJ} with I rows and J columns, we denote the mean of i^{th} row as m_{iJ}, the mean of j^{th} column as m_{Ij}, and the mean of the whole matrix as m_{IJ}. The

quantities are defined as follows:

$$m_{iJ} = \frac{1}{|J|} \sum_{j \in J} m_{ij} \tag{2.1}$$

$$m_{Ij} = \frac{1}{|I|} \sum_{i \in I} m_{ij} \tag{2.2}$$

$$m_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} m_{ij} \tag{2.3}$$

Another common strategy for solving biclustering problem is to convert the input matrix into a weighted bipartite graph, as shown in Fig. 2.1. A bipartite graph is a special type of graph, often denoted as $G = (U, V, E)$, where U and V represent two sets of nodes and E refers to the edge set. All edges $e \in E$ in a bipartite graph have only one end node located in U and the other end node located in V. No edge is allowed to link nodes in the same set. A data matrix M_{RC} can be easily converted into a bipartite graph. Denote the constructed bipartite graph as $G = (U, V, E)$: for each row in the matrix, a node $u \in U$ is created, for each column in the matrix, a node $v \in V$ is created. The elements in the matrix are regarded as pairwise similarities between the nodes in U and the nodes in V. A user-specified threshold t is used to construct the edges. A node pair (u, v) with the corresponding similarity $s(u, v)$ given by $m_{u,v}$ in the matrix is connected with an edge $e \in E$ in the resulting bipartite graph if and only if the similarity $s(u, v) > t$. Note that in our study, unless otherwise specified, we only deal with simple undirected graphs with no self-loops.The transformation from matrix into bipartite graph allows the strategies applied on graph clustering to be extended to solve the matrix biclustering problem (e.g. spectral clustering).

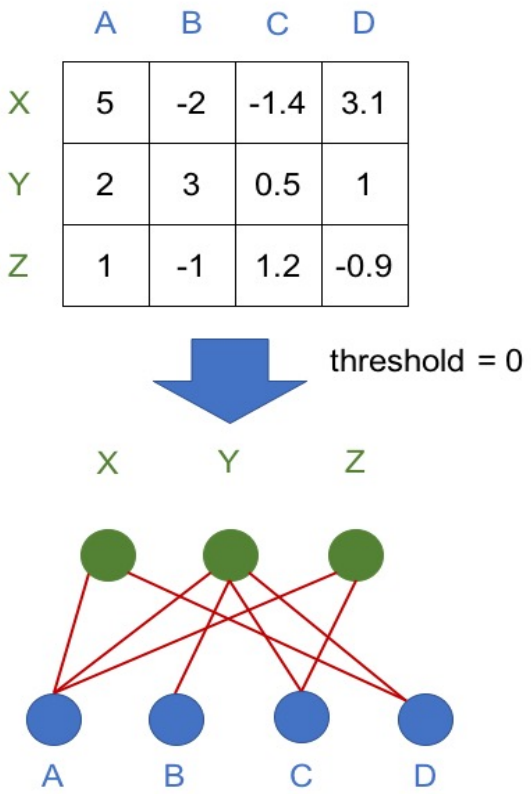

Figure 2.1: The transformation from a matrix into a bipartite graph. For each row and each column, a node is constructed in the resulting bipartite graph. The elements in the matrix are modeled as the edges in the bipartite graph. A user-specific threshold is used to judge if the two nodes are connected.

2.2.2 Bicluster Models

In biclustering, the type of the local patterns which the biclustering algorithms are seeking for are referred to as *bicluster models*.

The identification of biclusters in either matrices or bipartite graphs varies greatly regarding to the different bicluster models, namely the criteria of *being a bicluster*. We distinguish four cases of biclusters [124]: (1) biclusters with constant values; (2) biclusters with constant values in rows or columns; (3) biclusters with coherent values; (4) biclusters with coherent evolutions.

The first three bicluster models, i.e., biclusters with constant values, biclusters with constant values on rows or columns and biclusters with coherent values, are defined with regard to the numeric values, describing certain relations between the bicluster and the background. Such biclusters are to be extracted by searching a sub-matrix with values significantly different from the rest of the matrix (the "background"). For instance, the values in the bicluster may be significantly higher (up-regulated) or lower (down-regulated) than the background values. The implementations to search such biclusters vary greatly, e.g. greedy iterative searching strategy, exhaustive enumeration, etc. See Section 2.2.4 for more details.

The fourth bicluster model — the biclusters with coherent evolutions — are based on non-numerical information. Subareas in the data matrix with preserved order, rank, or consistent positive or negative changes are considered as coherent evolutions. In the coherent evolution model, the elements in the matrix are treated in a non-parametric way. The detection of such biclusters is discussed in the Section 2.2.4.

Fig.2.2 illustrates all four bicluster models with simple 3 x 3 matrices. The details and the underlying mechanisms are explained the in the following.

In gene expression, the constant model indicates a stable and similar expression level across genes and conditions. Bicluster with constant rows or columns refers to a situation where a subset of genes have stable expression changes across a subset of genes or conditions (Fig. 2.2 (2-3)). Bicluster with coherent values presents more complex relations between genes and conditions. In a bicluster with coherent values, the numeric values do not necessarily remain constant across rows or columns, but are assumed to follow an underlying model, either additive or multiplicative. Bicluster with coherent evolution is useful when the researcher is focusing on finding non-parametric relations between genes and conditions, for instance: up-regulation, down-regulation or preserved ranks.

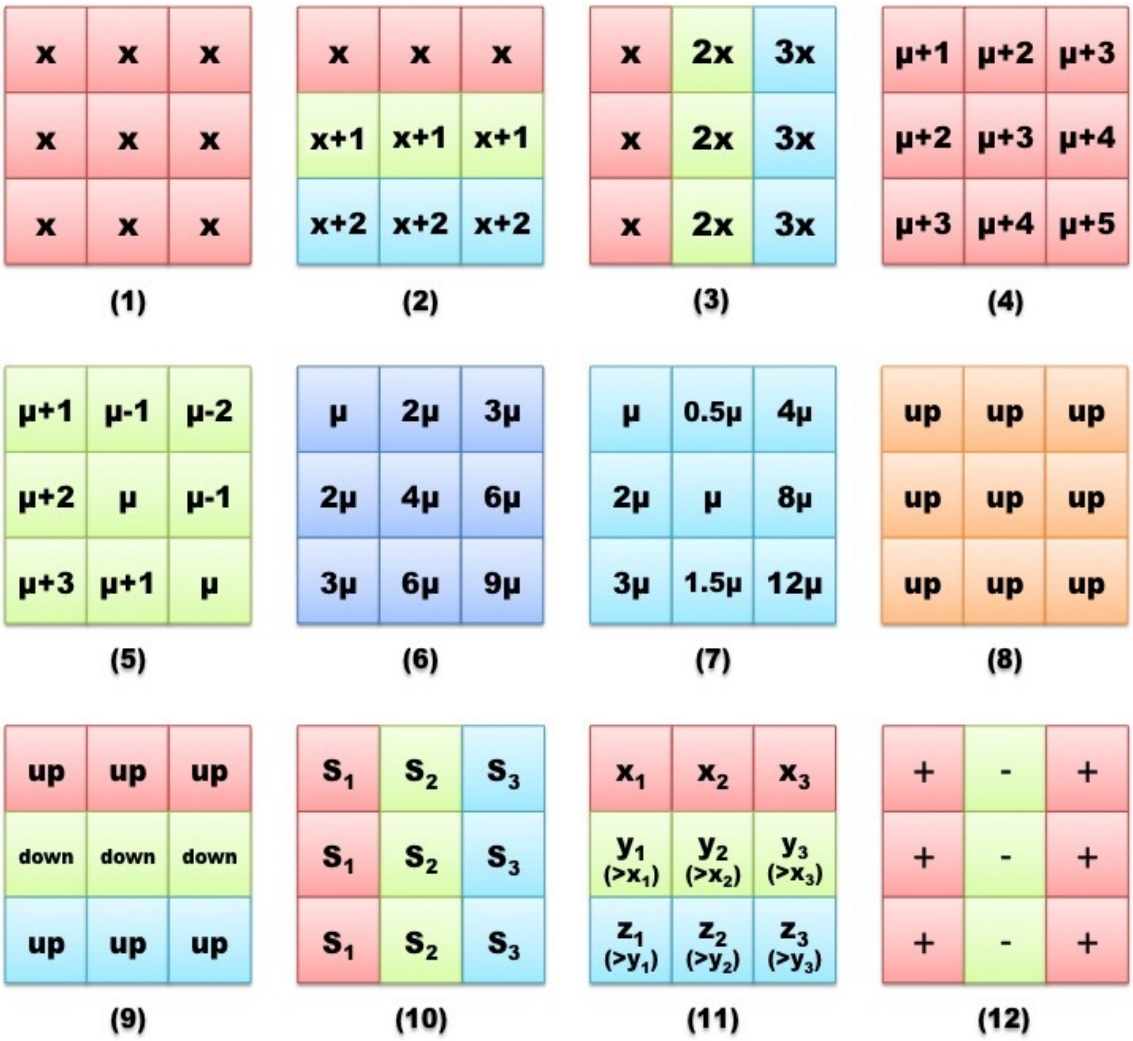

Figure 2.2: Demostration of bicluster models. (1) Constant bicluster, (2) Constant rows, (3) Constant columns, (4) Coherent values with additive model, (5) Coherent values with additive model, (6) Coherent values with multiplicative model, (7) Coherent values with multiplicative model, (8) Coherent evolution with constant up-regulation, (9) Coherent evolution with constant up-regulation and down-regulation arranged by rows, (10) Coherent evolution with constant status arranged by columns, (11) Coherent evolution with preserved order, (12) Coherent evolution with preserved (positive/negative) changes.

The evaluation of the clustering algorithms is also highly dependent on the chosen bicluster models. The scoring functions for evaluation should be carefully designed to reflect the specific characteristics of the assumed model. For instance, variances are commonly used as a quality indicator for constant biclusters. In the following

section, the features of the four bicluster models will be discussed in detail, showing how bicluster structures influence the corresponding biclustering algorithms.

Biclusters with Constant Values

Biclusters with constant values are the simplest of the four models. Given a submatrix M_{IJ}, a perfect constant bicluster should follow the rule that all values within the bicluster are equal:

$$m_{ij} = \sigma \tag{2.4}$$

In a real-world data set, a perfect constant bicluster is often unrealistic. The elements in the matrix are inevitably affected by noise. Thus, the elements inside a constant bicluster are better modeled as:

$$m_{ij} = \sigma + \epsilon_{ij}, \tag{2.5}$$

where ϵ_{ij} is noise associated with element m_{ij}. To search for constant biclusters, one of the most straightforward methods is to re-arrange the order of rows and columns in a way that rows and columns with equal elements are located closer to each other, with score functions carefully designed to remove the effect of noise.

The most commonly used scoring function to evaluate the quality of the constant biclusters, as mentioned in the previous section, is the variance of the sub-matrix, defined as follows:

$$\mathrm{Var}(M_{IJ}) = \sum_{i \in I j \in J} (m_{ij} - m_{IJ}) \tag{2.6}$$

However, using such scoring function may lead to *over-segmentation* of the biclustering results, i.e., the matrix is partitioned into biclusters with only one row and one column. Apparently, such biclusters will have zero variance but little biological meaning. To avoid obtaining such "optimal" solutions, additional restrictions controlling

the sizes or the number of the biclusters are usually employed to achieve maximal sizes of the resulting biclusters.

For instance, the partition-based algorithm named block clustering, designed by Hartigan [77] continuously splits the matrix into sub-matrices and tries to find a sub-matrix with lowest variance. To avoid the over-segmentation, block clustering requires a parameter K as the assumed number of biclusters in the results. The scoring function is modified to:

$$\mathrm{Var}(M_{IJ})_K = \sum_{k=1}^{K} \sum_{i \in I j \in J} (m_{ij} - m_{IJ}) \tag{2.7}$$

Rather than seeking for the biclusters with lowest individual variances, the algorithm now searches for K biclusters with an overall minimized variance computed by the formula above.

The strategy of Hartigan [77] was then further refined by Tibshirani *et al.* [187] by adding to the original algorithm a backward pruning step to filter off the high variance parts in the resulting biclusters.

Biclusters with Constant Rows or Columns

The assumption of an overall constant value in a bicluster with constant values is apparently over-simplified. A more complex bicluster model — biclusters with constant rows or columns — is thus of more practical interest. It is reported that the biclusters with constant rows or columns (Fig. 2.2 (2) and (3)), compared to the constant biclusters, are closer to the real world and chosen as the bicluster model by many algorithms [124].

Two models are commonly used as underlying mechanism for biclusters with constant rows or columns: additive model and multiplicative model. These two model the relation between the baseline row (or column) and other rows (or columns) in the

bicluster. The additive model is formulated as:

$$m_{ij} = \sigma + \alpha_i \tag{2.8}$$

$$m_{ij} = \sigma + \beta_j \tag{2.9}$$

The multiplicative model is formulated as:

$$m_{ij} = \sigma \times \alpha_i \tag{2.10}$$

$$m_{ij} = \sigma \times \beta_j \tag{2.11}$$

A simple strategy to identify biclusters with constant rows or columns is to perform normalization on rows or columns in the data matrix. This will transform the biclusters with constant rows or columns into the simple case of constant biclusters mentioned in the previous section, such that all algorithms for biclusters with constant values can be applied. Whether to normalize rows or columns differs from case to case: the row normalization is intended for biclustering with constant rows and column normalization is for biclusters with constant columns. Getz *et al.*'s study has proven the effectiveness of this normalization approach. Getz *et al.* also introduced a strengthened normalization method, enabling their tool to detect biclusters with constant rows or columns, as shown in Fig. 2.2 (2), (3), and biclusters with coherent values as well, as shown in Fig. 2.2 (4), (5), (6) and (7).

Non-normalizing tools to detect biclusters with constant rows or columns usually add noise to the bicluster model formula or try to identify biclusters with rows or columns falling in certain intervals. This relaxed restriction greatly enhances the robustness of the biclustering algorithm against the unexpected noise [23].

For instance, Califano *et al.* [23] defined a "σ-valid ks-pattern" bicluster. A submatrix M_{IJ} satisfies a "σ-valid ks-pattern" if $|I| = k$, $|J| = s$ and the difference

between the maximum and the minimum values in each row inside the sub-matrix satisfy:

$$\max\{r_i\} - \min\{r_i\} < \sigma \quad \forall i \in I, \tag{2.12}$$

where r_i represents the set of values in i^{th} row in the sub-matrix. Califano *et al.* [23]'s method seeks to detect the largest sub-matrix that satisfies the σ-valid ks-pattern in the input data matrix and conducted statistical tests to show the significance of the resulting biclusters.

Probabilistic models can also be applied to perform biclustering. For instance, Sheng *et al.* [165] modeled the matrix using a Bayesian framework and a Gibbs sampling strategy. This model approaches the biclustering problem by modeling each bicluster model with a probabilistic model of posterior frequencies. The data in a bicluster are considered to follow multinomial distributions and different columns within the bicluster are independent from each other. Assuming a gene-condition orientation, Sheng *et al.* checks by Gibbs sampling whether the values within a bicluster are consistent. With this approach, Sheng *et al.* [165] managed to identify biclusters with constant rows.

Segal *et al.* [160, 161] derived a probabilistic Bayesian network, implementing a general strategy that gives probabilities of gene expressions depending on different conditions. Based on the conditional probabilities, a Bayesian network of the gene expression dependencies is then built using a Probabilistic Relation Model (PRM) [160, 161], a rich representation language extending Bayesian networks with probabilistic semantics [58, 109], to measure the dependencies of the gene expression levels on certain conditions. Based on the probabilistic dependencies, Segal *et al.* managed to find biclusters with constant rows or columns, biclusters with coherent values and biclusters with coherent evolutions (see the following sections). The structure and the parameters of the Bayesian network are obtained by using a Condition Probability Distribution-tree (CPD-tree), which is an extended Bayesian network designed to

learn the local structures in conditional probability distributions, followed by a greedy strategy combined with a simulated annealing procedure to avoid local optima [57].

Biclusters with Coherent Values

The underlying models for biclusters with coherent values are more complex than the previously mentioned models. In biclusters with coherent values, the elements are affected by both row and column effects. This model also has two underlying mechanisms, namely the additive model and multiplicative model. Given a submatrix M_{IJ}, the additive model regards the element m_{ij} as:

$$m_{ij} = \sigma + \alpha_i + \beta_j \tag{2.13}$$

The additive model decomposes a value into three effects: the bicluster effect μ, the row effect α_i and the columns effect β_j. Fig. 2.2 (4) and (5) show two examples of biclusters with coherent values of additive model.

Biclusters with multiplicative coherent values models the elements as follows:

$$m_{ij} = \sigma \times \alpha_i \times \beta_j \tag{2.14}$$

The only difference is that the multiplicative model uses the product of the three effects (Fig. 2.2 (6) and (7)). Multiplicative models can easily be transformed into additive ones by computing the logarithm of each effect:

$$\sigma' = \log(\sigma) \tag{2.15}$$

$$\alpha'_i = \log(\alpha_i) \tag{2.16}$$

$$\beta'_j = \log(\beta_j) \tag{2.17}$$

$$\log(m_{ij}) = \sigma' + \alpha'_i + \beta'_j \tag{2.18}$$

A number of biclustering algorithms were developed to discover biclusters with coherent values. The first biclustering tool applied on microarray data, developed by Cheng and Church [30], was mainly focusing on the discovery of coherent biclusters generated by an additive model. In Cheng and Church's study, a bicluster was defined as a sub-matrix with small "mean square residue". The mean square residue $H(I, J)$ is the scoring function for assessing the quality of biclusters, defined as:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r(m_{ij})^2 \tag{2.19}$$

where,

$$r(m_{ij}) = m_{ij} - m_{Ij} - m_{iJ} + m_{IJ}. \tag{2.20}$$

m_{ij} is the element at row i and column j, m_{iJ} is the mean of the row i, m_{Ij} is the mean of the columns j, and m_{IJ} is the mean of the complete sub-matrix. According to the assumption of Cheng and Church [30], a perfect bicluster would have $H(I, J) = 0$. In such a bicluster, the element m_{ij} is exclusively determined by the row mean m_{iJ}, the column mean m_{Ij} and the subset mean m_{IJ}, as follows:

$$m_{ij} = m_{iJ} + m_{Ij} - m_{IJ} \tag{2.21}$$

Cheng and Church also introduced the concept of δ-bicluster. For a certain $\delta \geq 0$, a sub-matrix is a δ-bicluster if and only if $H(I, J) \leq \delta$. The perfect bicluster mentioned in the previous paragraph is a special case with $\delta = 0$.

In addition to the commonly used mean square residue as the scoring function, other measures were also developed to evaluate the quality of biclusters with coherent

values. One of them developed by Hartigan [77] is defined as:

$$r'(m_{ij}) = m_{ij} - m_{IJ} \tag{2.22}$$

This quality measure has been adopted by Cheng and Church [30] and Yang *et al.* [208] to assess the consistency of the biclusters,

Other algorithms like that by Cho *et al.* [32], compute all biclusters simultaneously, using total squared residue as the scoring function $H_T(I, J)$, defined as:

$$H_T(I, J) = \sum_{b=1}^{B} H_b(I, J) \tag{2.23}$$

where $H_b(I, J)$ is the mean square residue of bicluster b in sub-matrix M_{IJ}.

In the algorithm Flexible Overlapped biClustering (FLOC) [207, 208], an additional occupancy threshold θ is defined over the δ-bicluster, as the fraction of "specific elements" in each row and column. Given a sub-matrix M_{IJ}, the number of specific values in row i is denoted as $|J'_i|$ and the number of specific values in column j as $|I'_j|$. The scoring functions measuring the quality of the biclusters are defined as:

$$m_{iJ} = \frac{1}{|J'_i|} \sum_{j \in J'_i} m_{ij} \tag{2.24}$$

$$m_{Ij} = \frac{1}{|I'_j|} \sum_{i \in I'_j} m_{ij} \tag{2.25}$$

$$m_{IJ} = \frac{1}{v_{IJ}} \sum_{i \in I'_j, j \in J'_i} m_{ij} \tag{2.26}$$

$$m_{ij} = \begin{cases} m_{ij} - m_{iJ} - m_{Ij} + m_{IJ} & \text{if } m_{ij} \text{ is specified.} \\ 0 \quad \text{otherwise} \end{cases} \tag{2.27}$$

where v_{IJ} is the number of the specific values in the sub-matrix m_{IJ}, defined as the volume of the sub-matrix.

Yang *et al.* [208] also considered other mean measures like geometric mean and the square mean, defined as:

$$H(I,J) = \frac{1}{v_{IJ}} \sum_{i \in I'_j, j \in J'_i} |r(m_{ij})| \tag{2.28}$$

$$H(I,J) = \frac{1}{v_{IJ}} \sum_{i \in I'_j, j \in J'_i} r(m_{ij})^2 \tag{2.29}$$

Wang *et al.* [192] designed δ-pClusters with an additional measure named "*pscore*". A *pscore* is defined over a 2×2 sub-matrix M_{RC}, where $R = \{r_1, r_2\}$, $C = \{c_1, c_2\}$. The pscore is computed as follows:

$$pscore(M_{RC}) = |(m_{r_1c_1}) - (m_{r_1c_2}) - (m_{r_2c_1} - m_{r_2c_2})| \tag{2.30}$$

A sub-matrix M_{IJ} is a δ-pCluster if and only if for all 2×2 matrix $M_{RC} \subseteq M_{IJ}$, we have $pscore(M_{RC}) \leq \delta$. The model of δ-pClusters can also be used to detect additive bicluster model.

In addition to the additive and multiplicative models, Lazzeroni and Owen presented the plaid model [114]. Its major distinction is that the interactions between biclusters are considered as a contributing factor to the matrix elements. The plaid model models the matrix elements as a linear combination of different layers, representing various effects that influence the values, defined as:

$$m_{ij} = \sum_{k=0}^{K} \theta_{ijk} \rho_{ik} \kappa_{ij} \tag{2.31}$$

Here, θ_{ijk} indicates the effect of biclusters k on element i, j; ρ_{ik} and κ_{jk} are two binary indicators ($\rho_{ik} \in \{0,1\}$, $\kappa_{jk} \in \{0,1\}$), representing whether the row i and

the column j are in the bicluster k or not. In the plaid model, the element m_{ij} is represented as the sum of the effects from all biclusters that contain m_{ij}, with each bicluster modeled as a *layer*. The plaid model aims to minimize the square error formulated as follows:

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(m_{ij} - \theta_{ij0} - \sum_{k=1}^{K}\theta_{ijk}\rho_{ik}\kappa_{jk})^2 \tag{2.32}$$

This scoring function describes the squared error between the real value in the matrix and the theoretical value resulting from the plaid model. θ_{ij0} is considered as the background. Notably, by varying θ_{ijk}, the plaid model is flexible in identifying biclusters of different types: when $\theta_{ijk} = \mu_k$ the plaid model represents a simple constant model; by setting $\theta_{ijk} = \mu_k + \alpha_{ik}$ and $\theta_{ijk} = \mu_k + \beta_{jk}$, the plaid model can be used to discover biclusters with constant columns and rows; a complete additive formula such as $\theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{jk}$ enables the plaid to model the biclusters with coherent values.

Biclusters with Coherent Evolution

The model of biclusters with coherent evolution focuses more on the trends or changes in the data matrix, regardless of the exact values, as represented in Fig. 2.2 (8), (9), (10), (11) and (12).

Ben-Dor *et al.* [12] developed the Order-Preserving Sub-Matrix (OPSM) algorithm based on a non-parametric statistical model, under which the biclusters are defined as groups of rows with preserved linear ranks. The algorithm aims at finding a certain subset of the columns such that the rank of the rows is strictly increasing after permutation, as shown in Fig. 2.2 (11).

Another algorithm developed to search for coherent evolution patterns is xMOTIFs, designed by Murali and Kasif [134]. xMOTIFs has been proven successful in

finding preserved gene expression biclusters (called *motifs* in [134]), defined as a subset of rows (genes) with their expression levels preserved under a certain subset of conditions. The genes are first categorized into several *states*, pre-defined according to the actual values in the data matrix, e.g. up-regulated or down-regulated. The xMOTIFs method then aims to find the largest subset such that the state of the genes is preserved. Two user-specified parameters — α, β, representing the minimum fractions of the rows and the columns in the resulting biclusters over the total numbers of the rows and the columns in the matrix, are required to control the sizes. No bicluster with sizes smaller than the specified size-fractions are reported.

Similarly, Tanay *et al.* [183] defined the biclusters with coherent evolution as a subset of rows that have the same response across a subset of columns. Here, the concept "same response" is defined as the significant up-regulations/down-regulations of expression levels relative to the background signal.

2.2.3 Problem Complexity

The complexity of the biclustering problem depends on the required characteristics of the biclusters, specifically on the complexity of judging whether a sub-matrix satisfies the requirements. Most of the common variants of the biclustering problem are proven to be NP-complete. Consider the simplest case where the matrix consists of only 0s and 1s, and the task is to find a sub-matrix of 1s with maximum size. If we consider the matrix as a graph, where $e(u_i, v_j) \in E$ if and only if $m_{ij} = 1$, then it is trivial to show that this biclustering problem is equivalent to the problem of finding the maximum biclique in the graph, to which the classic NP-complete "clique problem" [20] can be reduced. More complex cases arise when the elements in the matrix are numeric values and the qualitites of the biclusters are dependent on the matrix elements. Provided that the large majority of the biclustering problems are NP-hard, algorithms designed to seek for exact solutions are apparently too expensive

since searching for exact solutions always suffers from exponential increase in running times. Heuristic approaches are often used for biclustering which significantly reduces the running times without much compromise on accuracy.

2.2.4 Biclustering Algorithms

The biclustering methods differ in several aspects. First, many methods are designed for particular bicluster model(s) and thus vary mostly in their scoring functions. Second, some methods identify one bicluster in one run while the others simultaneously discover all biclusters satisfying the criteria. Third, biclustering algorithms follow various strategies to detect biclusters, such as greedy iterative strategy, exhaustive enumeration, statistical models, divide and conquer, etc.. Additionally, some methods require training to refine their parameters. In the following part of this section I briefly introduce several algorithms that represent the state of the art.

The most straightforward method is to integrate the data analysis results of two separate clustering runs applied on the rows and the columns. Several studies used this strategy to find biclusters with constant values and biclusters with constant rows or columns. For instance, Getz *et al.* [64] developed a method named Coupled Two-Way Clustering (CTWC). In CTWC, the rows and the columns are first repeatedly clustered by a hierarchical clustering algorithm, forming two lists of row and column clusters. The algorithm then integrates the results from the row and column clustering. To avoid exhaustive enumeration of all cluster combinations, Getz *et al.* [64] identifies only the *stable clusters* based on the in-bicluster variances and discards all others. Afterwards, the biclusters are computed by merging small stable clusters or breaking up the large ones into smaller but more stable biclusters. A parameter T is used to control the bicluster size. Usually the algorithm starts with $T = 0$ and T increases as the algorithm proceeds, until T reaches a certain level. All biclusters which survive the procedure are believed to be qualified and reported. Note that the

rows and the columns can be clustered by any appropriate clustering method, e.g. hierarchical clustering, k-means, etc.

Tang *et al.* [185] developed the Interrelated Two-Way Clustering (ITWC) method following a similar strategy. ITWC also combines the clustering results obtained from rows and columns separately. In the first step, clustering is performed on rows and columns to obtain one-dimensional clusters, with a user-specified parameter K required to control the cluster numbers. Afterwards, ITWC selects a set of heterogeneous biclusters from the combinations as the *representative biclusters*. All generated biclusters are then sorted in descending order based on the cosine distances between them and the representative biclusters. Finally, the biclusters with higher scores are regarded as final output.

Block clustering developed by Hartigan [77] follows a divide-and-conquer approach. The algorithm starts with the entire data matrix as one large bicluster, called *block*. In each iteration, it searches for the best bipartition of the large block into two pieces that achieves the highest reduction of variances (see Section 2.2.2). The iteration stops when the total number of biclusters reaches a pre-defined value K.

Cheng and Church [30] applied biclustering analysis to gene expression analysis for the first time. The algorithm performs a greedy search on the data matrix and uses a scoring function to search for the local optima. This is a trade-off between the optimized solution and running time. Cheng and Church's algorithm [30] starts with an initial sub-matrix that contains all rows and columns. A specified threshold δ is used as the upper bound for the mean squared residue (See Section 2.2.2). In a greedy manner, the algorithm iteratively removes the rows or columns that could achieve the largest decrease in mean square residue, until the mean square residues of the remaining bicluster falls below δ. This deletion phase is followed by a row/column

addition phase where one row/column is added back in each iteration to achieve maximum sizes of the biclusters without exceeding the upper bound δ.

The FLOC [207, 208] algorithm uses a modified mean square residue definition presented in Section 2.2.2. Unlike Cheng and Church's method where only one bicluster is discovered on each run, FLOC applies a two-phase procedure to identify all biclusters in the solution simultaneously. In the first phase, K initial bicluster candidates are generated by a stochastic process that adds and deletes rows and columns from the entire matrix. In the second phase, the quality of the initial biclusters is improved iteratively. The algorithm stops when the overall mean square residue can no longer be reduced.

Iterative signature algorithm (ISA) is a nondeterministic algorithm that greedily searches biclusters with two major requirements [15]: (1) each row in a bicluster must have an average value larger than a certain threshold T_R; (2) likewise each column in a bicluster must have an average value larger than a certain threshold T_C. The algorithm starts with a random seed bicluster and iteratively updates the bicluster until convergence. The algorithm re-runs the iteration step with different seeds.

BiMax is a divide-and-conquer algorithm that seeks sub-matrices in the data matrix containing as many 1s as possible [149]. Thus BiMax only works on binary data sets. Therefore, before applying BiMax on the data sets, a common strategy is to select a threshold and set all the elements above the threshold to be 1, the others to be 0. Then BiMax recursively divides the matrix into smaller sub-matrices to optimize biclusters enriched with 1s.

In Factor analysis for bicluster acquisition (FABIA) [87] model, a bicluster is a subset of *similar* rows/columns. The multiplicative model is used to define similar vectors (a vector is a row/column), i.e., two vectors are similar to each other if the angle between them is zero. Such linear dependency can thus be formulated as the outer product of two vectors, λ and z, where λ refers to a baseline vector and z refers

to a vector of factors. Zeros in z indicates the part of the vector not present in the bicluster. The bicluster can thus be formulated as [87]:

$$\mathbf{X} = \sum_{i=1}^{p} \lambda_i \mathbf{z}_i^T + \boldsymbol{\varepsilon} = \Lambda Z + \boldsymbol{\varepsilon} \tag{2.33}$$

Variational expectation maximization is used to maximize the posterior of the model variables given the data matrix. Thresholds are given to decide the membership of columns and rows in each bicluster.

The plaid algorithm fits the data to an artificial model called plaid model [114] (See Section 2.2.2). In a plaid model, the value of a data element m_{ij} is affected by several effects: bicluster effect μ, i.e. the effect of all the biclusters within the data matrix; background effect θ; row effect α; column effect β and the random error e. X_{ij} is formulated as follows:

$$m_{ij} = \theta + \sum_{k=1}^{K} (\mu_k + \alpha_{ik} + \beta_{jk}) \rho_{ik} \kappa_{jk} + e_{ij} \tag{2.34}$$

k refers to the k^{th} bicluster in a the data set of K biclusters. The plaid algorithm iteratively fits the data to the plaid model and updates the parameters in the formula such that the minimum squared error between the model and the true data is minimized.

QUBIC is a deterministic algorithm that converts the problem of biclustering to finding dense subgraphs on a bipartite graph [117]. Bipartite graphs are generated based on the discrete data matrices. In the algorithm biclusters with non-zero constant columns are located. Then the matrices (graphs) are divided into down and up-regulated ranks. The biclusters are generated by iterative expansion of a seed edge. Afterwards, they are expanded based on the requirement on the values of the columns: the algorithm first requires all columns to be constant in the first iteration, then this requirement is relaxed to allow the addition of rows.

Spectral clustering uses singular value decomposition (SVD) to find a checkerboard pattern of biclusters in the data set such that each bicluster has a variance lower than a given threshold [105].

As mentioned in Section 2.2.2, Segal *et al.* [160, 161] used a Bayesian network and built a Probabilistic Relation Model (PRM) to measure the conditional and the joint distributions to infer the inter-relations between subsets of genes and conditions. The Bayesian structures are constructed using a Condition Probability Distribution-tree (CPD-tree)[57], followed by a greedy strategy combined with a simulated annealing procedure to avoid local optima. Afterwards the parameters in the distributions are computed by maximum likelihood. Segal *et al.* [160, 161] assumed multinomial distribution for discrete variables and Gaussian distribution for continuous variables. Then the algorithm is used to group the genes and the conditions such that the conditional probabilities of the gene expressions on experiment conditions are maximized.

Califano *et al.* [23] followed a pattern alignment strategy to discover the biclusters in the matrix, by modeling the column profiles as strings. A density constraint was applied to decrease the influence of random matching on large data sets. The approach searches a bicluster satisfying the δ-valid ks-pattern in a greedy manner, then expands the bicluster as much as possible without breaking the δ-valid ks-pattern rule. It starts with a bicluster with all columns, then tries to find a pattern match of all subsets satisfying the δ-valid rule. Afterwards, redundant biclusters are removed. The evaluation of the final biclusters, however, is not based on variance since the matrix is not normalized. A statistical test was applied to compute this statistical significance.

Tanay *et al.* [183] developed a Statistical-Algorithmic Method for Bicluster Analysis (SAMBA) based on exhaustive enumeration. SAMBA works on bipartite graphs converted from the matrix (See Section 2.2.1 for details). The edges within the graph represent the significant expression changes. Two models — one simpler model and

one refined model — are developed to search for biclusters with different confidence levels.

Table 2.1 shows a summary and comparison of the biclustering algorithms mentioned here.

Table 2.1: Summary and comparison of the biclustering algorithms.

Method Name	Bicluster Model	Strategy
Cheng and Church [30]	Constant & Constant Values	Divide and conquer
FLOC [207, 208]	Constant Values	Greedy
CTWC [64]	Coherent Values	Combined Clusters
ITWC [185]	Coherent Values	Combined Clusters
Spectral [105]	Coherent Values	Greedy
OPSM [12]	Coherent Evolution	Greedy
SAMBA [183]	Coherent Evolution	Exhaustive Enumeration
xMOTIFS [134]	Coherent Evolution	Greedy
ISA2 [15]	Coherent Values	Greedy
FABIA [87]	Coherent Values	Probabilistic Model
PRM [160, 161]	Constant Row/Column	Probabilistic Model
QUBIC [117]	Constant Row/Column	Divide and conquer
Plaid [115]	Coherent Values	Probabilistic Model
Bimax [149]	Constant Row/Column	Divide and conquer

2.2.5 Biclustering Structures

Biclustering algorithms often involve different assumptions with regard to the bicluster structures. Many algorithms like [30, 63, 64, 77, 114, 161, 185] assume multiple biclusters in the matrix while other models focus on finding just the best bicluster

[12, 82]. For example, the algorithms aiming at finding only one bicluster in the input matrix do not consider the interactions between biclusters. The search strategies also differ largely. See [124] for more details.

2.2.6 Application of Biclustering

Biological Applications

One of the most common applications of biclustering is gene expression data mining. The wide application of gene expression microarray technology enables establishing the connections between gene expressions and phenotypes. Such experiments linking the underlying genes and the phenotypes provide fundamental knowledge to functional genomics and transcriptomics, promoting the understanding in the mechanisms of gene regulations, the interactions between genes and pathways, the development and the evolution of individuals, and pathogenesis [9].

Table 2.2 (taken from [124]) shows a list of applications reported using biclustering techniques on microarrays.

Note that all previous studies using biclustering techniques were conducted using microarrays, yet the biclustering models are not limited to gene expression mining but also fit to be applied on any interesting issues with the underlying model of matrices or bipartite graphs. For example, Lazzorni *et al.* conducted biclustering analysis using biclustering on food and nutrition data to find food classes with similar nutrition facts [114].

Other applications

Biclustering can deal with data from heterogeneous sources. Such examples include not only expression studies or drug repositioning, but also research in the economic areas such as marketing or customer behavior. In the context of marketing, biclustering is used to find customers with similar behavior such as purchasing preferences,

service preferences or reading habits. A number of investigations were conducted, especially in E-commerce [88, 90, 189, 192, 207, 208].

In information retrieval, studies have been carried out on grouping documents based on the similarities of certain properties, such as key words, sentence structure, texts or images. In biclustering, the documents are modeled as rows and the specific properties (text, images, etc.) are modeled as columns [17, 45]. Similarly, the documents can also be modeled as bipartite graphs and the document clusters can be retrieved by performing graph biclustering [183].

Table 2.2: Taken from [124]. Applications of biclustering algorithms on gene expression analyses.

	Data Set	Applications	References
Yeast	Yeast Cell Cycle 1 [169]	Gene Functional Annotation & Coregulation Identification	[169] [184]
	Yeast Cell Cycle 2 [33]	Coregulation Identification	[30] [32] [120] [192] [207] [208]
	Yeast Stress 1 [63]	Gene Functional Annotation & Coregulation Identification	[160] [161] [184]
	Yeast Stress 2 [62]	Gene Functional Annotation	[184]
	Yeast Compendium [90]	Gene Functional Annotation	[160] [161] [184]

	Yeast Galactose Utilization [92]	Gene Functional Annotation	[184]
Human	Response of Fibroblasts to Serum [94]	Sample Classification	[187]
	Lymphoma Microarray (B-Cells) [1]	Coregulation Identification Sample Classification	[30] [32] [105] [134] [184]
	Lymphoma Affimetrix [103]	Sample Classification	[105]
	Leukemia Affimetrix I (All /AML) [67]	Sample Classification	[21] [105] [134]
	Leukemia Affimetrix 2 [8]	Sample Classification	[165]
	Colon cancer [3]	Sample Classification & Coregulation Identification	[64] [134]
	Multiple Sclerosis [212]	Sample Classification	[185]
	Cancer Cell Line Affimetrix [195]	Sample Classification	[23]
	Breast Cancer 1 [82]	Coregulation Identification	[12]

Breast Cancer 2 [104]	Sample Classification	[105]
CNS Embryonal Tumor [147]	Sample Classification	[105]

Chapter 3

Background: Bicluster Editing and Parameterized Tractability

One common strategy to solve biclustering algorithms is to convert the matrix into bipartite graphs, as shown in Fig. 2.1. Through such conversion, graph clustering approaches can be directly applied to biclustering. In this chapter, an important graph clustering model, namely bicluster editing is introduced. Bicluster editing is the basic model behind n-CluE and thereby of central importance for this thesis. In this chapter, the theoretical background of bicluster editing and the concept of parameterized tractability are presented, which provides the theoretical support to solve bicluster editing problem both exactly and heuristically.

3.1 Cluster Editing

The problem of data clustering has been a research focus in data mining for decades. Clustering on graphs, as one of the most important branches in the clustering, has been extensively studied and reviewed in [78, 83, 128, 168].

A common strategy for clustering is to choose a similarity threshold and to construct the corresponding graph according to the following rules: (1) create a node in

the graph for each entity/object, and (2) edges are drawn between two nodes if and only if they are labeled as "similar". Under such setting, we denote the end points u and v of an edge as "similar", written as $u \sim v$. The edge set of the similarity graph is the collection of all similar node pairs, $E = \{(u, v) | u \sim v\}$.

The graphs constructed from the above strategy are not necessarily "transitive", which means for arbitrary three nodes u,v and w, $u \sim v$ and $v \sim w$ does not necessarily imply $u \sim w$. We aim to convert the preliminarily constructed graph into a transitive graph only consisting of disjoint clusters with minimum number of edge deletions/insertions (for an unweighted input graph) or minimum sum of penalty (for a weighted input graph). In the context of graph theory, if the graph satisfies any of the equivalent conditions below, then we call it "transitive":

1. For any three vertices u, v and $w \in \binom{V}{3}$, we have $(u, v) \in E$ and $(v, w) \in E \Longrightarrow (u, w) \in E$.

2. An acyclic connected subgraph of 3 vertices does not exist, i.e., for each u, v and $w \in \binom{V}{3}$, we have $E \cap \{(u, v), (v, w), (u, w)\} \neq 2$.

3. G is a disjoint union of cliques (a clique is a complete graph).

Under this setting, the definition of the unweighted cluster editing problem can then be formulated as follows:

Unweighted Cluster Editing: Given a graph $G = (V, E)$, can G be converted into a transitive graph with a minimum number of edge insertions and deletions?

Cluster editing is one of the classic problems in theoretical computer science research and has been widely used in practical applications. However, fixing the edge weight to +1,-1 and 0 cannot accurately describe the similarities between different biological entities. Thus the cluster editing problem on graphs with real-numbered edge weights, namely *weighted cluster editing*, provides a better model for the biological applications. The setting of the weighted cluster editing problem is derived from

its unweighted version. Given an input graph $G = (V, E)$, where V is the set of nodes and E is the set of the edges. $\binom{V}{k}$ is denoted as the set of k-element subsets of V. $\{u, v\}$ is an unordered pair of $\{u, v\} \in \binom{V}{2}$. The similarity between two nodes is a symmetric function $s\binom{V}{2} \to \mathbb{R}$. Given a specific threshold δ, we call u and v similar, denoted as $u \sim v$, if and only if $s(u, v) > \delta$. The edge set is therefore defined as the set of all similar node pairs, $E = \{\{u, v\} | u \sim v\}$. Self-loops are not allowed in the graph. Our aim is to convert the input graph G into a transitive graph with edge insertions and deletions. Each insertion and deletion of $\{u, v\}$ incurs a certain cost dependent on the edge weight and the threshold. For edge deletion, the cost is defined as the edge weight minus the threshold, $cost(u, v) = s(u, v) - \delta$; for edge insertion, the cost is $cost(u, v) = \delta - s(u, v)$. The total cost of the conversion $cost(G \to G')$ is defined as: $cost(G \to G') = cost(E \backslash E') - cost(E' \backslash E)$. The weighted cluster editing problem can be formulated as follows:

Weighted Cluster Editing: Given a weighted input graph $G = (V, E)$ and a similarity function $s\binom{V}{2} \to \mathbb{R}$, try to convert the input graph G into a transitive graph G' with edge insertions and edge deletions such that the total conversion cost $cost(G \to G')$ is minimum.

Apparently the unweighted graph can be regarded as a special case of the weighted graph with a similarity function of $s\binom{V}{2} = \{-1, +1\}$ and the threshold is fixed to 0. The editing cost is defined as $cost(G \to G') = |E \backslash E'| + |E' \backslash E|$.

Both versions have been investigated and shown to be NP-complete by Shamir *et al.* [163].

3.2 Bicluster Editing

In the spectrum of the problem formulations for biclustering, the transformation of biclustering on matrices to the clustering on bipartite graphs is one of the most

important directions to solve the problem from a graph theory perspective. A matrix M_{IJ} can be easily converted into a bipartite graph where for each row $r_i \in I$ and each column $c_j \in J$, a corresponding node is constructed. The edge weight can be formulated in different ways, depending on the application cases. Usually, one models edge weights as the similarities between the nodes based on some specific criterion.

After the graph is constructed, different models can be applied to solve the biclustering problem. Other than the bicluster editing models, a number of other models on bipartite graphs have contributed in this direction.

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

A biclique is a complete bipartite graph. The problem of maximum vertex weighted biclique has been solved in polynomial time by the algorithm proposed by Yanakakis [210].

Exact Cardinality Biclique: Given a bipartite graph $G = (U, V, E)$ and two positive integers k and l, ask whether there exists a biclique $B = (U', V', E')$, such that $B \subseteq G$, $|U'| = k$ and $|V'| = l$.

Maximum Balanced Vertex Cardinality Biclique: Given a bipartite graph $G = (U, V, E)$ and a positive integer k, ask whether there exists a biclique $B = (U', V', E')$, such that $B \subseteq G$, $|U'| = |V'| \geq k$.

Dawande *et al.* [41, 61] have proven that both of the above two problems are NP-complete.

Maximum Edge Cardinality Biclique: Given a bipartite graph $G = (U, V, E)$, try to find a biclique (U', V', E') in G such that $|E'|$ is maximal.

The maximum edge cardinality biclique has been shown to be NP-complete by Peeters *et al* [145].

Minimum Edge Deletion Biclique: Given a bipartite graph $G = (U, V, E)$, find a biclique (U', V', E') in G such that $|E \backslash E'|$ is minimal.

Minimum Edge Deletion Weighted Biclique: Given a weighted bipartite graph $G = (U, V, E)$ and a similarity function s: $s(e) \to \mathbb{R}^+$ $(e \in E)$, find a biclique (U', V', E') in G such that $\sum_{e \in E'} s(e)$ is minimal.

Dawanade *et al.* have proven both the weighted and unweighted version of minimum edge deletion biclique to be NP-complete [42]. Hochbaum has provided an approximation algorithm with worst-case approximation factor 2 [85].

In this thesis the model of bicluster editing is adopted to solve the biclustering problem. The bicluster editing problem is the counterpart of cluster editing on bipartite graphs.

The bipartite graphs are constructed following the same strategy: (1) for each entity/object we construct a node in the graph; (2) edges are inserted between the "similar" node pairs. Two endpoints of an edge u and v is thus called "similar", written as $u \sim v$.

The resulting bipartite graph is not always "transitive". For a bipartite graph $G = (U, V, E)$, we have the following equivalent conditions characterizing if it is transitive:

1. For each subset of four nodes, $\{u, v, w, x\} \in \binom{V}{4}$, where $\{u, w\} \in \binom{U}{2}$, $(v, w) \in \binom{V}{2}$, we have $\{u, v\} \in E$, $\{w, v\} \in E$ and $\{w, x\} \in E \Longrightarrow \{u, x\} \in E$.

2. G does not contain an acyclic connected subgraph of four nodes, i.e., for each $\{u, v, w, x\} \in \binom{V}{4}$, where $\{u, w\} \in \binom{U}{2}$, $\{v, w\} \in \binom{V}{2}$, we have $|E \cap \{\{u, v\}, \{w, v\}, \{w, x\}, \{u, x\}\}| \neq 3$.

3. G is a disjoint union of bicliques (i.e. complete bipartite graphs).

The aim of bicluster editing is to convert the given input bipartite graph (weighted or unweighted) to transitive bipartite graphs with minimum editing cost. The definitions of edge weights and editing costs remain the same as those in cluster editing.

The unweighted bipartite graph is a special case of the weighted bipartite graph where the edge weights of all connected node pairs are assigned to +1 and the weights of all non-connected node pairs are assigned to -1. The threshold in the bipartite graph is fixed to 0.

The bicluster editing problem can be formulated as:

Unweighted Bicluster Editing: Given an unweighted bipartite graph $G = (U, V, E)$ where U and V are the nodes sets and E is the edge set, can G be converted into a transitive bipartite graph with minimal number of edge insertions and edge deletions?

Weighted Bicluster Editing: Given a weight bipartite graph $G = (U, V, E)$ and a similarity function $s\binom{V}{2} \to \mathbb{R}$, convert the input graph G into transitive bipartite graph G' with edge insertions and edge deletions such that the conversion cost $cost(G \to G')$ is minimal.

The bicluster editing model tries to capture the exclusive row and column biclusters, since no overlapping between different bicliques are allowed.

3.3 NP-hardness

The NP-completeness of the unweighted bicluster editing problem was proven by [5] by a polynomial reduction from the classic NP-complete problem of the 3-Exact 3-Cover problem, formulated as:

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

Apparently, the weighted version of bicluster editing is also NP-complete by a simple reduction from the unweighted bicluster editing problem.

3.4 Parameterized Tractability

Both unweighted and weighted bicluster editing problems have been proven to be NP-complete. To overcome the challenge of exponential explosion (i.e., running times grow exponentially or worse with the sizes of the input instance) caused by NP-hardness, research efforts have been pivoted to parameterzied complexity theory and fixed-parameter algorithms. The parameterized complexity theory provides a strategy to solve the NP-hard problem by restricting as much as possible the exponential growth in running times [137]. The main theory of parameterized tractability strives for the insight of a specific problem by investigating the influences on time complexity of the parameters closely related to the given problem [137]. By restricting some critical parameters to small values, the original problem might be able to be solved efficiently. Consider the well-known graph problem of vertex cover:

Vertex cover: Given an input graph $G = (V, E)$, try to find a subset of vertex V' with minimum size, such that for each edge $e \in E$, e has at least one end vertex in V'.

For the vertex cover problem, if the size of the final cover, as the critical parameter, is kept as a small value, then the solution can be found with a relatively small exponential factor, unrelated to the size of the input graph. One of such solutions can be found in [138] with the exponential factor as 1.29^k, where k is the fixed parameter of the size of the vertex cover.

Following the strategy of fixed-parameter tractability, the NP-hard problem might be solved with a running time of $O(f(k)\dot{|}I|^c)$, where f is a function that solely depends on a parameter k, usually exponentially or worse; $|I|$ is the input size of the problem and c is a non-negative constant. A recent review of fixed-parameter tractability can be found in [175]. Notably, not all NP-hard problems can be solved using fixed-parameter techniques. For instance, the running time of the NP-hard problem of

Dominating Set cannot be reduced using an analogous strategy to fixed-parameter tractability, as explained in [47].

Kernelization is an essential pre-processing step in the development of a fixed-parameter algorithm. By proper reduction rules, the sizes of the input problems can be further cut down, yielding a reduced problem instance, namely the problem kernel. The goal of a kernelization is to design a function of f, such that an input instance with the size I and parameter k is transformed into a new problem instance I' and new parameter k' with $k' \leq k$ and $|I'| \leq g(k)$, where g is a function only dependent on k. The original problem instance (I, k) has a solution if and only if the reduced instance (I', k') has a solution, see [47] for the formal mathematical framework of the fixed-parameter tractability and kernelization.

Investigations in the direction of parameterized tractability of cluster editing and bicluster editing turned out to be a fruitful research area. A series of different algorithms have been developed for the unweighted cluster editing problem with sophisticated kernelizations and refined branching strategies [47, 56, 69]. The latest kernelization pipeline has a result of only $4k$ vertices in the reduced instance [71]. Gram *et al.* [44, 70] have proposed an algorithm with time complexity of $O(2.27^k + |V|^3)$ where V is the vertex set in the given input graph. Böcker *et al.* further refined the branching strategy and achieved a lower running time bound by $O(1.83^k + |V|^3)$ [18]. In the same study of Böcker *et al.* [18], the authors contributed also to the weighted version of cluster editing, and achieved a problem kernel of $k^2 + 3k + 2$ vertices and $\frac{1}{2}k^3 + \frac{5}{2}k^2 + 5k + 2$ edges.

For bicluster editing, Protti *et al.* [150] proved that the running times can be bound by $O(4^k + |E|)$ by a simple branching strategy, where E stands for the edge set in the input graph. A problem kernel consisting of $4k^2 + 6k$ vertices was also proposed in Protti *et al.*'s report [150]. Guo *et al.* [72] further improved the kernelization for bicluster editing to a linear problem kernel of $4k$ vertices, using a similar strategy

applied previosly on cluster editing problems [71]. The running time was thereby cut down to $O(3.24^k + |E|)$. The weighted version of bicluster editing has not, however, been fully investigated, which is one of the focuses of this thesis. We followed a similar strategy as in Böcker *et al.* [18] and brought forward a new kernelization and branching strategy for weighted bicluster editing.

3.5 Fixed-Parameter Algorithm for Cluster Editing

As mentioned in the previous section, fixed-parameter algorithms are a special class of algorithms based on fixed-parameter tractability, aiming to solve NP-hard problem with higher efficiency. According to the rules of "transitivity" explained in Section 3.1, no P3 (a path of 3 nodes) is allowed in the result graph. The key to solve cluster editing problems is to "repair" all existing P3s into disjoint cliques. In order to achieve this goal, for each P3 found in the input graph, we have the following three possibilities:

Given a graph $G = (V, E)$, suppose three nodes $\{u, v, w\}$ form a P3, where $\{\{u, v\}, \{v, w)\}\} \subseteq E$, we can choose one of the following editing actions to repair the P3.

- Remove the edge $\{u, v\}$.
- Remove the edge $\{v, w\}$.
- Insert the missing edge $\{u, w\}$.

Fig. 3.1 illustrates the three editing possibilities. The most straightforward way to solve the cluster editing problem, namely the *brute force branching strategy* starts with randomly selecting one P3 found in the input graph and recursively tries the three editing possibilities. After each edge insertion/deletion, the algorithm checks the whole graph for un-repaired P3s. If there exists any unfixed P3, the algorithm

then proceeds to repair the newly found P3; if there is no P3 left, then the current graph is recorded as one solution and the algorithm backtracks. In this manner, the algorithm searches the complete solution space and picks up the final solution with minimal editing cost. The whole algorithm finishes within $O(3^n)$. Fig. 3.2 illustrates two different solutions.

Apparently, the brute force branching strategy is inefficient in processing large input instances. The width of the branching tree grows exponentially with the increasing input sizes and leads to an exponential growth of running times. Applying the parameterized theory on cluster editing changes the landscape of the problem. As explained in the beginning of this section, the time complexity of a given problem is related not only to the input sizes but also to the parameterized solution size. Parameterized theory is set to focus on the cases when the parameter is relatively small. For unweighted cluster editing, if we choose the sum of the minimum number of required edge insertions and deletions as the parameter k, then the selected parameter k can be significantly smaller than the whole input size n. This might greatly reduced the required running time. In the context of parameterized theory, cluster editing can be re-formulated as:

Unweighted Cluster Editing (Reformulated): Given a graph $G = (V, E)$ and a nonnegative integer k, can G be converted into a transitive graph with no more than k edge modifications?

Weighted Cluster Editing (Reformulated): Given a graph $G = (V, E)$ and a nonnegative real number k, can G be converted in to a transitive graph with cost no more than k?

It is easy to show that the time complexity of the brute force strategy is $O(3^k)$. With a data reduction procedure (kernelization) and a refined branching strategy, Gramm *et al.* [70] achieve a time complexity of $O(2.27^k + n^3)$.

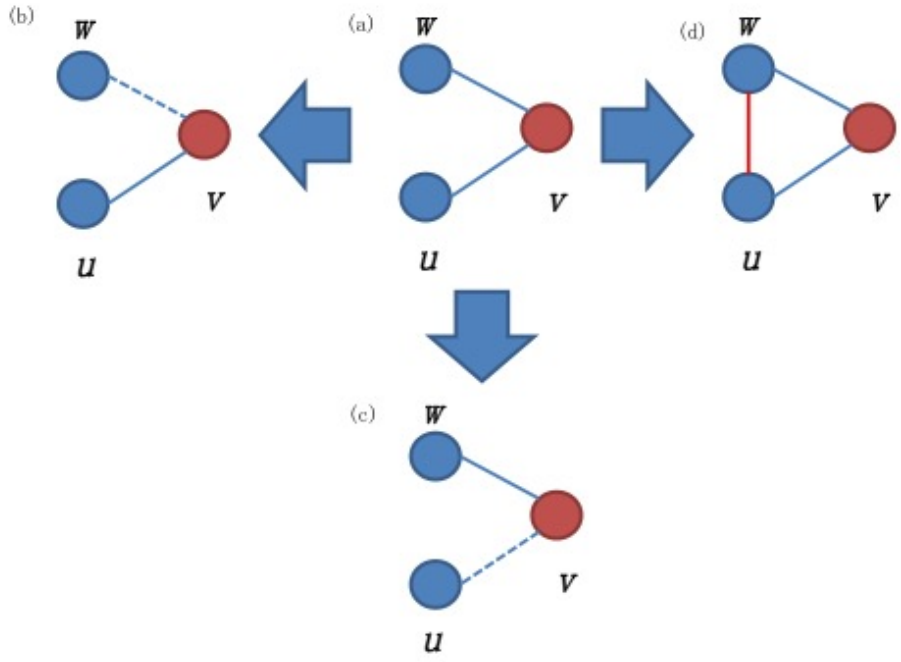

Figure 3.1: The three editing possibilities for a given P3. (a) The given P3 consisting of nodes u, v and w. (b) Repairing of P3 by the deletion of the edge $\{w, v\}$ (c) Repairing by deletion of the edge $\{u, v\}$. (d) Repairing by the insertion of $\{w, u\}$.

The fixed-parameter strategy for bicluster editing, which is part of the main contributions of this thesis, will be thoroughly demonstrated and discussed in Chapter 5

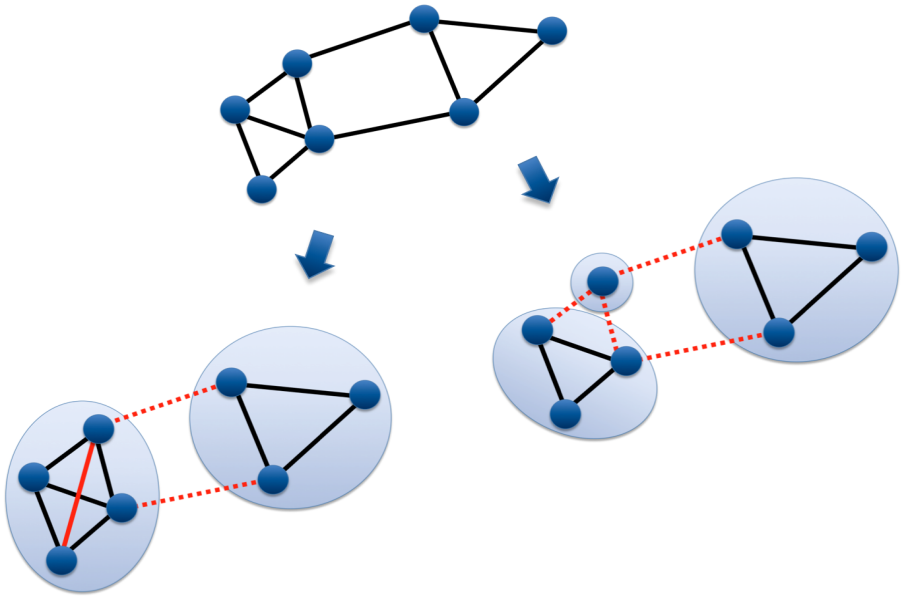

Figure 3.2: An example of solving the cluster editing problem. The upper part is the input instance. The lower-left and the lower-right part are two possible solutions.

Chapter 4

Complexity of The Dense Bicluster Editing Problem

Before we proceed to our implementation for solving practical problems, it is beneficial to first present my theoretical research regarding the classic bicluster editing problem and some of its useful variants. In this chapter, we focus on Π-bicluster editing, which is by definition a relaxed version of the bicluster editing problem. The theoretical study here investigates the possibility to solve efficiently the variant of bicluster editing, namely Π-Bicluster Editing, given the fact that the classic bicluster editing problem is NP-hard.

The content of this chapter is based on the published research article listed below:

- Peng Sun, Jiong Guo, and Jan Baumbach. Complexity of dense bicluster editing problems. *Computing and Combinatorics*, pages 154–165, 2014

4.1 Preliminaries

An undirected graph $G = (U, V, E)$, where U and V are two sets of nodes and E is the set of edges, is a *bipartite graph* if $\forall e \in E$, edge e has exactly one end vertex

in U and the other end vertex in V. Let $W = U \cup V$. For an arbitrary $W' \subseteq W$, the *induced subgraph* $G[W']$ is the subgraph over the vertex set W' with the edge set $\{\{u,v\} \in E | u,v \in W'\}$. An induced subgraph $G[W'] = (U', V', E')$ is a biclique if $\forall u \in U'$ and $\forall v \in V'$, we have $\{u,v\} \in E$. The *open neighborhood* $N(v)$ of $v \in W$ is the set of vertices that are adjacent to v in G. The *degree* of a given vertex v is denoted by $d(v)$, referring to the cardinality of $N(v)$. The *closed neighborhood* of v is denoted by $N[v]$, i.e., $N[v] = N(v) \cup \{v\}$. The open and closed neighborhoods of a set of vertices $W' \subseteq W$ are defined as $N(W') = \bigcup_{u \in W'} N(u) \backslash W'$ and $N[W'] = N(W') \cup W'$, respectively. Let $W' \subseteq W$, we use $G - W'$ as the abbreviation for $G[W \backslash W']$ and for a vertex $v \in W$, let $G - v$ denote $G - \{v\}$. If $G - v$ has more connected components than G, then we call v as a *cut vertex*. Similarly, let E' be a set of edges, then $G - E'$ denotes the graph $G' = (U, V, E \backslash E')$. For a graph $G = (U, V, E)$, denote $\overline{E} = \{\{u,v\} | u \in U \wedge v \in V \wedge \{u,v\} \notin E\}$ as the set of *missing edges*. A pair of vertices $\{u,v\}$ is called a *missing edge* if $\{u,v\} \in \overline{E}$. For two sets of vertices X and Y, let $E(X,Y)$ be the set of edges between X and Y, i.e., $E(X,Y) = \{\{u,v\} \mid u \in X \ \wedge \ v \in Y \ \wedge \ \{u,v\} \in E\}$. For a vertex set X, denote $E(X)$ as the abbreviation for $E(X,X)$. For a set of vertex X' and a bipartite graph $H = (X, Y, E)$, denote the intersection between X' and H as the set of common vertices, i.e., $X' \cap H = (X' \cap X) \cup (X' \cap Y)$.

A problem is *fixed-parameter tractable* (FPT) with respect to a certain parameter k, if there is an algorithm that decides the problem in $f(k) \cdot n^{O(1)}$ time. Here, n denotes the size of the input and f is a computable function. The framework of *fixed-parameter tractability* was developed by Downey and Fellows [55]. A crucial tool in the development of fixed-parameter algorithms is polynomial-time preprocessing data reduction. Its goal is for a given instance x with parameter k, to transform the problem into a new instance x' with parameter k', such that the size of the new instance x' is upper-bounded by some function only depending on k. The instance

(x, k) is a yes-instance if and only if (x', k') is a yes-instance and $k' \leq k$. The reduced instance must be computable in polynomial time. The whole data reduction process is called *reduction to a problem kernel* or *kernelization* and the reduced instance is called *problem kernel*.

4.2 Introduction

In graph-based data clustering methodologies, data entities are modeled as vertices and a certain function is defined to quantify the "relationship" between two vertices (e.g. similarities). The clustering problem could also be viewed from the graph modification angle, i.e., to modify the graph by edge insertions and deletions into a Π-cluster graph, the so-called Π-cluster editing problems. Here, a graph is a Π-cluster graph, if each of its connected components satisfies Π, where Π is a certain density measure. The most famous problem among the Π-cluster editing is cluster editing, where Π is "being a clique". cluster editing has been extensively studied and proved as NP-complete among the earliest NP-complete problems [10, 163].

In some real-world applications, "being a clique" is increasingly criticized as over-restrictive [162]. Thus some relaxed models might be more advantageous in a variety of application scenarios. Theoretical studies have also been conducted on relaxed versions of cluster editing. Guo *et al.*[74] studied the fixed-parameter tractability of s-plex editing. In another study, Guo *et al.* extended their research further to several other relaxed models: s-defective cliques, average-s-plexes and μ-cliques [73], in which the NP-completeness and the fixed-parameter tractability are proved.

Similarly to Π-cluster editing, an s-biplex is a connected bipartite graph $G = (U, V, E)$ with $d(u) \geq |V| - s$ for all $u \in U$ and $d(v) \geq |U| - s$ for all $v \in V$. Note that a normal biclique is thus a 0-biplex. A bipartite graph G is called an *s-biplex cluster graph* if all its connected components are s-biplexes. Therefore, s-biplex editing is the

special case of bicluster editing with Π equal to "s-biplex". Here, the NP-completeness of s-biplex editing is shown. Then the sizes of *minimal forbidden induced subgraphs* are upper-bounded by $O(s)$ and a branching strategy can be derived which indicates the fixed-parameter tractability of s-biplex editing.

In general graphs, average-s-plex is proposed as a "density measure", defined as the mean of the degrees of all vertices in a given graph [74]. In a bipartite graph, we define the *average degree* for two vertex sets separately: $\overline{d}_U = |E|/|U|$ and $\overline{d}_V = |E|/|V|$. A connected graph $G = (U, V, E)$ is thus an *average-s-biplex* if $\overline{d}_U \geq |V| - s$ and $\overline{d}_V \geq |U| - s$, with $1 \leq s \leq \min\{|U|, |V|\}$. This density measure can be considered as a further relaxation of s-biplex, with no requirement on the *minimum* degree. In this work, we show the NP-completeness of average-s-biplex editing. Afterwards, a reduction to a more general problem is conducted, followed by a polynomial-time kernelization procedure which produces a graph with at most $2k((s+1)(4k+6s)+1)$ vertices. This implies fixed-parameter tractability for average-s-biplex editing.

The concept of *defective clique* has been reported previously to be useful in biological network analysis [213]. NP-completeness and fixed-parameter tractability of s-defective clique editing and deletions are already known [73]. A connected bipartite graph $G = (U, V, E)$ is an s-defective biclique if $|E| \geq |U| \cdot |V| - s$. We prove that s-defective bicluster editing is NP-complete. Then, the sizes of minimal fobidden induced subgraphs of s-defective bicluster graphs are shown to be bounded by $2s+3$, which leads directly to the fixed-parameter tractability of s-defective bicluster editing with respect to the parameter (s, k). For more information on parameterized complexity, we refer to [55] and [137]. Due to limited space, some proofs are deferred to Appendix A.1.

4.3 s-Biplexes

4.3.1 NP-Completeness

In this section, we show the NP-completeness of s-biplex editing by a reduction from 3-exact-3-cover.

Theorem 4.1. *For every constant $s \geq 0$, s-biplex editing is NP-complete.*

Proof. If s=0, then the problem is equivalent to bicluster editing and thus is NP-complete. For any $s \geq 1$, we reduce the NP-complete 3-exact-3-cover (3X3C), where given a collection $\mathcal{C}$ of triplets (a set of 3 elements is called a triplet) from an element set $A = \{a_1, a_2, a_3, ..., a_{3n}\}$ such that each element of A is a member of at most three triplets, one asks to find out a sub-collection $\mathcal{I} \subseteq \mathcal{C}$ of size n that covers A, i.e., every element of A appears in some triplet in $\mathcal{I}$. The set $\mathcal{I}$ is called an "exact cover".

We construct an s-biplex editing instance as follows: Let $m = (72 + s)n$. A bipartite graph $G = (U, V, E)$ is then constructed, based on the following procedure: For each element in A, one corresponding vertex is created in U, and for each triplet $S \in \mathcal{C}$, a set of m vertices is added to U. The same construction is performed to create vertices in V, that is: $U = U_1 \cup U_2$, $V = V_1 \cup V_2$, $U_1 = \{u_1, u_2, ..., u_{3n}\}$, $V_1 = \{v_1, v_2, ..., v_{3n}\}$, $U_2 = \bigcup_{S \in \mathcal{C}} \{u_{S_1}, u_{S_2}, ..., u_{S_m}\}$, $V_2 = \bigcup_{S \in \mathcal{C}} \{v_{S_1}, v_{S_2}, ..., v_{S_m}\}$.

The edge set E in G consists of five subsets: First, we connect every $u_i \in U_1$ to its corresponding $v_i \in V_1$, $1 \leq i \leq 3n$. Second, for each triplet $S \in \mathcal{C}$, let $S = \{a_x, a_y, a_z\}$, $(1 \leq x, y, z \leq 3n)$. We connect $u_i \in U_1$ and $v_j \in V_1$ for all $i, j \in \{x, y, z\}$ with $i \neq j$. Third, between U_2 and V_2, for each $S \in \mathcal{C}$, denote $U_S^m = \{u_{S_1}, u_{S_2}, , ..., u_{S_m}\}$ and $V_S^m = \{v_{S_1}, v_{S_2}, ..., v_{S_m}\}$. We connect $u_{S_i} \in U_S^m$ to $v_{S_j} \in V_S^m$ for all $1 \leq i, j \leq m$. Finally, for each $S = \{a_x, a_y, a_z\} \in \mathcal{C}$, $(1 \leq x, y, z \leq 3n)$, every $u_i \in U_1 (i \in \{x, y, z\})$ is connected to all vertices in $V_S^m \subseteq V_2$, and every $v_i \in V_1$ $(i \in \{x, y, z\})$ is connected to all vertices in $U_S^m \subseteq U_2$. More precisely: $E = \bigcup_{i=1}^{5} E_i$, $E_1 = \{\{u_i, v_i\} |\ i = 1, ..., 3n\}$, $E_2 = \{\{u_i, v_j\} |\ \exists S = \{a_x, a_y, a_z\} \in \mathcal{C} \ \wedge \ i, j \in \{x, y, z\} \ \wedge \ i \neq j\}$,

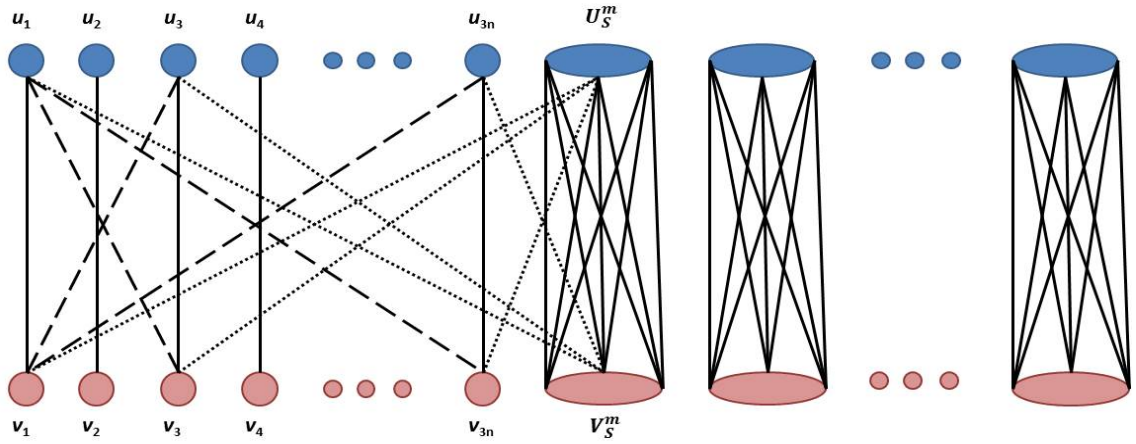

Figure 4.1: The construction of the reduction. u_1, u_3 and u_{3n} belong to the same triplet in $\mathcal{C}$; v_1, v_3 and v_{3n} belong to the same triplet in $\mathcal{C}$. Each U_S^m and V_S^m has $m = (72 + s)n$ vertices.

$E_3 = \{\{u_{S_i}, v_{S_j}\}|\ \exists S \in \mathcal{C} \ \wedge\ u_{S_i} \in U_S^m \ \wedge\ v_{S_j} \in V_S^m\}$, $E_4 = \{\{u_i, v_{S_j}\}|\ \exists S = \{a_x, a_y, a_z\} \in \mathcal{C} \ \wedge\ i \in \{x, y, z\} \ \wedge\ v_{S_j} \in V_S^m\}$, $E_5 = \{\{v_i, u_{S_j}\}|\ \exists S = \{a_x, a_y, a_z\} \in \mathcal{C} \ \wedge\ i \in \{x, y, z\} \ \wedge\ u_{S_j} \in U_S^m\}$.

For each triplet set $S \in \mathcal{C}$, we denote: $U_S = \{u_x, u_y, u_z|\{a_x, a_y, a_z\} \in S\}$, $V_S = \{v_x, v_y, v_z|\{a_x, a_y, a_z\} \in S\}$, $W_S = U_S \cup V_S$, $U_S^m = \{u_{S_1}, ..., u_{S_m}\}$, $V_S^m = \{v_{S_1}, ..., v_{S_m}\}$, $W_S^m = U_S^m \cup V_S^m$.

Obviously, the construction can be carried out in polynomial time. Let $M = 2m(3|\mathcal{C}| - 3n)$ and $N = |E_2| - 6n$. The parameter k is equal to $M + N$. For the rest of the proof, please refer to Appendix A.1. □

4.3.2 Forbidden Induced Subgraphs

In this section, we describe a set of *forbidden induced subgraphs* $\mathcal{G}_F$. A graph G is an s-biplex cluster graph if and only if G does not contain any forbidden induced subgraphs in $\mathcal{G}_F$. If $s = 0$, then we have a bicluster editing problem and the forbidden subgrah is a path of four vertices. If $s \geq 1$, the structures of forbidden induced subgraphs are much more complex and we are faced with an exponentially increasing

number of different possibilities. To solve the problem, we show in the following that the numbers of vertices in forbidden induced graphs are bounded by $O(s)$ vertices. Based on this characterization, a branching strategy solving s-biplex editing can be established.

We start with some preliminaries. A connected induced subgraph in G, $H = (R, T, E')$ is *minimal forbidden induced subgraph* if H is not an s-biplex cluster graph but every induced proper subgraph of H is an s-biplex cluster graph. We call a vertex v in H "$forbidden$" if v is incident to more than s missing edges (we call u incident to a missing edge to v if $\{u, v\} \notin E$). A subset of vertices R' is called "*forbidden subset*" if R' contains at least one forbidden vertex. To show the upper-bound for minimal forbidden induced subgraphs, two distinct cases are studied separately: (1) subgraph H contains forbidden vertex (vertices) only in R (or T), and (2) H contains forbidden vertices in both R and T. We first prove four claims regarding the properties of a minimal forbidden induced subgraph. Next, we show that every minimal forbidden induced subgraph of biplexes contains at most $3s + 3$ vertices in both the two cases mentioned above, for all $s \geq 1$.

Lemma 4.2. *Let $H = (R, T, E')$ be a minimal forbidden induced subgraph. If R is a forbidden subset, then $\min_{u \in R}\{d(u)\} = |T| - s - 1$*

Proof. See the proof in Appendix A.1.2. □

Lemma 4.3. *Let $H = (R, T, E')$ be a minimal forbidden induced subgraph. If H has forbidden vertices both in R and T, then H can only be a path of length $2s + 3$, and only the two endpoints of the path are forbidden vertices.*

Proof. First we prove the claim that H contains no more than two forbidden vertices if H has forbidden vertices both in R and T. Conversely, we assume that there are > 2 forbidden vertices in H, since we know that in every graph, there are at least 2

non-cut vertices. Let u, v be the 2 non-cut vertices. We have 6 cases with respect to u and v:

Case i: u, v are both forbidden vertices, and $u, v \in R$. Then we can remove u without separating H. In the subgraph $H - u$, we have $d_{H-u}(v) = |T| - s - 1$ and $H - u$ is also forbidden, contradicting with minimal forbidden induced subgraph. If $u, v \in T$, same proof applies.

Case ii: u, v are both non-forbidden vertices, and $u, v \in R$. Since R, T are both forbidden subsets, there exists a forbidden vertex $w \in R$, such that $d(w) = |T| - s - 1$. The subgraph $H - u$ is also forbidden since $d_{H-u}(w) = |T| - s - 1$. If $u, v \in T$, the same proof applies.

Case iii: u is a forbidden vertex, v is a non-forbidden vertex and $u, v \in R$. Then just remove v and $H - v$ is still forbidden.

Case iv: u is a forbidden vertex in R, v is a non-forbidden vertex in T. Since R, T are both forbidden subsets, there exists a vertex $w \in T$, such that $d(w) = |R| - s - 1$. We remove v from T. Then $d_{H-v}(w) = |R| - s - 1$ and thus $H - v$ is still forbidden.

Case v: u is a non-forbidden vertex in U and v is a non-forbidden vertex in V. A proof similar to Case iv applies.

Case vi: u is a forbidden vertex in R, v is a forbidden vertex in T. Let w be a forbidden vertex in H and $w \neq u$, $w \neq v$. Without loss of generality, we assume $w \in R$. Then we can remove u from H. Then $d_{H-v}(w) = |T| - s - 1$ and thus $H - v$ is still forbidden.

To summarize the six cases, since we have two non-cut vertices and at least three forbidden vertices with at least one forbidden vertex in R and at least one in T, we can always find a forbidden vertex x and a non-cut vertex y in the same vertex set (in R or in T). Clearly, removing y does not affect the property of the forbidden vertex x and thus the subgraph $H - y$ is still forbidden. This contradicts the assumption.

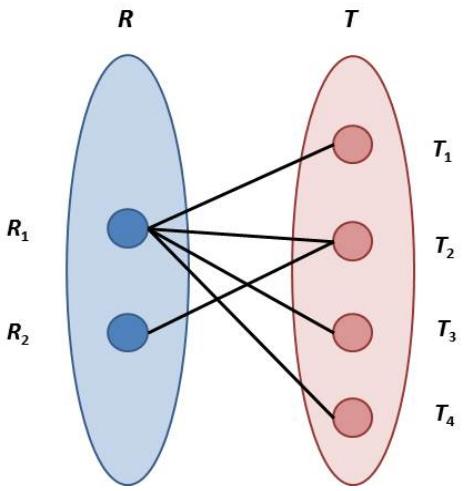

Figure 4.2: Example of a minimal forbidden induced subgraph with $s = 2$. The vertex R_2 has no edge between itself and T_1, T_3 and T_4 (R_2 is incident to three missing edges). Thus the subgraph is forbidden. However, any induced subgraph of it is a s-biplex.

Hence, if H is a minimal forbidden induced subgraph with forbidden vertices in both R and T, then H cannot contain more than two forbidden vertices.

Next, we prove that H can only be a path of $2s+3$ vertices. Let u^* and v^* be the two forbidden vertices in H. Suppose $u^* \in R$ and $v^* \in T$. Consider a third vertex w^*, $w^* \neq u^*$ and $w^* \neq v^*$. Clearly, such vertex w^* exists. If w^* is a non-cut vertex, then consider the subgraph $H - w^*$. If $w^* \in R$, in $H - w^*$, u^* is still a forbidden vertex; if $w^* \in T$, then in $H - w^*$, v^* is still a forbidden vertex. In either case, H is not minimal. Therefore, we know that in H, all vertices other than u^* and v^* must be cut vertices. Thus in H, we have $|R| + |T| - 2$ cut vertices. Obviously, H can only be a path and u^*, v^* can only be the two endpoints of the path in R and T. □

Lemma 4.4. *Let $H = (R, T, E')$ be a minimal forbidden induced subgraph with forbidden vertices only in R. Let $R_0 \subseteq R$ be the subset of all forbidden vertices and $R_1 = R \backslash R_0$. Let $T_0 = N(R_0)$ and $T_1 = T \backslash T_0$. Then we have:*

1. *$\forall u \in R_1$, u is a cut vertex.*

2. *$\forall v \in T_0$, v is a cut vertex.*

3. *If* $|R_0| > 1$, *then* $\forall u \in R_0$, u *is a cut vertex.*

4. *If* $|T_0| > 1$, *then for an arbitrary vertex* $v^* \in T_0$, *let* $\mathcal{H} = \{H_1, H_2, ..., H_l\}$ *be the set of disjoint components after removing* v^*. *Then for each* $H_i = (X_i, Y_i, E_i)$, $1 \leq i \leq l$, *we have* $X_i \cap R_1 \neq \emptyset$.

5. *There exists at least one vertex* $v \in T_1$ *with* $d(v) = 1$.

Proof. See the proof in Appendix A.1.3. □

Lemma 4.5. *Let* $H = (R, T, E')$ *be a minimal forbidden induced subgraph with forbidden vertices only in* R. *Let* $R_0 \subseteq R$ *be the subset of all forbidden vertices,* $R_1 = R \backslash R_0$. *Let* $T_0 = N(R_0)$ *and* $T_1 = T \backslash T_0$. *Then we have* $|R| + |T| \leq 3s + 3$.

Proof. Consider an arbitrary vertex $v^* \in T_0$. By Lemma 3, v^* is a cut vertex. Let $\mathcal{H} = \{H_1, H_2, ..., H_r\}$, $r > 1$, $H_i = (X_i, Y_i, E_i)$ be the set of disjoint connected components after removing v^*. Without loss of generality, let $\{H_1, H_2, ..., H_l\}$ be the subset of $\mathcal{H}$, $l \leq r$, such that $X_i \cap R_0 \neq \emptyset$, for all $1 \leq i \leq l$. We have the following two cases:

Case i. If $l \geq 2$, we know there is at least two disjoint components that intersect with R_0. Hence consider $\forall u \in X_1$ and $\forall v \in Y_j$ $(1 < j \leq l)$, we have $\{u, v\} \notin E'$. Similarly, we have $\{u', v'\} \notin E'$, for all $u' \in X_j$ $(1 < j \leq l)$ and all $v' \in Y_1$. Thus, we have $|Y_1| \leq s + 1$ and $\sum_{j=2}^{l} |Y_j| \leq s + 1$, since otherwise we would have a $u \in R$ incident to more than $s + 1$ missing edges, contradicting with Lemma 1. Because the number of missing edges incident to any vertex in R cannot be larger than $s + 1$, we have $|Y_1| + |T_1| \leq s + 1$ and $\sum_{j=2}^{l} |Y_j| + |T_1| \leq s + 1$. Thus we have $|T| = |Y_1| + \sum_{j=2}^{l} |Y_j| + |T_1| \leq s + 1 + s + 1 = 2s + 2$. Moreover, since we know $\min_{w \in T} d(w) = 1$ and T does not contain forbidden vertex, we have $|R| \leq s+1$. Thus the total size of the forbidden induced subgraph H is $|R| + |T| \leq 2s + 2 + s + 1 = 3s + 3$.

Case ii. if $\forall v \in T_0$, we have $l = 1$, then for all $v \in T_0$, the removal of v will not separate R_0. For an arbitrary vertex v^* in T_0, let $\mathcal{H} = \{H_1, H_2, ...H_r\}$, $H_i = (X_i, Y_i, E_i)$ be the disjoint connected components after removing v^*. Without loss of generality, let $R_0 \subseteq X_1$. Then $T_0 \subseteq Y_1$ and we can find at least one u^* with $u^* \in (N(v^*) \cap R_1)$, such that $u^* \notin N(v')$ for $v' \in T_0$, $v' \neq v^*$. Therefore, each vertex v^* in T_0 has at least one "unique" neighbor in R_1. Thus $|T_0| \leq |R_1| \leq s+1-|R_0| \leq s$. Moreover, we have $|T_1| \leq s+1$, since otherwise the vertices in R_0 are incident to more than $s+1$ missing edges. Then the total size of the forbidden induced subgraph H is $|R| + |T| \leq s+1+s+s+1 \leq 3s+2$. In summary, the claim is proved. □

Combining Lemma 2 and Lemma 4, we have the following theorem:

Theorem 4.6. *If a graph G is not an s-biplex cluster graph, then we can find a forbidden subgraph in G in polynomial time with the size bounded by $3s+3$.*

Finally, we have:

Corollary 4.7. *S-biplex cluster editing is fixed-parameter tractable with respect to (s,k).*

4.4 Average-s-Plexes

In this section, we consider the average-s-biplex editing problem, proving its NP-completeness and its fixed parameter tractability. with respect to parameters (s, k). To show its NP-hardness, a two-step reduction is presented: First, we reduce a well-known NP-complete maximum balanced biclique (MBB) to equal-size bicluster editing, afterwards, a reduction from equal-size bicluster editing to average-s-biplex editing is conducted. The equal-size bicluster editing (ESBE) is defined as follows:

> **Input**: An undirected bipartite graph $G = (U, V, E)$ and two integers $k, d \geq 0$.

Question: Can G be transformed by editing at most k edges into d disjoint bicliques $\{C_1, C_2, C_3, ...C_d\}$, $C_i = (U_i, V_i, E_i)$, $1 \leq i \leq d$, such that $|U_i| = |U_j|$ and $|V_i| = |V_j|$ for all $1 \leq i, j \leq d$?

The edge deletion version of this problem requires only edge deletions.

Theorem 4.8. *Equal-size bicluster editing is NP-complete.*

Proof. See the proof in Appendix A.1.4. □

Theorem 4.9. *For every constant $s \geq 1$, average-s-biplex editing is NP-complete.*

Proof. See the proof in Appendix A.1.5. □

We present a kernalization procedure for average-s-biplex editing with respect to the parameter (s, k). In order to show this, first we reduce the problem into an integer-weighted version and afterwards we describe three reduction rules that can be carried out within polynomial time.

We introduce two types of weights to describe the weighted version of average-s-biplex editing: Vertex weights and edge weights, inspired by the idea of the reduction of the weighted version of cluster editing, i.e. for any pair of vertices that cannot be separated by k edge modifications, we merge them into one "multi-vertex". Obviously, for all vertices merged, they end up in the same average-s-biplex in all optimal solutions.

We denote the vertex weight as $\sigma(u)$ which keeps track of the number of vertices merged into u. The vertex weight of a set of vertices S is defined as: $\sigma(S) = \sum_{v \in S} \sigma(v)$. Moreover, let $\delta(u)$ be the subset of vertices $\{u_1, u_2, ..., u_r\}$, $r \geq 1$ that merged into u, i.e., $\sigma(u) = |\delta(u)|$. The edge weight, $\omega(u, v)$, is defined between two arbitrary entities (The concept "entity" represents vertices, multi-vertices and sets of vertices), storing the number of edges between them. The degree of a vertex u is defined as: $d'(u) = \omega(u, N(u))$. Thus, for a weighted bipartite graph $G = (U, V, E)$, the average degree of the vertices in U is defined as: $\overline{d_U} = \frac{\omega(U,V)}{\sigma(U)}$.

Hence a bipartite graph $G = (U, V, E)$ is a weighted average-s-biplex, if $\overline{d_U} \geq \sigma(V) - s$ and $\overline{d_V} \geq \sigma(U) - s$. The weighted version of the problem can be defined as:

Input: A graph $G = (U, V, E)$, with vertex weight $\sigma(u)$ as a function:

$$\sigma(u) : \begin{cases} U & \rightarrow & [\ 1, |U|\ ] \\ V & \rightarrow & [\ 1, |V|\ ] \end{cases}$$

and edge weight $\omega(u, v)$ as a function:

$$\omega(u, v) : E \quad \rightarrow \quad [\ 1, |U||V|\ ]$$

and a nonnegative integer k.

Question: With edge modifications whose total weight is at most k, can G be edited into a weighted average-s-biplex cluster graph?

Note that if we set $\sigma(u) \coloneqq 1$, $\delta(u) \coloneqq \{u\}$ and for each $\{u, v\} \in E$, $\omega(u, v) \coloneqq 1$, an instance of average-s-biplex editing can be easily reduced to an instance of WEIGHTED average-s-biplex editing. In this reduction, parameters k and s are not changed.

The following three reduction rules are designed for weighted average-s-biplex editing, which lead to a problem kernel with no more than $2k((s + 1)(4k + 6s) + 1)$ vertices.

- **Rule 1.** Remove all connected components in G that are already weighted average-s-biplexes.
- **Rule 2.** For two vertices $u, v \in U$ or $u, v \in V$, let $S(u, v) \coloneqq N(u) \cap N(v)$. If $\min\{\omega(u, S(u, v)), \omega(v, S(u, v))\} > k$, then we merge u and v, by replacing u and v with a new vertex v', such that v' satisfies:

  - $\sigma(v') = \sigma(u) + \sigma(v)$

  - $\omega(v', x) = \omega(u, x) + \omega(v, x)$ for every x with $\{u, x\} \in E$, $\{v, x\} \in E$

Lemma 4.10. ***Rule 2.*** *is correct.*

Proof. See the proof in Appendix A.1.6 □

The function of Rule 2. is to merge (or replace) the vertices that we cannot afford separating. Based on the same idea, we consider another scenario: If a vertex u has a large set of neighbors that only connects to u but to no other vertex, then we cannot possibly delete the edges between u and all its "unique" neighbors. Let $N^*(u) \subseteq N(u)$ be a set of vertices such that $\forall v \in N^*(u)$, v satisfies: (1) $N(v) = \{u\}$ and (2) $\sigma(v) = 1$. Rule 3. is then presented to reduce the size of $N^*(u)$:

- **Rule 3** For each $u \in G$, if $|N^*(u)| > k$, then we replace $N^*(u)$ with a subset of vertices containing $(k+1)$ vertices: $\{v_0, v_1, v_2, ..., v_k\}$, such that $\omega(u, v_0) = \omega(u, N^*(u)) - k$ and $\omega(u, v_i) = 1$ for all $1 \leq i \leq k$.

Lemma 4.11. *Rule 3 is correct.*

Proof. See proof in Appendix A.1.7. □

Theorem 4.12. *(Weighted) average-s-biplex editing is fixed-parameter tractable with respect to parameter (s, k) and admits a kernel of at most $2k((s+1)(4k+6s)+1)$ vertices.*

Proof. See proof in Appendix A.1.8. □

4.5 Defective Bicliques

We prove now the NP-completeness of s-defective bicluster editing

Theorem 4.13. *For every $s \geq 0$, s-defective bicluster editing is NP-complete.*

Proof. See proof in Appendix A.1.9. □

Next, we show the fixed-parameter tractability of s-defective bicluster editing by proving that for every $s \geq 1$, all minimal forbidden induced subgraphs contain at most $2s + 3$ vertices and hence we are able to find a minimal forbidden subgraph in polynomial time.

Lemma 4.14. *For every $s \geq 1$, every minimal forbidden induced subgraph of s-defective bicluster graphs contains at most $2s + 3$ vertices. Given a graph that is not an s-defective bicluster graph, a minimal forbidden induced subgraph can be found in $O((|U| + |V|) \cdot |E|)$ time.*

Proof. Denote $H = (R, T, E')$ as a minimal forbidden induced subgraph of s-defective bicluster graph. Clearly, H is connected. Towards contradiction we assume H contains more than $2s + 3$ vertices. We distinguish 2 cases:

Case i. There exists a cut vertex in H. Let $u^* \in R$ be a cut vertex. Obviously, by $s + 4 \leq 2s + 3$ for $s \geq 1$, we can always find in H a connected subgraph H' such that H' contains u^* and other $s + 3$ vertices and u^* is a cut vertex in H'. Let $H' = (R', T', E'')$ We prove that H' is forbidden. By removing u^*, we obtain a set of disjoint connected components $\mathcal{H} = \{H_1, H_2, ..., H_l\}$, $H_i = (R_i, T_i, E_i)$. Thus, we

have the number of missing edges e_m in H' is at least:

$$\begin{aligned}
e_m &\geq \frac{1}{2}\sum_{i=1}^{l}(|R_i|(|T'|-|T_i|)+|T_i|(|R'|-1-|R_i|)) \\
&= \frac{1}{2}\sum_{i=1}^{l}|R_i||T'|+\frac{1}{2}\sum_{i=1}^{l}|T_i|(|R'|-1)-\sum_{i=1}^{l}|R_i||T_i| \\
&= (|R'|-1)|T'|-(|R_1||T_1|+\sum_{i=2}^{l}|R_i||T_i|) \\
&\geq (|R'|-1)|T'|-(|R_1||T_1|+(\sum_{i=2}^{l}|R_i|)(\sum_{i=2}^{l}|T_i|)) \quad (*) \\
&= (|R'|-1)|T'|-(|R_1||T_1|+(|R'|-1-|R_1|)(|T'|-|T_1|)) \\
&= |T_1|(|R'|-1-|R_1|)+|R_1|(|T'|-|T_1|) \\
&\geq |R'|+|T'|-3 \quad (**)
\end{aligned}$$

Inequality (*) holds because for any integer $a_1, b_1, a_2, b_2 > 0$, we have $a_1 \cdot b_1 + a_2 \cdot b_2 \leq (a_1+b_1)(a_2+b_2)$. Inequality (**) is the minimum value of the function $f(|T_1|,|R_1|) = |T_1|(|R'|-1-|R_1|)+|R_1|(|T'|-|T_1|)$, with $1 \leq |R_1| \leq |R'|-1$ and $1 \leq |T_1| \leq |T'|$. Thus, we have $e_m \geq s+1$. Since $|R'|+|T'| = s+4$, we have H' being a forbidden subgraph, thus contradicts the assumption.

Case ii. If there is no cut vertex in H, then we know that $\forall v \in H$, v must be incident to missing edge(s), otherwise we can just remove v from H without changing the forbidden subgraph property. Let $n = |U|+|V|$, m_0 be the minimum "anti-degree" ("anti-degree" is the number of missing edges incident to a given vertex) in H and m_t be the total number of missing edges in H. Hence we have the inequalities: (1) $\frac{1}{2} \cdot n \cdot m_0 \leq m_t$ and (2) $m_t - m_0 \leq s$.

Inequality (1) holds because each vertex is incident to at least m_0 missing edges and altogether we have no more than m_t missing edges. Inequality (2) holds because H is a minimal forbidden subgraph and the removal of vertex v will decrease the number of total missing edges by at least m_0. Since H is minimal, $\forall u \in H$, $H-u$ is

not forbidden and thus has no more than s missing edges. Solving the inequalities, we have: $n \leq \frac{2s+2m_0}{m_0} = \frac{2}{m_0}s + 2 \leq 2s + 3$.

Thus if $n > 2s + 3$, then at least one inequality above is not satisfied and hence H is not a minimal forbidden induced subgraph. To locate a minimal forbidden induced subgraph, we first check if the given connected graph G is an s-defective bicluster. If not, we check for each $v \in G$, the subgraph $G - v$. If $G - v$ is still not an s-defective biclique, then we remove v from G. Thus to find a minimal forbidden induced subgraph takes at most $O((|U| + |V|)|E|)$ time. □

Theorem 4.15. *s-defective bicluster editing is fixed-parameter tractable with respect to (s,k).*

4.6 Outlook

For all three problems, further algorithmic improvements are possible: For s-biplex and s-defective bicluster editing, a more elegant and efficient problem kernel is possible, and for average-s-biplex editing, an efficient branching strategy other than brute-force is beneficial to be applied on the reduced problem kernel. Moreover, in many practical applications, for example in computational biology, high-quality heuristic algorithms should always be taken into account. Finally, it is also interesting to consider other meaningful density measurements and study their classical and parameterized complexity.

Chapter 5

n-CluE

In this chapter, we introduce a software package "n-CluE" for solving biclustering problems and n-clustering problems in the framework of bicluster editing and n-cluster editing. The package "n-CluE" implements one exact algorithm and two heuristic algorithms. The exact algorithm is based on the fixed-parameter tractability theory discussed in the Section 3.4. A kernelization procedure and a branching strategy is developed to traverse all possible editing behaviors and find the best set of insertions and deletions that leads to smallest editing cost.

The edge deletion heuristic improves the running time of the fixed-parameter algorithm by locating the edge deletions that are most beneficial to convert the input graph into disjoint bicliques, in a greedy iterative manner. This reduces the running time to $O(|E|(|E| + |V|^2) + |V|^3)$.

The second heuristic is inspired by a well-known graph layout algorithm: the Fruchterman-Reingold algorithm [59]. The basic goal of this algorithm is to find an arrangement in an u-dimensional ($u > 1$) space such that similar nodes are located closer to each other than the dissimilar nodes. The first version of this "*layout-based*" algorithm — Bi-Force, is designed for biclustering problems on bipartite graphs and

matrices [179]. We then extended Bi-Force to n-Force to deal with n-partite graphs of n-cluster editing.

The content of this chapter is based on the published research articles listed below:

- Peng Sun, Jiong Guo, and Jan Baumbach. Biclue-exact and heuristic algorithms for weighted bi-cluster editing of biomedical data. In *BMC proceedings*, volume 7, page S9. BioMed Central Ltd, presented at GLBIO2013, 2013
- Peng Sun, Jiong Guo, and Jan Baumbach. Integrated simultaneous analysis of different biomedical data types with exact weighted bi-cluster editing. *J Integr Bioinform*, 17, 2012
- Peng Sun, Nora K Speicher, Richard Röttger, Jiong Guo, and Jan Baumbach. Bi-force: large-scale bicluster editing and its application to gene expression data biclustering. *Nucleic Acids Research*, page gku201, 2014
- Peng Sun, Jan Baumbach, and Jiong Guo. Efficient large-scale bicluster editing. In *German Bioinformatics Conference 2014*, pages 54–60

5.1 Introduction

The major part of n-CluE contains three algorithms: one exact algorithm based on fixed-parameter tractability, one faster-running heuristic algorithm based on optimal edge deletion estimation, and another heuristic algorithm named n-Force, motivated by the well-known physics-inspried graph layout algorithm of Fruchterman and Reingold [59].

The three algorithms are designed for different scenarios: The exact fixed-parameter algorithm is developed for small input graphs. The edge deletion heuristic performs best on medium-size graphs where edge deletions are more important than edge insertions. N-Force is the only method that can solve n-cluster editing problems

to deal with heterogeneous data of n ($n > 2$) different sources and is designed for large-scale input graphs with thousands of nodes.

Assuming $|s(u,v)| > 1$ for all possible u and v, the fixed-parameter algorithm finishes in $O(4^k)$ time; the edge deletion heuristic algorithm needs $O(|E|\dot{(}|E|+|V|^2)+|V|^3)$ time and n-Force is bounded by $O(D_2 \cdot n^2)$ for D_2 iterations.

We first introduce and discuss the three algorithms. Afterwards, a systematic evaluation is conducted to compare the performances of the three algorithms in n-CluE on artificially generated graphs.

5.2 Fixed-Parameter Algorithm

The strategy based on fixed-parameter tractability to solve Cluster Editing has been briefly introduced in Section 3.5. Fixing the "*P3s*" is the central part.

Bicluster editing is similar to its counterpart — cluster editing: we transform a given bipartite graph into a transitive bipartite graph by edge insertions and deletions with minimal costs for these modifications. We consider a bipartite graph $G = (U, V, E)$ transitive if it satisfies any of the following equivalent conditions:

- For an arbitrary subset of four vertices, (u, v, w, x), where $(u, w) \in U$ and $(v, x) \in V$, we have $(u, v) \in E$, $(w, v) \in E$ and $(w, x) \in E \Rightarrow (u, x) \in E$.
- No acyclic connected subgraph of four vertices exists, i.e., for each (u, v, w, x) in the graph, where $(u, w) \in U$ and $(v, x) \in V$, we have $|E \cap \{(u, v), (w, v), (u, x), (w, x)\}| \neq 3$.
- G is a union of disjoint bicliques (i.e. complete bipartite graphs).

We assume that the input graph consists of only one single connected component since we can apply the algorithms on each connected component separately, without loss of generality. An optimal solution of the bicluster editing problem would never

join separate components, since we can always find a cheaper solution where all separated components remain separate [74].

In this study, we use "P4" in short for "an acyclic connected subgraph of four vertices". As mentioned before, a bipartite graph is transitive if and only if it contains no P4. Denote $B(G)$ to be the set of all P4s, i.e. $B(G) = \{(u, v, w, x) \mid |E \cap \{(u, v), (w, v), (u, x), (w, x)\}| = 3\}$. G is transitive if and only if $B(G) = \emptyset$.

Our fixed-parameter algorithm performs two major steps: *Data Reduction* and *Branching Strategy*.

5.2.1 Data Reduction

Data reduction is a preprocessing step of the fixed-parameter algorithm that reduces the instance size by removing those parts of the problem instance that do not need to be repaired and thereby do not need to be considered in the following steps. We first recognize all connected components as individual inputs. Then the algorithm checks whether each component is already a biclique or not. If this is the case, then the algorithm removes the whole component from the input and reports it as a part of the solution. This procedure finishes within $O(|V| + |E|)$ time.

5.2.2 Branching Strategy

Branching refers to a search tree procedure to find and edit the P4s using edge insertions and deletions. We have four possibilities to convert a P4 into bicliques: removing one of the three edges, resulting in two bicliques, or complete the P4 with one edge insertion (Fig. 5.1). More specifically:

Suppose (u, v, w, x) is an arbitrary P4 and let $(u, v), (w, v), (w, x)$ be the three edges in the P4. The following four cases are checked recursively,

- Insert (u, x) by setting the weight of (u, x) to "permanent" (Fig. 5.1 b)

- Delete (u, v) by setting the weight of (u, v) to "forbidden" (Fig. 5.1 c)
- Delete (w, v) by setting the weight of (w, v) to "forbidden" (Fig. 5.1 d)
- Delete (w, x) by setting the weight of (w, x) to "forbidden" (Fig. 5.1 e)

The search tree procedure starts when a P4 is located. Four branches are created for one P4 in the search tree; each represents one of the editing possibilities. Then we recursively visit the four branches one by one, performing the corresponding edge insertions or deletions and update k to $k' = k-$(insertion or deletion cost). We implement the whole algorithm in a recursive manner. If the editing in a certain branch leads to $k' < 0$, then the corresponding branch is skipped. The algorithm stops when the entire tree is visited, and returns the optimal solution found. This branching strategy checks in worst case running time of $O(4^k)$ if a solution of at most cost k exists.

5.3 Edge Deletion Heuristic

In the fixed-parameter algorithm, we are aiming for repairing all the P4s to make G transitive. The repairing action is either an edge insertion or edge deletion. It is obvious that the difficult part of the problem is to correctly locate the edge deletions, for the edge deletions determine the number of resulting disjoint bicliques. Therefore, it would be beneficial to find the most promising positions of edge deletions first. Then edge insertions can easily be carried out by inserting all edges required to make each disjoint component transitive. This is the main idea behind our edge deletion heuristic algorithm.

We define a function to score the edge removal candidates and greedily delete the edge with highest score in each step, until further deletions do not improve the solution. For each P4 (u, v, w, x) (where $(u, v), (w, v), (w, x) \in E$), we define deviation from transitivity of G — $D(G)$ as:

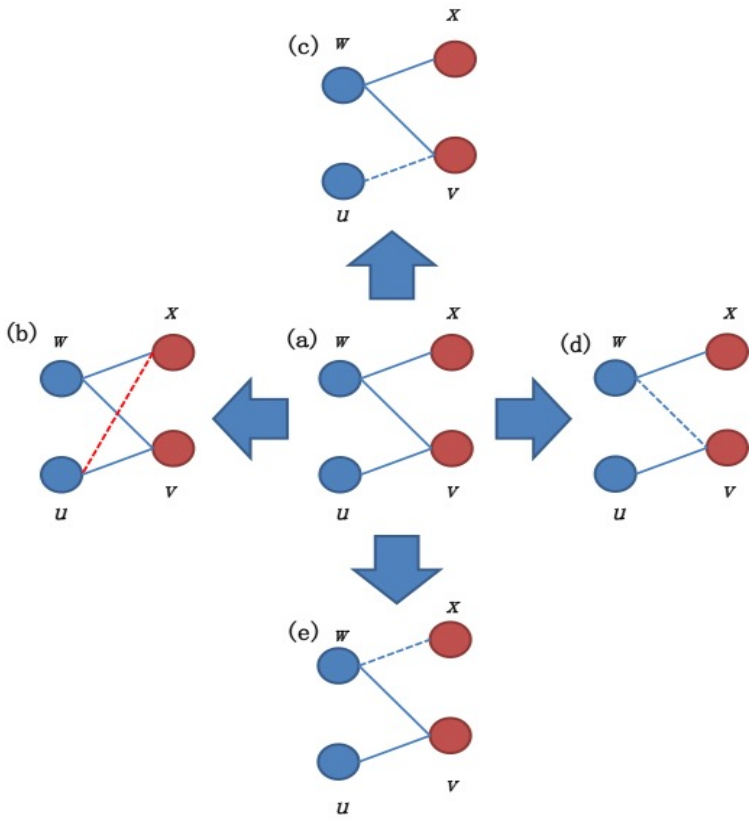

Figure 5.1: Blue dashed lines correspond to edge deletions and red dashed lines correspond to edge insertions. Four possibilities of repairing a P4 are presented: (b) Insertion of the missing edge (u, x); (c) deletion of the edge (u, v); (d) deletion of the edge (w, v); and (e) deletion of the edge (w, x).

$$D(G) = \sum_{\{u,v,w,x\} \in P4} \min\{|s\{u,v\}|, |s\{v,w\}|, |s\{w,x\}|, |s\{x,u\}|\}$$

The scores of edge deletions are computed as follows: Let (u, v) be an arbitrary edge in $G = (U, V, E, s)$. $G' = (U, V, E \backslash (u, v), s')$ is G after the removal of (u, v), where $s'(x, y) = s(x, y)$ for all possible x, y pairs, except $s'(u, v) = -\infty$ ((u, v) set to "forbidden"). Then we define:

$$\Delta_{uv}(G) = D(G) - D(G') - s(u, v)$$

as the *transitivity improvement* of edge (u, v) where $s(u, v)$ is the cost of edge deletion.

The edge deletion heuristic algorithm consists of three functions: *REMOVE_ CULPRIT(G)*, *TRANSITIVE_ CLOSURE_ COST(G)* and *EDGE_ DEL_ MAIN(G)*. *REMOVE_ CULPRIT(G)* returns the edge with highest *transitivity improvement* ($argmax_{(u,v)\in E}\{\Delta_{uv}(G)\}$) and removes it from G; *TRANSITIVE_ CLOSURE_ COST(G)* returns the total cost of all edge insertions required to convert G into a biclique, assuming G is connected; *EDGE_ DEL_ MAIN(G)*is the main function of the edge deletion heuristic.

The first invocation of *REMOVE_ CULPRIT*(G) can be finished in $O(|E| \cdot |V|^2)$ time, since computing each $\Delta_{uv}(G)$ can be finished in $O(|V|^2)$ time, for only those P4s that contain (u, v) are considered. The subsequent routine calls require $O(|V|^2)$ time to update the scores of the edges that were influenced by the deletion of (u, v), and finally $O(|E|)$ time to find the maximum scored edge. This results in a total running time of $O(|V|^2 + |E|)$. *TRANSITIVE_ CLOSURE_ COST(G)* sums up the cost for a transitive closure, accepting a running time of $O(|V|^2)$.

EDGE_ DEL_ MAIN(G) returns a solution object, containing the edge modifications and the costs needed for converting the input graph into a transitive one. We keep the assumption that G is connected.

In our heuristic, checking for connected components requires $O(|E|+|V|)$ time and *REMOVE_ CULPRIT*(G) requires $O(|V|^2+|E|)$ time. *TRANSITIVE_ CLOSURE_ COST*(G) takes $O(|V|^2)$ time for each disjoint component. Therefore, the total running time of our algorithm is $O(|E|(|E| + |V|^2) + |V|^3)$. Find pseudo-code of the *EDGE_ DEL_ MAIN*(G) in Appendix A.2.

5.4 n-Force

n-Force mainly seeks to arrange all nodes of a graph in a two-dimensional plane (or n-dimensional space) such that "similar" nodes are located closer to each other than

to others. n-Force, afterwards, assigned the nodes in each "dense" part of the graph layout to one bicluster by single linkage clustering or k-means clustering based on the Euclidean distances. The algorithm is carried out in a three-step procedure: (a) layout generation, (b) bicluster partitioning, and (c) post-processing.

5.4.1 Layout Generation

In this stage, the coordinates of all nodes are generated and re-arranged in a way that the nodes with higher similarities are located next to each other, and far away from those that are dissimilar. n-Force computes "physical forces" between all pieces of nodes, i.e., the magnitudes of the forces with which similar nodes attract each other, dragging them closer while dissimilar nodes repel each other, pushing them farther away. The whole algorithm starts with an initial layout where nodes are evenly located on the surface of an n-dimensional sphere. The radius R of the sphere is a parameter of n-Force. The strengths of attracting/repelling forces depend on the current positions of the two nodes, attraction/repulsion coefficient and the corresponding cost to delete the edge or to insert the missing edge between the two nodes. The rearrangement is performed in an iterative manner. In each round, the movement of each node is the cumulative effect of the attractions and repulsions of all other nodes. Afterwards, all nodes are re-positioned to the new locations simultaneously according to the magnitudes of the movements. The whole procedure is repeated for I times. The attracting/repelling effect from node v to u is computed by the following formula:

$$f_{u \leftarrow v} = \begin{cases} \dfrac{cost(uv) \cdot f_{att} \cdot log(d(u,v)+1)}{|V|} & \text{for attraction} \\ \dfrac{cost(uv) \cdot f_{rep}}{|V| \cdot log(d(u,v)+1)} & \text{for repulsion} \end{cases}$$

In the formula above, $f_{u \leftarrow v}$ represents the attracting/repelling effect from node v to u, i.e., the magnitude of the movement of u caused by v. When there is an

edge between u and v, the two nodes attract each other and if otherwise, they repel each other. f_{att} and f_{rep} are the attractive and repulsive factors, respectively. $d(u, v)$ represents the Euclidean distance between node u and v. Obviously, the threshold t, which controls whether two nodes are connected with an edge or not, affects the density/granularity of the bicluster editing model: the smaller t is, the fewer biclusters there are and the larger their sizes, and vice versa.

To accelerate the convergence of the nodes to stable positions, a cooling parameter is used to limit the maximal magnitudes of attractions and repulsions, similar to self-organizing maps [107]. This means in a certain iteration i, the movement magnitude cannot exceed the current cooling parameter M_i. The cooling parameter starts with an initial value M_0 as a parameter in n-Force and decreases with every iteration.

At the end of this stage, the positions of all nodes are fixed and similar nodes should be close to each other. In the next step, we make use of this assumption and partition the layouted graph in a way that optimizes the editing costs.

5.4.2 n-Cluster Partitioning

Based on the coordinates of the nodes obtained in the previous stage, we partition the graph into disjoint n-clusters using either of two different geometric clustering methods (user selection): single-linkage clustering (SLC) and k-means. Both, SLC and k-means are standard methods in computational cluster analysis [204]. The density parameters of the two algorithms (distance threshold δ for SLC and the number of clusters k for k-means) are varied systematically (SLC: $\delta = 0, \sigma, 2\sigma, \ldots, M_0 + R$ in steps of σ, k-means: $k = 2, 3, 4, \ldots, |V|/3$). For each clustering result we compute the editing costs necessary to create this solution. Finally, we keep the solution that has minimum editing costs before we proceed to post-processing.

SLC is parameterized by a maximum distance δ where the distance between two clusters is represented as the distance between two closest elements in the two clusters.

The distance between two given clusters c_1 and c_2, for instance, is computed as: $d(c_1, c_2) = \min_{u \in c_1, v \in c_2} d(u, v)$. Note that we use the Euclidean distance here. We start with an arbitrary node u and define it to be in bicluster b_1. Next, for each unassigned node x and a distance $d(x, b_1) < \delta$, we include x into b_1. This step is repeated until no such node x can be found anymore. Then the procedure starts again: Pick an undefined node u' and define it to be in b_2 and all the nodes located close enough to u' are assigned as b_2. The clustering procedure is finished until all nodes are assigned to clusters.

To optimize the clustering result, the distance threshold δ is varied in a certain range to seek for the lowest n-cluster editing cost. We start with $\delta_0 = 0$ and increase δ by a certain step σ until it reaches a given δ_{max}. In the default setting of n-Force, δ_{max} is set to $M_0 + R$. This is to insure that almost all possible δs are evaluated. N-Force also accepts a user-defined δ_{max}. However, note that although smaller δ_{max} can speed up the algorithm, it risks overlooking the possibly better threshold options.

K-means is a common method in cluster analysis, aiming to partition n elements into k clusters. K-means starts with a random partition of all the elements into k different clusters and iteratively updates their centroids. In this study, we applied k-means with k ranging from 2 to $|V|/3$, with a ceiling value of 500. The partition with smallest n-cluster editing cost is reported. Finally, we compare the optimized results of single-linkage clustering and K-means and pick up the result with lowest editing cost.

5.4.3 Post-Processing

Here, we try to further reduce the clustering costs, which includes two steps: (a) n-cluster merging and (b) nodes moving.

To reduce the number of redundant n-clusters, particularly the singletons, we try to merge n-clusters. First, all n-clusters are ordered by the number of nodes in an

ascending order. Let $B = (b_1, b_2, ..., b_l)$ be the l ordered n-clusters, where $|b_i| \leq |b_j|$, for all $i \leq j$. For all pairs of n-clusters b_i and b_j with $1 \leq i < j \leq l$, we calculate the cost that would emerge from merging the two, i.e., $cost(b_1, b_2, ..., b_i \cup b_j, ..., b_l)$. Once a B' with a lower overall cost than before is found, we re-define the n-clusters according to B' by merging b_i and b_j. This step is repeated until no beneficial merging can be done anymore.

After merging the clusters, another post-processing step similar to Restricted Neighborhood Search Clustering [100] is carried out. Let $B = (b_1, b_2, ..., b_{l'})$ be the biclusters after the merging step, for each b_i and b_j, such that $1 \leq i < j \leq l$, we compute the costs that would result from moving $v \in b_i$ to b_j. If the overall cost can thereby be reduced in this step, v is moved to b_j. Similarly, this step is repeated until no node move is beneficial.

This is the final result of the n-Force algorithm. The software implementation's output is a list of nodes together with their n-cluster memberships and their final layout positions. For each instance, we also report the number of editing actions (edge insertion and deletions) as well as the total cost to compute this solution.

5.4.4 Demonstration of Layout Concept

See Fig. 5.2 for how the nodes are re-arranged during the "*layout generation*" step: the nodes with higher similarities tend to move closer as the algorithm proceeds. The input graph is generated by the following rules:

- The input graph contains 180 nodes.
- The 180 nodes are evenly divided into 3 sets.
- In the input graph, we randomly selected 7 pre-defined n-clusters.
- The edge weights are generated following Gaussian distribution. The edge weights of the intra-cluster edges follow a Gaussian distribution with mean

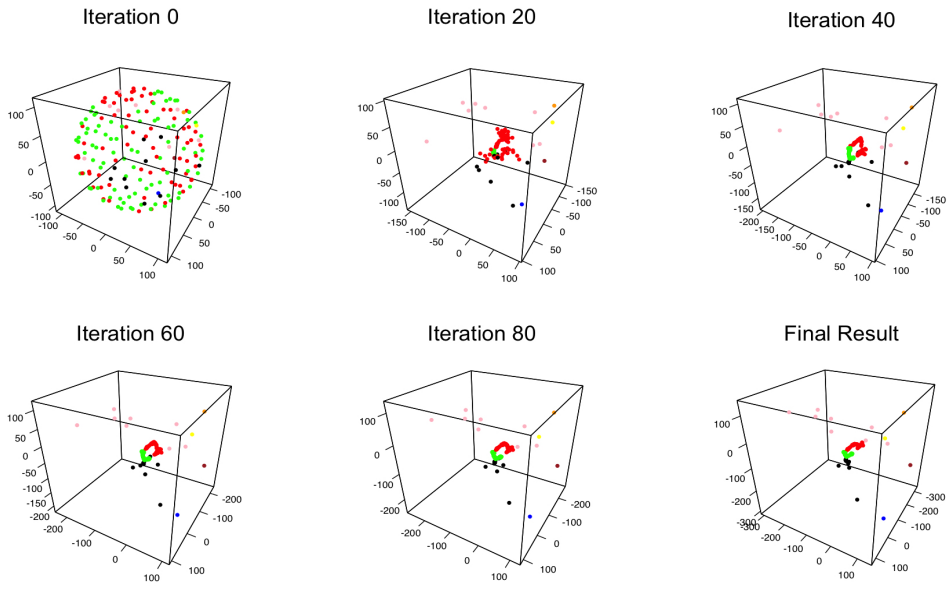

Figure 5.2: The movements of the nodes during layout generation step. The colors of the nodes indicate the different pre-defined groups assigned to the corresponding node before the start of n-Force.

= 15, std= 10. The cross-cluster edges follow the Gaussian distribution with mean = -15, std = 10.

The input graph was created to simulate the real biological data set under the assumption that the real-world data sets are not far from transitive. The nodes in the same pre-defined cluster are more similar to each other than to the ones in different pre-defined clusters. The aim of n-Force is to move similar nodes (in this case, the nodes belonging to the same pre-defined cluster) closer to each other. As shown in Fig. 5.2, in the final results, most of the nodes with same color were located comparatively close to each other, with only a few outliers (such as the pink nodes). Moreover, it took n-Force only 40 iterations to generate the preliminary final layout, indicating that the similar nodes are grouped together in an efficient way.

5.4.5 Training

n-Force is a heuristic algorithm with several parameters to be optimized: The number of iterations I, the attraction and repulsion coefficients, f_{att} and f_{rep}, the initial maximum magnitude M_0 and the radius for initial layout R. Hence, two evolutionary training strategies were implemented: A general training procedure and an input-specific parameter training.

General Training

The "*general training*" was conducted to obtain the default setting of the parameters. We tried to find a set of parameters that fits a "general scenario", i.e. graphs with varying error-edge-rates (for the definition of error-edge-rate, see Sec. 5.5). Graphs for general training were generated according to the protocol described in Sec. 5.5. By varying the deviations of the two Gaussian distributions, graphs with 9 different error-edge-rates were generated: 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, with 10 repeats for each error-edge-rate resulting in a training graph set of 90 graphs.

The training was conducted in an evolutionary manner: First we randomly selected 1000 parameter sets within certain ranges: $(0, 10)$ for f_{att} and f_{rep}, $(0, 300)$ for the iterations I, $(0, 1000)$ for initial maximum magnitude M_0, and $(0, 400)$ for radius R. Then we applied the randomly selected 1000 parameter sets on our artificial graph set and picked the best three parameter sets (minimal n-cluster editing total costs). These sets were used as starting point for the following training procedure:

One training iteration consists of two steps: (a) for each parameter set, compute the sum of the costs for solving all graphs; (b) generate new parameter sets in an evolutionary manner based on the old sets and their costs. After running the algorithm on all input graphs, the parameter sets were ordered ascendingly by total costs. A new list of parameters for the next training iteration was generated based on the first three parameter sets with least costs in the previous iteration: We kept the two best

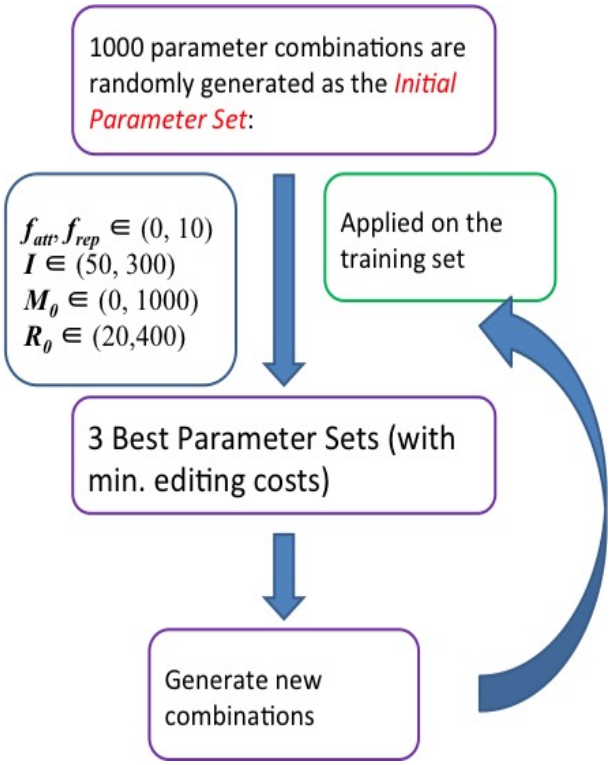

Figure 5.3: General training.

parameter sets and put them directly in the new parameter list. The third parameter set in the new list was computed as the mean of the best three parameter sets in the previous iteration. Then, the best three sets were permuted to obtain the fourth, fifth and sixth sets in the new list. The next three sets for the new iteration were randomly picked up around the best set in the previous iteration. For each parameter α (α: f_{att}, f_{rep}, M_0, R and I), we randomly picked up a number within the range of $(0.9\alpha, 1.1\alpha)$. Finally, in a similar way, we randomly picked the last three sets in the new list, only altering the ranges to $(0, 2\alpha)$. The resulting 12 new parameter sets entered the next iteration of training. Then the whole procedure was repeated. After a given number of iterations, we picked up the best set of parameters as the final optimized set. Fig. 5.3 shows the work-flow of the general training.

The initial parameter combination obtained from general training is: $I = 134$, $f_{att} = 2.484$, $f_{rep} = 1.323$, $M_0 = 51.84$ and $R = 112.5$.

Specific Training

Besides general training, an additional "*specific training*" is conducted by the software for each specific input case to further refine our best general parameter set to fit to each specific graph input. We make use of the following trick: Without loss of generality, an n-cluster editing problem instance is assumed to contain only one connected component, since disjoint components can be treated separately without interfering the results of other components. Real biological data often contains more than one connected component. We further assume that smaller components have a similar graph structure as the larger ones. For a given input graph, we train the input-specific parameters on the smaller disjoint components in order to adapt our algorithm to the specific input data, without compromising the running time too much (as smaller instances can be computed much faster than bigger ones). The whole procedure works as follows: all connected components of a given input graph are sorted by their numbers of nodes. Then, the parameter set, optimized via general training, is further trained on the small disjoint components. We start with the smallest components, following the same evolutionary training procedure as in the general training. On the second-smallest component, we repeat this process but with less training iterations (T_{max} - 0.5 $\times$ size of the component). We stop this parameter training when a connected component size of T_{max} is reached (here we use $T_{max} = 40$) and apply the best parameter set found so far to all bigger problem instances.

5.4.6 Runtime Analysis

Let $n = |U| + |V|$ for an input graph $G = (U, V, E)$. In the "layout generation" step, where n-Force arranges the positions of all nodes, it consumes $O(n^2)$ time to compute the mutual attracting/repulsing forces in each iteration. Thus the layout generation step finishes in $O(I \cdot n^2)$, where I is the number of iterations. The single-linkage clustering runs in $O(D_1 \cdot n^2)$, where D_1 is the number of different thresholds used.

With the number of iterations limited to be 200, k-means terminates in $O(n)$ time [2, 125]. Finally for post-processing, each iteration takes $O(n^2)$ time and the total running time is bounded by $O(D_2 \cdot n^2)$ for D_2 iterations. Since D_2 might increase with n, we added an empirical limit of 500 iterations to D_2. In most cases (see Chapter 7 and Chapter 8), n-Force did not reach this limit and we observed only small numbers of iterations before it terminated.

In summary, the overall running time for n-Force grows quadratic in the number of nodes.

5.5 Performance Evaluation

The performance of the three algorithms was assessed by comparisons on artificial graphs. Each artificial graph with n nodes was created by randomly assigning the pairwise similarities based on the following rules: randomly picked up k ($k \in [1, n]$) nodes and defined them to be in one cluster. This step was repeated on the remaining $n - k$ nodes until no node was left. Similarities were computed with two Gaussian distributions: $N(\mu_{intra}, \sigma^2_{intra})$ and $N(\mu_{inter}, \sigma^2_{inter})$. The first one was used to assign the similarities between two nodes belonging to the same pre-defined bicluster (intra-bicluster similarities), and the latter was used to assign the similarities between two nodes from different pre-defined biclusters (inter-bicluster similarities). We adjusted the parameters in the Gaussian distributions to control the "error-edge-rate", i.e., the probability of the occurrence of an *intra missing edges* (missing edges within a pre-defined bicluster) or an *inter edge* (edge between two different pre-defined biclusters). The edge threshold t_0 was set to be 0. A set of such bipartite graphs with varying error-edge-rates is created: From "almost-bicluster" (error-edge-rate of 0.14) graphs to fully random graphs (error-edge-rate of 0.5). Almost-bicluster graphs with relatively low error-edge-rates are used to simulate real-world biological networks, which

usually need only a small number of edge modifications to turn a bipartite graph into a bicluster graph. To evaluate the bicluster editing algorithms, we assessed their robustness for input graphs with varying error-edge-rates.

Two experiments were conducted. 80 artificial graphs with various sizes but constant small error-edge-rate (arbitrarily chosen as 0.14) were generated. For robustness assessment, we fixed the sizes of input graphs to be 80 and varied the error-edge-rate $(0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4)$. Edge deletion heuristics and n-Force were applied on the inputs to test their capacity of keeping low running time while error-edge-rates increased. The running time for each input was limited to two hours.

5.6 Results and Discussion

5.6.1 Editing Strength

In this section, the performances of the three algorithms are compared. We used two different data sets: Almost-bicluster graphs and graphs with various error-edge-rates. The three algorithms were applied to all graphs, while recording the editing costs and the running times. Maximum running time was set to two hours.

Table 5.1 shows the running times and accuracy of the three algorithms. The fixed-parameter algorithm was able to achieve very small running times on small-sized and medium-sized graphs, yet as the sizes of graphs grow, the performance of the fixed-parameter algorithm suffered, indicating the NP-hardness of the underlying problem. When the number of vertices exceeds 40, the fixed-parameter algorithm could not finish within reasonable time. On the other hand, the edge deletion heuristic algorithm required significantly less time than the fixed-parameter algorithm on bigger graphs. In terms of costs, the performance of the edge deletion heuristics was almost as good as that of the fixed-parameter algorithm. In summary, the heuristic finds solutions that were almost equally good but in significantly less time.

Table 5.1: Performance comparison between n-Force and the edge deletion heuristic (EDH) as well as the fixed-parameter algorithm (FPA). The cost here refers to the editing costs. The results here are the average of five repeated runs. P.T. stands for "parameter training" (of n-Force's heuristic parameters), R.T. is the run time (given in seconds). The smallest editing cost and running time are marked with bold font. Note that when the sizes of the input graphs grew larger than 100 nodes, no specific training was conducted such that the two n-Force variants gave the same results. Execution of all tools was stopped after two hours.

		n-Force No P.T.		n-Force P.T.		EDH		FPA	
Vertices	Edge	Cost	R.T.	Cost	R.T.	Cost	R.T.	Cost	R.T.
20	[20-36]	95.17	0.21	92.70	55.15	109.40	**0.076**	**86.94**	2.10
25	[30-49]	173.61	0.236	169.90	129.12	228.77	**0.17**	**165.27**	84.35
30	[46-89]	252.49	**0.31**	247.61	131.70	350.43	0.405	**241.27**	233.61
35	[47-115]	363.52	**0.40**	365.95	329.66	378.155	0.77	**347.18**	949.34
40	[86-114]	540.74	**0.52**	517.85	272.19	667.69	1.19	**510.86**	912.648
50	[142-204]	908.24	**0.79**	891.87	366.38	961.19	8.37	**880.74**	1523.21
60	[273-335]	1510.30	**1.10**	1510.30	1.17	1549.06	49.58	**1498.30**	3160.32
70	[223-438]	1853.43	**1.56**	1853.43	1.66	**1852.086**	73.32		
80	[313-509]	**2348.18**	**1.98**	**2348.18**	2.064	2449.92	307.21		
90	[417-641]	3252.69	2.54	3252.69	2.56				
100	[525-1220]	3833.84	3.29	3833.84	3.11				
110	[526-1378]	4840.47	3.91	4840.47	3.86				
120	[770-1573]	5621.08	4.62	5621.08	4.60				
130	[807-1773]	6928.51	5.71	6928.51	5.76				
140	[890-1440]	7327.50	6.84	7327.50	6.85				

As shown in Table 5.1, n-Force without parameter training (P.T.) was fastest. With P.T. running time increased slightly. Note that P.T. was performed only for smaller problem instances (up to 50 nodes; see algorithm description). Thus, running times and costs were the same for larger graphs, since P.T. was switched off. n-Force is generally fastest, followed by edge deletion heuristic (EDH) and the exact fixed-parameter algorithm (FPA). As FPA is an exact algorithm, it always yielded the smallest editing costs. For larger problem instances, FPA did not terminate anymore within two hours.

Compared to the edge deletion heuristic (EDH), n-Force yielded editing costs closer to those of fixed-parameter algorithm (FPA). Particularly, on the graphs with

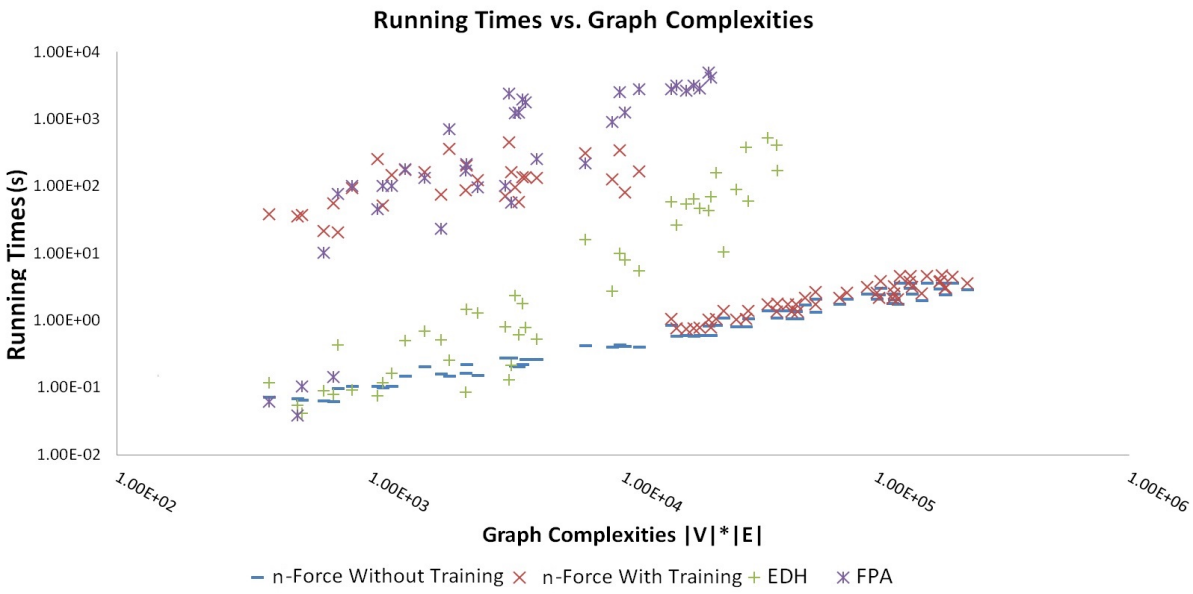

Figure 5.4: Running times against graph complexities. The running times are plotted against the graph complexities of the input instances ($|V| \cdot |E|$). Note the effect of the parameter training (P.T.) of parameters, which is turned of for larger graphs (see text).

30 and 40 vertices, n-Force algorithms (with and without P.T.) gave editing costs around 40% and 30% smaller than those of EDH (38.79% and 41.53% for 30 nodes, 23.48% and 29.94% for 40 nodes). Generally, the n-Force algorithm with P.T. output smaller results than without P.T..

We also compared the running times against graph complexities (see Fig. 5.4). Here "graph complexity" refers to the product of the number of nodes and the number of edges in a given graph. Clearly, n-Force outperforms the other two algorithms.

The accuracy of n-Force (with and without P.T) is plotted against that of the EDH heuristic as a function of the differences in editing costs between the heuristic (EDH and n-Force) and the exact algorithm (FPA) in Fig. 5.5 for smaller graphs (where FPA terminated). n-Force clearly achieved better overall editing costs than EDH. With the standard parameter set obtained from general training, n-Force managed to achieve smaller costs than EDH. Nevertheless, in many cases, n-Force with P.T.

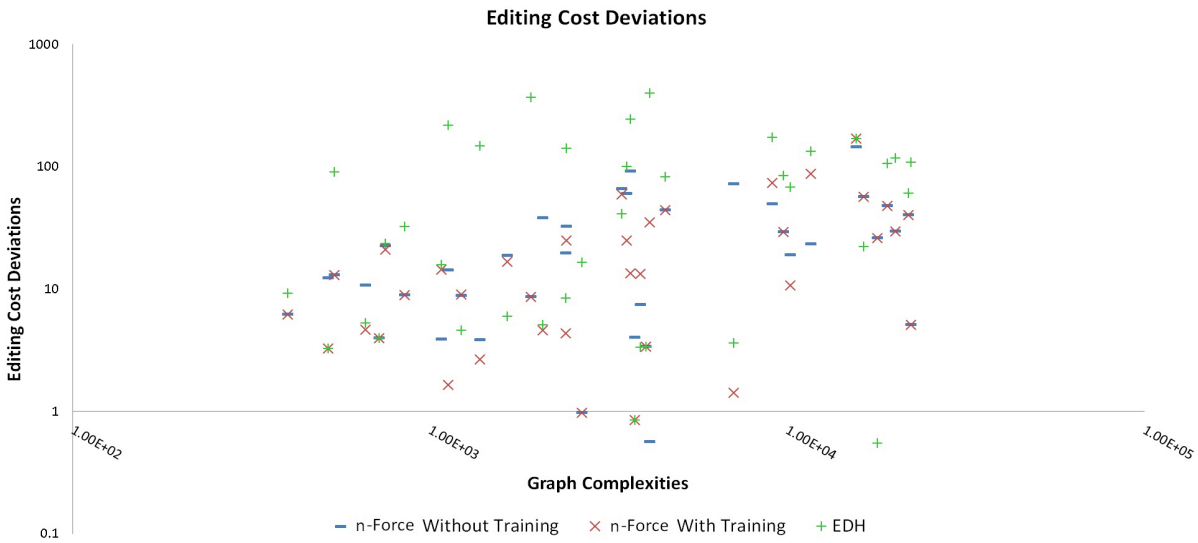

Figure 5.5: Deviations (in editing costs) from the optimal solution of the FPA algorithm.

returned solutions with lower cost. This justifies our strategy to evolutionarily train the heuristic parameters on small problem instances (where this can be achieved fast) and relying on the assumption that a parameter set will work for larger connected components of the same graph similarly well.

5.6.2 Robustness

Fig. 5.6 illustrates the robustness of n-Force to varying error-edge-rates. Artificial graphs with seven different error-edge-rates were generated, with 10 repeats for each rate. We now compare the editing costs and the running times of n-Force and EDH on these artificial data sets. As expected, Fig. 5.6 (a) shows that with increase of error-edge-rates, the editing costs for both algorithms increases polynomially as well. Fig. 5.6 (b) shows that the running times of both tools were generally quite robust regarding changing graph structures and the running time correlates only with the input sizes.

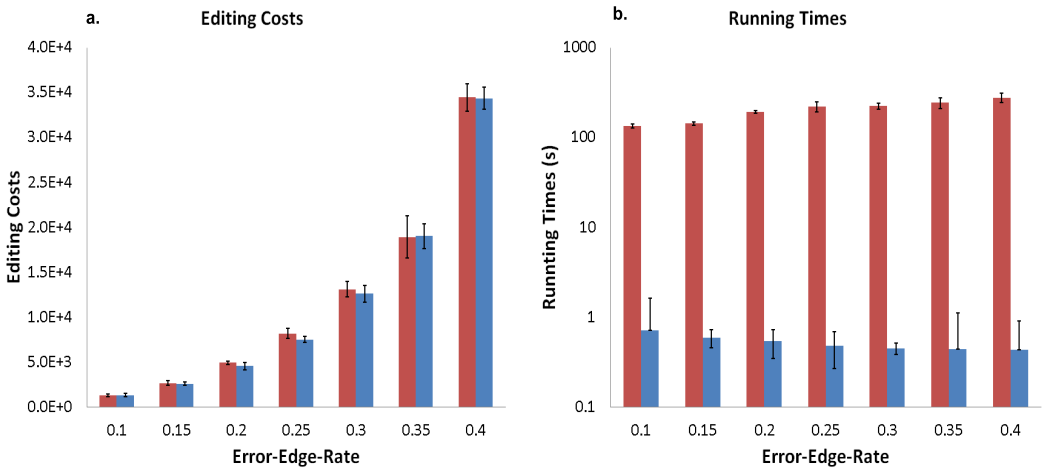

Figure 5.6: Robustness of the Edge Deletion Heuristics (*red*) and n-Force (*blue*). Input graphs are generated with different error-edge-rates and the running times are measured. Note the *log*-scales of the y-axes.

5.7 Conclusion

We discussed n-CluE, a software package dedicated to solve (weighted) bi-/n-cluster editing problems. It offers three algorithms: a fixed-parameter algorithm, an edge deletion heuristic and a graph-layout motivated heuristic named n-Force. The performances of the three algorithms were evaluated and compared on artificial graphs. We showed that n-CluE is able to deal with graphs of various sizes and to meet different requirements. The running times of the fixed-parameter algorithm explode when the input size exceeds a certain value (50 vertices) while the edge deletion heuristic still works fine for graphs of larger sizes. Also the two heuristics are very robust against different levels of noise. The results of the evaluation indicate that our software package is powerful enough to deal with real-world data sets. In the next chapters, we apply our tool to various biomedical data sets.

Chapter 6

Prediction of Novel Genotype-Phenotype Associations

In this chapter, we applied n-CluE to Genome-Wide Association Study (GWAS) data sets to detect novel associations between genotypes and phenotypes. Two of the three n-CluE algorithms were applied to two GWAS data sets and discovered. 86 novel associations.

The content of this chapter is based on the published research articles listed below:

- Peng Sun, Jiong Guo, and Jan Baumbach. Integrated simultaneous analysis of different biomedical data types with exact weighted bi-cluster editing. *J Integr Bioinform*, 17, 2012

- Peng Sun, Jiong Guo, and Jan Baumbach. Biclue-exact and heuristic algorithms for weighted bi-cluster editing of biomedical data. In *BMC proceedings*, volume 7, page S9. BioMed Central Ltd, presented at GLBIO2013, 2013

6.1 Introduction

Genome Wide Association Study (GWAS) presents one of the fastest emerging areas of today's biological research. These studies examine the co-occurrence (association) of genetic variants (genotypes) with a certain phenotypic trait. Typically, millions of single-nucleotide polymorphisms (SNPs) are investigated as genetic variants and major diseases are examined as traits. These studies normally compare the genotypes of two groups of people: healthy people (controls) and diseased people (cases). Then statistical tests are used to verify if there is any significant association. This is a typical example of a bipartite data type, i.e. two types of measurements and relations between concrete instances of the two types.

Since the first GWAS was published in 2005 on age-related macular degeneration [102], the number of GWAS publications has been growing dramatically. Up to June 2014, there have been $1,751$ publications on GWAS, according to National Human Genome Research Institute (NHGRI) Catalog of Published Genome-Wide Association Studies [196]. Although the discovered associations have revealed many disease – traits associations, yet how the interactions of the genes confer a risk to diseases still remains widely unclear. Traditional analysis methodology of GWAS associates one pair of SNP and phenotype in one statistical test, which tends to incur false positives and false negatives. Moreover, many gene/SNP markers, conferring a low or moderate risk by themselves, interact with each other and have a significant combined risk. Hence, these markers often fail to be detected. Novel computational approaches considering combined effects in analysing GWAS data might provide more meaningful results and insights.

Here, GWAS associations are modeled as graphs, where vertices correspond to SNPs (genotypes) and traits (phenotypes) while edges symbolize significant associations between them. We proceed one step further by associating a group of sequence variations (SNPs) with a group of traits/diseases, forming a "group to group" as-

sociation, rather than the traditional SNP-trait association. We applied n-CluE to different GWAS data sets and discovered new associations that have not been reported before. We believe such results, based on several associations instead of one pair-wise relation, to be of higher confidence.

6.2 Materials and Methods

In order to demonstrate the applicability of n-CluE to real world biomedical data, we studied GWAS data retrieved from two sources: (1) an online available database developed by A. D. Johnson *et al.* [97], containing 56,412 significant SNP associations with 52,554 unique SNPs and 87 different diseases/traits. (2) National Human Genome Research Institute (NHGRI) Catalog of Published Genome Wide Association Studies, an online catalogue of SNP-traits from published GWASs, with 5,476 unique SNPs and 526 different diseases [84]. The edge weights are defined as: $s(uv) = -log(P)$, (P is the p-value of the given association). We adopted the most frequently used p-value threshold of 0.05, corresponding to $log(0.05) = 1.301$ in our graph.

6.3 Results and Discussions

Due to the incompatibility of terminologies utilized in these two data sources, we did not merge them. The resulting graphs generated from our datasets contain 415 connected components in total, with 136 from the graph generated from Johnson's dataset and 279 from NHGRI dataset, respectively. Fig. 6.1 shows a histogram of the initial distribution of component sizes $|V|$.

We applied our fixed-parameter algorithm and the edge deletion heuristics separately on each disjoint connected component and identified exact solutions for 413 components (99.5% of all the components). The two algorithms gave identical results. We found in total 86 new associations that were not detected as significant in the two

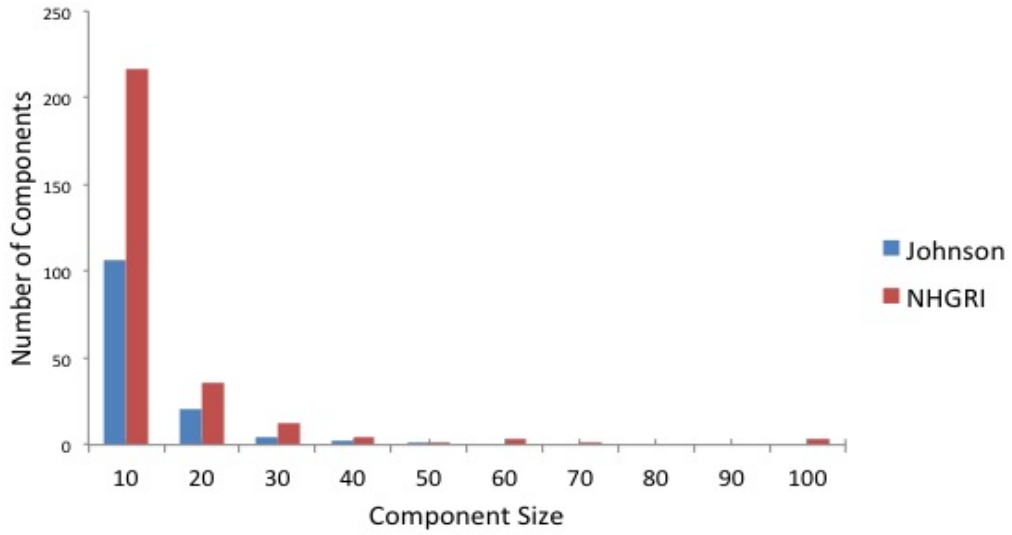

Figure 6.1: Distribution of the connected component sizes $|V|$ of the graphs generated from two GWAS data sources. The red bars represent the data from NHGRI and blue bars represent data from Johnson's online dataset. The figure does not include the two biggest connected components; one from NHGRI (3,609 vertices) and one from Johnson's online dataset (50,161 vertices).

GWAS studies. Table 6.1 shows the distribution of the new associations and their corresponding diseases/traits. For "Conduct disorder (case status)" and "Isochemic Stroke", 11 associations are found, followed by "Atrial fibrillation/atrial flutter" and "Permanent tooth development", each of which have 10 new associations. Note that our predictions are largely related to the user-given similarity threshold, i.e. 0.05 in our studies.

We applied n-CluE algorithms to two different GWAS datasets. Our results show that the algorithms work well on most of the GWAS data, finding 86 new associations in total. These newly discovered associations might be useful as guidelines for further wet lab studies. Although the best way of estimating the accuracy of our method is to verify the newly discovered associations experimentally, yet by comparing the original associations and the new ones, we might be able to assess the confidence of our results. Results show that our algorithm clustered related phenotypes together, i.e.

most of the SNPs we found to be associated with new phenotypes were previously reported to be associated with related phenotypes. For instance, rs10033464 was reported previously to be associated with "atrial fibrillation/atrial flutter" and in our results we found it associated with "atrial fibrillation". rs17145713, rs1158867 and rs6120849, which we assigned to be associated to "plasma coagulation factors", are labeled with "Plasma levels of Protein C" previously (Protein C is one of the important plasma coagulation factors [127]). Besides, our algorithm clustered the phenotypes that were found to be related by clinical studies. The 11 new SNPs we identified to "isochemic stroke" are originally tagged as associated with "atrial fibrillation" and it's been reported that atrial fibrillation can increase the risk of isochemic stroke [197]. These results might imply the confidence of the newly discovered associations before any experiments performed for verification.

Table 6.1: New associations obtained using biclustering editing. The items with "*" come from the NHGRI data set while the remaining emerge from an online data set by Johnson *et al.* [97].

Traits/Disease	No. of Newly Found Associations
Conduct disorder (case status) *	11
Ischemic stroke	11
Atrial fibrillation/atrial flutter*	10
Permanent tooth development*	10
Conduct disorder (symptom count) *	9
Primary tooth development (time to first tooth eruption) *	8
Cleft lip*	7
Primary tooth development (number of teeth) *	5
Alcoholism (alcohol dependence factor score)*	4

Plasma coagulation factors*	3
Vitamin D insufficiency*	3
Vitamin D levels*	2
Atrial fibrillation*	1
Nonsyndromic cleft lip with or without cleft palate*	1
Plasma levels of Protein C*	1
Total	86

Chapter 7

Biclustering of Gene Expression Data

As described in the first chapter, gene expression matrices form a perfect example of bipartite graphs. Their hidden information is better extracted by simultaneous clustering of both genes and conditions. In this chapter, to demonstrate the strength of the bicluster editing model on gene expression data mining, we applied n-Force, the layout-based heuristic, in the n-CluE package to artificial and real-world gene expression data sets and compared the results with eight existing biclustering tools. Our results indicated that the model of bicluster editing perfectly fits the scenario of gene expression data mining and our heuristic algorithm can be applied on large-scale data sets and finishes within reasonable running times.

The content of this chapter is based on the published research articles listed below:

- Peng Sun, Nora K Speicher, Richard Röttger, Jiong Guo, and Jan Baumbach. Bi-force: large-scale bicluster editing and its application to gene expression data biclustering. *Nucleic Acids Research*, page gku201, 2014

- Peng Sun, Jan Baumbach, and Jiong Guo. Efficient large-scale bicluster editing. In *German Bioinformatics Conference 2014*, pages 54–60

7.1 Introduction

Given gene expression data sets for different cellular conditions, biclustering is more powerful than traditional clustering in capturing biologically meaningful subsets of condition-specific genes. Biclustering approaches are generally capable of discovering such local patterns and became increasingly popular due to their ability to simultaneously cluster biological data from different sources in order to discover local bi-correlations patterns. Several systematic comparisons have been published, using various measurements to evaluate a number of prevailing biclustering tools on both synthetic and real-world data sets [53, 149, 188].

7.2 Materials and Methods

7.2.1 Synthetic Data Matrices

We only applied n-Force to the microarray data sets, for the large sizes of the input graphs cannot be finished within reasonable times (two hours) by fixed-parameter algorithm or edge deletion heuristic. For a comprehensive comparison of the performance between n-Force and eight other biclustering tools, we created synthetic data matrices based on six different models. Each synthetic data matrix consists of 300 rows and 200 columns, within which a pre-defined bicluster with 30 rows and 30 columns was randomly selected. For each of the following models, 10 data matrices were generated for repetitive simulation. With this strategy we generally followed the protocol suggested by Eren *et al.* [53].

- Constant biclusters: the values in the matrix of randomly selected 30 rows $\times$ 30 columns bicluster were set to a constant expression level of 0. The background values, i.e., the elements in the matrix that are not within the pre-defined

bicluster were chosen randomly but independently from Gaussian distribution $N(0, 1)$.

- Constant-upregulated biclusters: as in the previous model but the expression levels in the 30×30 bicluster were fixed to 5, i.e., simulating constant-upregulation.

- Shift-scale biclusters: Before generating each data matrix, a base row $R_b = \{a_{b,1}, a_{b,2}, ..., a_{b,200}\}$ was chosen. For every row r_i in the pre-defined bicluster, a scale factor α_i and a shift factor β_i were randomly generated. Each element a_{ij} in the pre-defined bicluster was both shifted and scaled from the base row: $a_{ij} = \alpha_i \cdot a_{bj} + \beta_i$. The selected rows in the pre-defined bicluster could be positively or negatively shifted (or scaled), depending on the values of the shift (or scale) factors. The elements in base row and background were drawn independently from Gaussian distribution $N(0, 1)$. All scale factors and shift factors were drawn independently from distribution $N(3, 1)$.

- Shift biclusters: similar to the Shift-scale model, but with fixed scale factors of $\alpha_i = 1$.

- Scale biclusters: similar to Shift-scale model, but with fixed shift factors of $\beta_i = 0$.

- Plaid biclusters: this model is an additive bicluster model, first introduced in [114]. Each matrix element is modeled as the sum of several different effects, including background effect θ, cluster effect μ, row effect α, and column effect β:

$$a_{ij} = \theta + \sum_{k=1}^{K} (\mu_k + \alpha_{ik} + \beta_{jk}) \rho_{ik} \kappa_{jk}$$

, where a_{ij} is the element in the matrix, ρ and κ are the indicators for the membership in bicluster k for row i and column j. All effects were independently and identically distributed according to the Gaussian distribution $N(0, 1)$.

7.2.2 Comparison against Eight Biclustering Algorithms

To evaluate the performance of n-Force on biclustering problems, we refer to the work of Eren *et al.* [53]. Eight (out of twelve) prevalent online available biclustering algorithms were downloaded, including: Cheng and Church [29], BiMax [149], FABIA [87], ISA [15], Plaid [114], QUBIC [117], Spectral [105] and xMOTIFs [134]. Five of the eight methods are integrated in the R package "biclust". The three remaining software packages (FABIA, ISA and QUBIC) were downloaded from the project web sites. Four other tools were not included in this study since no corresponding online resources are available or errors exist in the programs. Note that the omitted algorithms are not among the best-performing tools in the study of Eren *et al.*. The details of the biclustering algorithms including the references and the important parameters influencing the performances of the algorithms are listed in the Table 7.1. For the details of the eight algorithms, please refer to Section 2.2.4.

7.2.3 Parameters

Appropriate parameter setting is crucial to the performance of each algorithm. Algorithms cannot simply be applied with default parameters as not all of them fit all bicluster analysis scenarios. We carefully optimized the parameters of each tool such that they show their best performances on both, the synthetic data as well as the gene expression data.

For the synthetic data sets, all algorithms that require a user-given number of biclusters were given the correct number. For gene expression data, the number was set to be 50 biclusters. Parameters other than "number of biclusters" were optimized

Table 7.1: The applied biclustering tools and their parameter space.

Algorithm	References	Parameters
n-Force	–	Edge threshold: t_0.
FABIA	[87]	Number of biclusters: p.
QUBIC	[117]	Range of possible ranks: r; Percentage of regulating conditions for each gene: q; Number of biclusters: p.
Cheng and Church	[29]	Variance threshold: δ; Multi-deletion parameter: α.
Plaid	[114]	Number of max. iterations for each layer: M_I; Max. number of layers: M_L.
Bimax	[149]	Number of biclusters: n; Min. row size: $minr$; Min. column size: $minc$.
Spectral	[105]	Normalization method: $norm$; Min. row size: $minr$; Min. column size: $minc$.
xMOTIFS	[134]	Number of biclusters: n.
ISA	[15]	Number of seeds: n_s.

through performance on synthetic data sets, i.e., we tried various parameters (or combinations of parameters) for each algorithm and took the parameter (or combination of parameters) that could achieve the best performance. Particularly, for the algorithms requiring more than one parameter, a grid-search strategy was implemented.

7.2.4 Evaluation on Synthetic Data

The performance of all biclustering algorithms on synthetic data was evaluated by comparing the set of result biclusters against the pre-defined biclusters. As suggested in the work of Eren *et al.* [53], we chose the Jaccard coefficient to compute the

similarity of two different biclusters. Let b_1 and b_2 be two biclusters, we define:

$$s(b_1, b_2) = \frac{|b_1 \cap b_2|}{|b_1 \cup b_2|},$$

where $|b_1 \cup b_2|$ and $|b_1 \cap b_2|$ are the number of nodes in the union and intersection of b_1 and b_2, respectively. Obviously, the largest value of Jaccard coefficient is 1 when b_1 and b_2 are identical and the lowest value 0 is reached when two sets are disjoint. It can be interpreted as the percentage of shared elements of two biclusters.

For two sets of biclusters, the pre-defined set of biclusters T (true set) and the result set of biclusters R (from the nine algorithms), we calculated two scores: recovery and relevance scores, defined to quantify the similarities between T and R. Recovery score indicates the percentage of the truth set that is found in the result. It is maximized when $T \subseteq R$. Similarly, relevance score represents the percentage of the result set that is overlapped with the true biclusters. It is maximized when $R \subseteq T$. Formally:

$$Recovery : S(T, R) = \frac{1}{|T|} \sum_{b_1 \in T} \max_{b_2 \in R} s(b_1, b_2)$$

$$Relevance : S(R, T) = \frac{1}{|R|} \sum_{b_1 \in R} \max_{b_2 \in T} s(b_1, b_2)$$

Again, note that we are in coherence with Eren *et al.* here [53].

7.2.5 Evaluation on Real Gene Expression Data

For gene expression data, a different evaluating method must be used since true biclusters are unknown *a priori*. We validated the results by computing GO term enrichments for all the biclusters. Principle Component Analysis (PCA) imputation was used to compute the missing values in the gene expression data sets [170]. Enrichment analysis was carried out by using GOstats [54], on three categories (Biological Process Ontology, Molecular Function Ontology, Cellular Compartment Ontology).

In hypergeometric tests, genes within each bicluster were used as the input vector, and genes involved in the gene expression study were used as the gene universe. Afterwards, multiple test correction was performed to adjust the p-values by using the method from Benjamini and Hochberg [86]. A bicluster was considered "enriched" in a certain GO category if any adjusted p-value of any GO term was smaller than $P = 0.05$. Again, we agreed and followed Eren *et al.*'s suggestions with this protocol [53].

7.3 Results and Discussion

7.3.1 Synthetic Matrices

Given that n-Force, as shown in Chapter 5, solves the *Bicluster Editing Model* well enough, we now seek to apply this model to biclustering of biological data sets. As mentioned many times before, we follow the evaluation protocol published in a recent review paper from Eren *et al.* of [53]. In Figure 7.1, we compare the *relevance* and *recovery* of n-Force as well as of the above introduced existing tools. Figure 7.2 compares the running times of all the algorithms with inputs of fixed columns of 300 and rows of varying sizes.

We believe the Bicluster Editing model underlying n-Force to be more robust regarding different data set types compared to the existing biclustering algorithms. The main assumption behind bicluster editing, and thus n-Force, is that the average similarities within the biclusters are above the user-given threshold while the mean similarities between elements from different biclusters is below the threshold. This way, the threshold as single density parameter controls the size and granularity of the biclustering results. If n-Force is configured to output only the largest bicluster, it seeks the largest sub-matrix in the data set with significant difference between elements inside the bicluster compared to the background. Thus, n-Force successfully

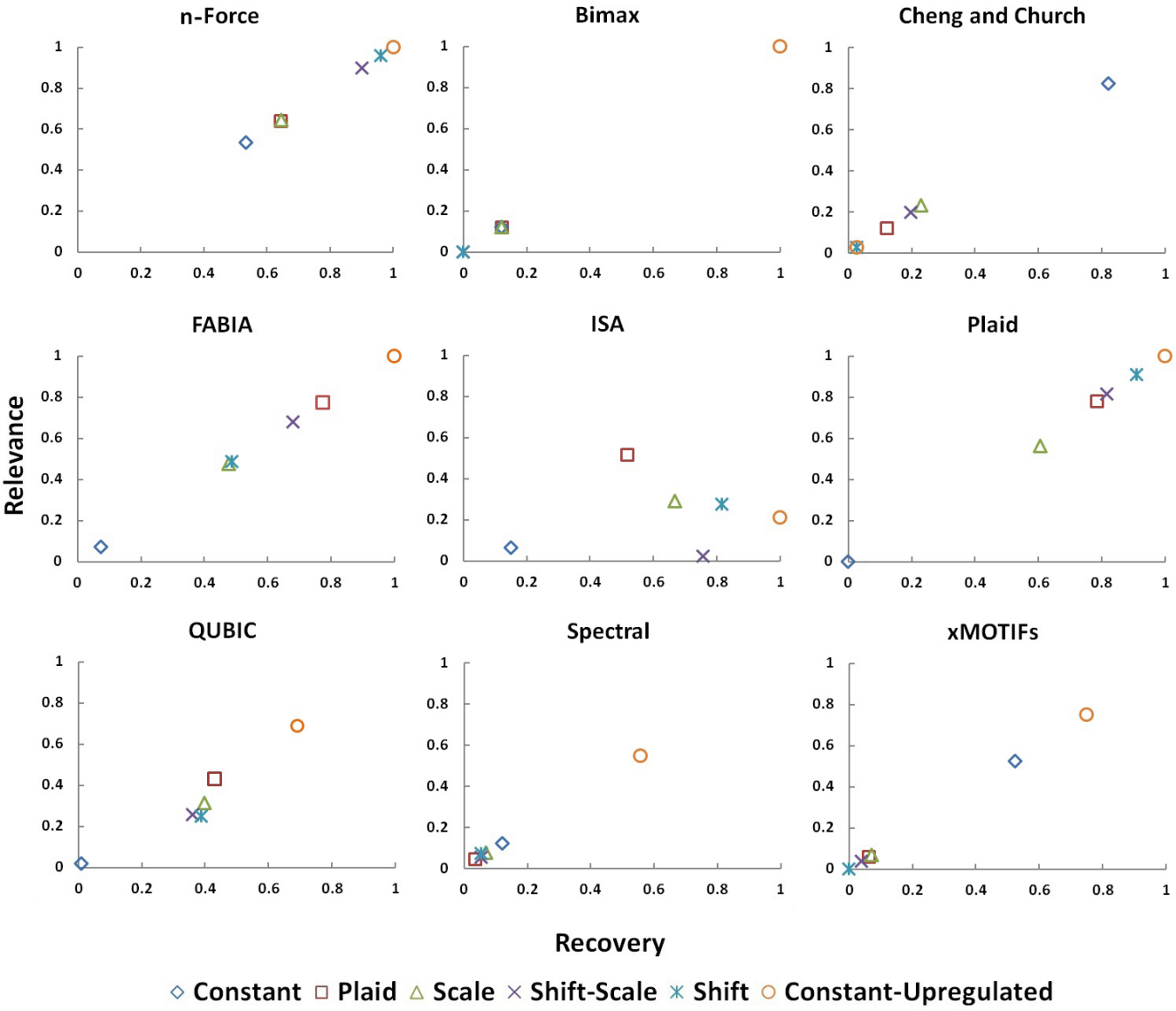

Figure 7.1: The comparison of n-Force against eight existing biclustering algorithms on synthetic data sets. Each plot includes the average recovery vs. relevance of data sets from five different data sampling models (see text).

recovered all the biclusters for the constant-upregulated model. For the inputs of shift and shift-scale model, since elements inside the bicluster were shifted by a certain magnitude, n-Force was also able to recover most of (around 85% $\sim$ 95%) the pre-defined biclusters. In the scale model where data elements were comparatively weakly

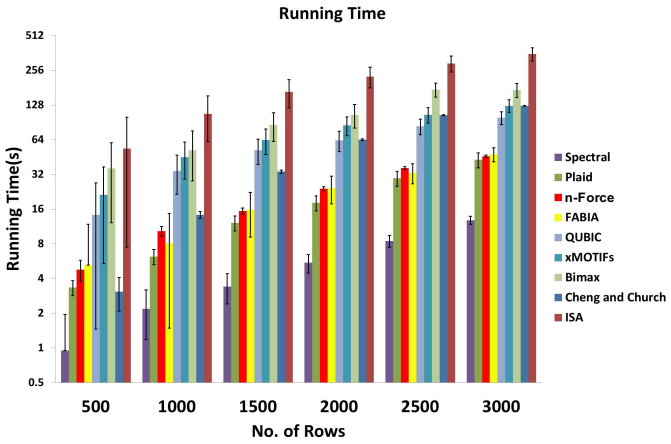

Figure 7.2: Running times of the nine algorithms for increasing number of rows in the expression matrix. The y-axis is in *log*-scale.

shifted from the background, the results were a little bit worse but still over half of the pre-defined biclusters were recovered (60% $\sim$ 70% for the scale model). For the plaid model where most of the elements were generated only based on the "background effect", we conducted biclustering to extract "high-deviated" data and over 60% of the pre-defined biclusters were discovered. For the constant model, around 55%$\sim$60% of the pre-defined biclusters were successfully captured.

7.3.2 Gene Expression Data

We now continue with the protocol from Eren *et al.* and apply all nine algorithms to real-world biological data: gene expression microarray data from the GEO database (GDS181, GDS589, GDS1027, GDS1319, GDS1406, GDS1490, GDS2225, GDS3715, and GDS3716; see Table 7.2 for a summary). Their performance was evaluated by means of GO Term Enrichment Analysis.

Before GO term enrichment analysis was performed the biclusters identified by the nine algorithms were further filtered to reduce redundancy: Biclusters with more than 80% overlap were removed. Afterwards, GO term enrichment analysis was conducted on the filtered biclusters, for all three categories (Biological Process, Molecular Func-

Table 7.2: GDS data sets

Data set	Genes	Samples	Description
GDS181	12559	84	Gene expression profiles from diverse tissues, organs, and cell lines with normal physiological state.
GDS589	8799	122	Examination of normal physiological gene expression in 11 peripheral and 15 brain regions in three common outbred rat strains.
GDS1027	15866	154	Sulfur mustard effect on lungs: dose response and time course.
GDS1319	22548	123	Various C blastomere mutant embryos analyzed to deconvolve C blastomere lineage-specific expression patterns specified by the PAL-1 homeodomain protein.
GDS1406	12422	87	Analysis of 7 brain regions of 6 inbred strains of mouse.
GDS1490	12422	150	Mouse neural tissue profiling.
GDS2225	15923	6	Mechanical strain effect on fetal lung type II epithelial cells.
GDS3715	12559	110	Insulin effect on human skeletal muscle.
GDS3716	22215	42	Breast cancer: histologically normal breast epithelium

tion and Cellular Compartment). Table 7.3 shows the number of enriched biclusters in all three categories. Figure 7.3 gives the fractions for different significance levels of the biclusters found by all algorithms. Bimax found the most biclusters, however most of them were not enriched at reasonably high p-value cut-offs. Thus the average enrichment level for Bimax is comparably low. Similarly, Cheng and Church, QUBIC and Spectral have similar problems with high numbers of false positives. In contrast, most of the biclusters found by n-Force and Plaid are highly enriched. Although xMotifs also provided many enriched biclusters, it did not find any biclusters for the data sets GDS1027, GDS1319 and GDS3715. n-Force clearly outperformed the other tools as in average approximately 55% of the reported biclusters are also enriched with high p-value confidence cut-offs, more than with the competing eight tools.

A wet lab analysis of the biological relevance of the biclusters identified is beyond the scope of this study. However, the GO terms in the enriched biclusters found by n-Force were also examined. The 12 GO terms with lowest p-values are given in Table 7.4, with 4 terms in each category. A further investigation into the results is necessary

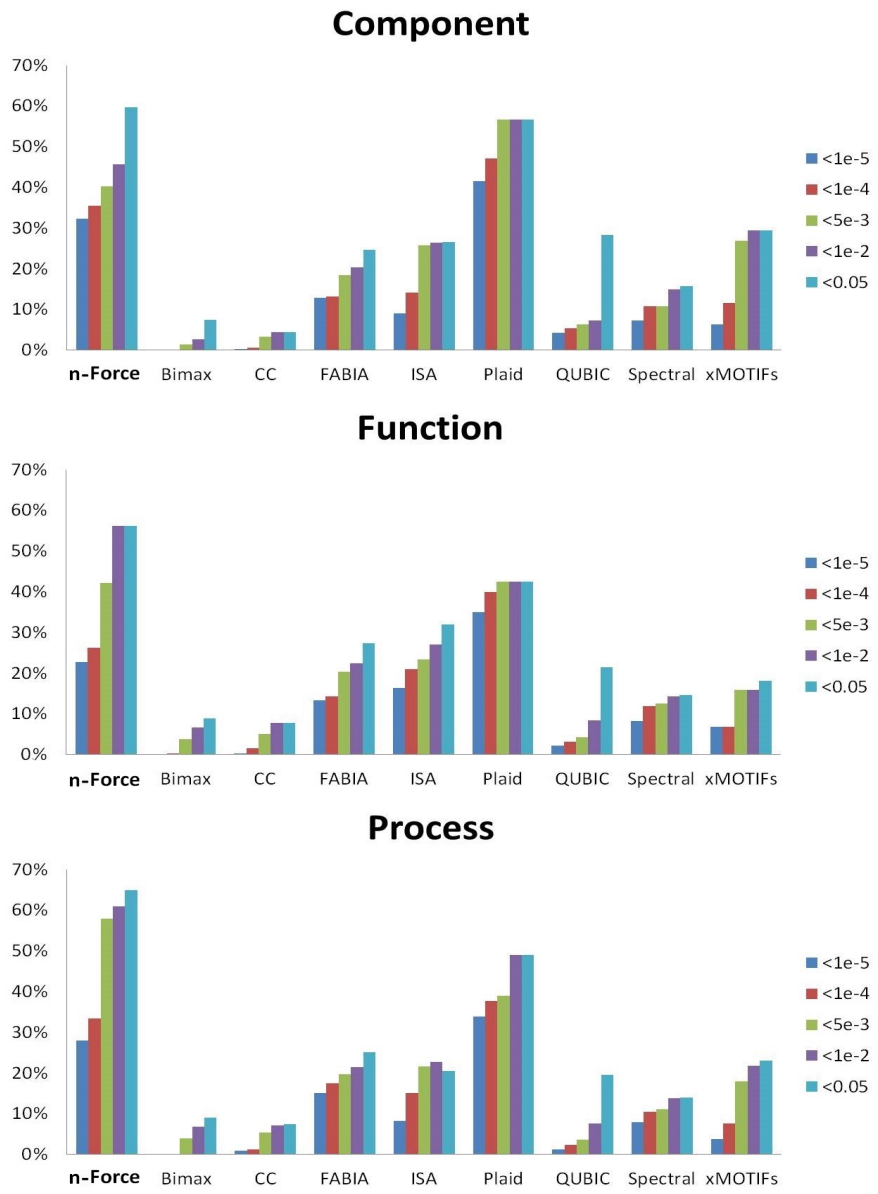

Figure 7.3: Proportions of GO-enriched biclusters for different algorithms on five significance level (see text).

to validate the biological relevance of the biclusters found. However, some of the most enriched GO terms might also be suggestive. For instance, GDS589 represents the gene expression profiles in brain and peripheral regions and thus biosynthesis is expected to be more active. n-Force identified bicluster enriched in GO:0009260, which is related to ribonucleotide biosynthetic process. Another examples comes from

Table 7.3: The results of GO enrichment analysis, including the numbers of reported biclusters and the numbers of enriched biclusters.

Algorithm	Found	Enriched (%)
n-Force	129	76(58.91%)
FABIA	189	47(24.87%)
QUBIC	873	200(22.91%)
Cheng and Church	1962	107(5.45%)
Plaid	180	87(48.33%)
Bimax	2439	205(8.41%)
Spectral	1095	161(14.70%)
xMOTIFS	339	79(23.30%)
ISA	261	67(25.67%)

the dataset GDS3716, which focuses on the analysis of histological normal breast epithelia from breast cancer patients. n-Force found biclusters heavily enriched in GO terms related to translational regulations, for instance GO:0006415.

The proportions of enriched biclusters reported by n-Force support our conclusion that the bicluster editing model is a well-working formulation for biclustering. However, the numbers of biclusters discovered by n-Force is comparably low. This might be because n-Force is no fuzzy partitioning approach such that by definition all identified biclusters disjoint from each other.

7.4 Conclusion

We compared n-Force to eight existing tools by following an established evaluation protocol from Eren *et. al.*'s review paper. We show that n-Force outperformed the existing tools on synthetic data sets and on real-world gene expression data.

Table 7.4: Four most enriched GO Term for each GO Category.

GO Term	GO Category	Dataset	P-values	Terms
GO:0006415	Biological Process	GDS1316	9.66E-50	translational termination
GO:0006613	Biological Process	GDS181	9.31E-18	cotranslational protein targeting to membrane
GO:0009260	Biological Process	GDS589	1.59E-06	ribonucleotide biosynthetic process
GO:0042274	Biological Process	GDS1027	4.10E-15	ribosomal small subunit biogenesis
GO:0044424	Cellular Compartment	GDS1319	3.10E-34	intracellular part
GO:0044444	Cellular Compartment	GDS1406	1.57E-37	cytoplasmic part
GO:0005737	Cellular Compartment	GDS1490	7.07E-50	cytoplasm
GO:0043229	Cellular Compartment	GDS2225	4.77E-16	intracellular organelle
GO:0003735	Molecular Function	GDS3715	6.57E-64	structural constituent of ribosome
GO:0003723	Molecular Function	GDS3716	1.15E-09	RNA binding
GO:0015078	Molecular Function	GDS589	1.99E-10	hydrogen ion transmembrane transporter activity
GO:0003735	Molecular Function	GDS181	2.65E-13	structural constituent of ribosome

Chapter 8

Drug Repositioning

Drug design is very expensive, time-consuming and becoming economically increasingly risky. Computational approaches for inferring potential new purposes of existing drugs, referred to as drug repositioning, play an increasingly important role in current pharmaceutical studies. Existing methods focus on chemical compound similarity, or on drug-gene and gene-disease associations. Here we first summarize the recent development of computational drug repositioning from the aspects of repurposing strategy and the data source. Second, we integrate drug-gene-disease information and derive an n-cluster editing triangulation model, which we further combine with a semantic literature mining approach. The model predicts 31,731 new drug-disease associations ("novel prediction set") of which 11,517 (36.3%) co-occur in literature ("high confidence set") with 1,382 cases where the drug is explicitly mentioned to treat the disease ("treats annotation set"). Model robustness was evaluated systematically by repeatedly removing and perturbing known drug-disease pairs. In conclusion, we suggest that the utilization of drug-gene-disease triangulation coupled to sophisticated text analysis provides a robust approach for identifying new drug candidates for repurposing. We anticipate this to be highly useful for treatment alternative identification and cost reduction.

The content of this chapter is based on the research article listed below:

- Peng Sun, Jiong Guo, Rainer Winnenburg, and Jan Baumbach. Integrated literature mining and drug-gene-disease triangulation reveals ten thousand new purposes for existing medication. *Drug Discovery Today*, (In press), 2016

8.1 Introduction

8.1.1 Challenges in Drug Development

The pharmaceutical industry is facing great challenges emerging from a decreased speed in the discovery of new drugs and drug targets caused by various reasons. Stricter regulations and scrutiny have been imposed on pharmaceutical companies due to the rising safety concerns and the transparency of pharmaceutical manufacturers, which has a constant negative impact on the industry [7]. The major problem, however, is the lack of productivity in Research & Development which greatly slows down the increase of profitability [143]. Although the number of approved drugs resurged in 2015 [132], it is accompanied with continuously rising costs [146]. The classic conservative drug development strategy, limited to "one drug, one target" paradigms, does not consider or evaluate the "off-target" effects or the probability of multiple drug indications, yet some of them have proven successful in the market later. Sildenafil and minoxidil are well-known examples. They have been repurposed for the treatment of erectile dysfunction and hair loss, respectively. Similar examples also include: ropinirole, originally developed for anti-Parkinson disease but later found effective against Restless Legs Syndrome and SSRI-induced sexual dysfunction [142]; bevacizumab, originally developed to resistant metastatic tumor cells, has been proven effective to treat abnormal retina vascularization [154].

Drug repositioning has strong potential to provide promising solutions in current drug design. The development cycle through repositioning may be reduced by as long

as five years compared to the traditional drug discovery pipelines [52]. Moreover, repositioned drugs incur significantly reduced safety risks for patients, as almost all known drugs have been thoroughly studied with respect to their toxicity, their metabolism, and possible side effects in human [48].

Successful drug repositioning stories are rare and rather random events [48]. Well-known examples are either accidentally discovered side effects, or based on extensive research on drug properties, which is infeasible in general and much too expensive to be applied on a large scale [167]. Thus, a major medical bioinformatics challenge is to predict promising drug repositioning candidates for pharmaceutical screening, laboratory tests and clinical trials. Most existing methods for computational drug repurposing follow one of the two major strategies: "drug-based" and "disease-based" approaches, depending on the data sources. They predict putative novel drug indications by exploiting detailed information of either drugs or diseases [49]. Such studies mainly focus on mining the shared properties between two drug molecules including structures [68, 75] and side effects [191]. Other methods approach the problem by computing drug-target binding properties [48] or searching for similar molecular activities [93]. The existing studies yielded fruitful and insightful results. Yet, they exclusively focus on one aspect of drug repositioning: either the drug, the target (gene), or the disease. More recent studies have proven the potential of combining some of the different data types by using computational information fusion [38, 133, 152]. Here, we show the recent progresses of data mining and data integration in the computational drug repositioning research, followed by a detailed description of how n-CluE can be used to integrate the information of drug, gene and disease networks.

8.1.2 Repositioning Strategies

Methods Based on Drug Structures

A number of publicly available databases provide massive amounts of data on drug molecular structures, chemical properties and high-throughput screening results pertain to drugs [139, 166, 181], offering great opportunities to perform structure – property analysis useful for drug repurposing. The rationale behind this strategy is the "Structure Determines Properties" paradigm, i.e., molecules with similar structures tend to have similar chemical properties and, thus, act similarly in biological system. A variety of similarity measures based on different structural features have been used to compute the similarity of two drug molecules. Such efforts include the widely used chemo- and bioinformatics library Chemical Development Kit (CDK) [171] which provides implementations for many common methods in structural chemistry/biology studies. Likewise, Swamidass *et al.* [181] constructed a drug – target network based on structural similarity which leads to potential drug repositioning provided the situation that multiple drugs or diseases share same target. A more recent trend demonstrated the benefit of integrating chemical information with other properties for computational drug repositioning. For example, Wang *et al.* [194] reported a support vector machine (SVM)-based model named PreDR implementing a customized kernel function to predict novel drug – disease associations. PreDR integrates chemical structure, molecular activity and phenotype information, such as side-effects. Its performance has been assessed by comparing to other similar methods. Similarly, Tan *et al.* [182] integrated chemical structure, drug-target binding information and gene similarity for drug repurposing inferences.

Methods Based on Omics Data

The fast growth of omics data provides an unprecedented opportunity for computational biology to reveal more insights in drug behaviors and disease mechanisms. Genome-wide expression data, in particular, are widely used to profile the effect of drug activity and have been explored for potential drug repurposing. The Connectivity Map (CMap) project by Lamb *et al.* [113] is one of the remarkable efforts aiming to construct a systematic map of the functional associations among diseases, genetic perturbation and drug behaviors, based on the genome-wide expression profiles of the human cancer cells injected with different drugs and bioactive molecules. Thus, CMap enables systematic comparison of drug associated gene expression profiles. Dudly *et al.* [49], for instance, followed the CMap's strategy and computed a therapeutic score for every drug repositioning candidate for Inflammatory Bowel Disease. The statistical significance was then derived based on a randomization model using the therapeutic score. *In vivo* model validation was performed for the most promising drug repositioning candidate.

Keiser *et al.* [98] have developed a systematic tool (Similarity Ensemble Approach, short: SEA), to compute the drug target similarity by comparing the profiles of the binding ligands. The drug off-target effects were then derived and captured from the ligand-based target similarity. The top-scored repositioning candidate was later validated in an *in vivo* rodent model.

Methods Based on Phenotypes

Drug-related phenotypes also provide valuable insights in order to profile drug effects. Ye *et al.* [211] constructed a drug-drug network from clinical side-effect information to generate putative drug-disease associations with the underlying hypothesis that shared side-effect profiles may lead to shared indications. Yang *et al.* [209] also

integrated side-effect profiles into drug repositioning features and built a Naïve Bayes model to make suggestions of new drug uses.

Investigating the disease similarity is also a promising approach to identify drug repurposing opportunities, based on the hypothesis that similar diseases may have similar therapies. Chiang *et al.* [31] derived disease similarities from the shared treatments, and subsequently executed a guilt-by-association approach to predict new drug indications.

The recently developed concept of phenome, defined as the comprehensive set of expressed phenotypes, provides new insights for drug repurposing. Phenome-wide association studies (PheWAS), dedicated to systematically investigating the genotype-to-disease associations, have proven their great potential in discovering the genetic profiles of diseases [81]. Such analyses enhance the genotype-phenotype associations detected by other studies (e.g. genome wide association study, GWAS, see Chapter 6) and shed new light on drug repositioning. Rastegar-Mojarad *et al.* [152] constructed a phenotype-genotype-drug network using PheWAS data to identify multiple diseases sharing common genetic etiology, which may be as well treated by the same drug.

8.2 Materials

Table 8.1 summarizes the utilized data repertoire and data sources. Here, we give a more detailed description on how the data was retrieved and pre-processed.

8.2.1 Drugs

Information on 1,543 drugs was collected from the database Drugbank (Version 4.3) [198]. We only consider approved drugs for our analysis here. Other types, e.g. investigational or withdrawn drugs, have not been included. The chemical and the molecular properties of the drugs are represented by the canonical Simplified Molec-

ular Input Line Entry Specifications (SMILES) [6], which are afterwards hashed into chemical fingerprints using the Chemical Development Kit (CDK) [171]. Our pipeline then uses the well-known Tanimoto coefficient to compute similarities between two hashed fingerprints. It is defined as the size of the shared structures/substructures divided by the union of the two fingerprints, as follows:

$$sim(x_1, x_2) = \frac{|f(x_1) \cap f(x_2)|}{|f(x_1) \cup f(x_2)|} \tag{8.1}$$

where $f(x)$ refers to the fingerprint of drug. The resulting values form a symmetric similarity matrix, which can be interpreted as weighted similarity graph with drugs as nodes and edge weights ranging from 0 (totally dissimilar drugs) to 1 (identical drugs). We applied Transitivity Clustering (TC) [201] to this matrix using a refined threshold described in the following section, resulting in pre-clusters of similar drugs. Note that TC guarantees that the average "Tanimoto similarity" within pre-clusters is above the threshold, while the average similarity of drugs from different pre-clusters are below. We executed the same pre-clustering procedure for the gene and disease data sets as well (described below).

8.2.2 Genes and Drug-Gene Pairs

It has been widely argued that the drugs sharing similar targeted genes tend to have common therapeutic functions [101, 111, 118], an assumption that we exploit systematically with our drug-gene-disease triangulation methodology. First, we extracted 11,802 drug-gene associations from Drugbank [198] together with 1,622 target genes (sequences from NCBI) [14], also refer to summary Table 8.1. We computed a pairwise similarity matrix between all target genes by performing a reciprocal all-vs-all BLAST amongst all amino acid sequences using E-value cutoff of 0.01. The $-log_{10}$ of the higher BLAST E-values was used as similarity and, hence, as edge weights in the

corresponding gene similarity graph. This graph was processed with TC using the homology detection protocol described by Röttger *et al.*'s [156] in order to identify pre-clusters of homologous genes.

8.2.3 Diseases

3,407 diseases/phenotypes were mined from the Comparative Toxicogenomics Database (CTD, released on Jan 26, 2015) [40], mapped to Concept Unique Identifiers (CUIs) using the Unified Language System (UMLS) [19]. UMLS (release 2015AA) is a knowledge framework of over 100 controlled terminologies and vocabulary used in biomedical research, including useful nomenclature ontologies such as Medical Subject Headings (MeSH, released on May 11. 2015) [119] and Systemized Nomenclature of Medicine-Clinical Terms (SNOMED-CT, released on March 2015) [116]. Pairwise disease similarities were derived by adopting the scoring scheme developed by Pedeson *et al.* [144] that uses the Latent Semantic Indexing method [43] by representing CUIs as co-occurrence vectors to profile the contexts of the medical terms. Similarities of the medical terms are then calculated as the cosine angle between the corresponding conceptual vectors. This vector-based approach, extending Latent Semantic Indexing and Latent Semantic Analysis, has proven useful in biomedical data retrieval [34] and clinical data mining [35], and it was shown to be effective compared to alternative methodologies [89]. Applying this strategy to compute all-vs-all similarities across the disease data set resulted in similarity values ranging from 0 to 1. Again we used TC (threshold 0.9) to derive pre-clusters of diseases.

8.2.4 Drug-Disease Pairs

We retrieved 271,169 drug-disease associations from (CTD) [40] with links to our disease data set. We used this later on as Gold Standard Positives (GSP) data set

Table 8.1: Data sources used for triangulation.

Nodes and Edges	Sizes	Source(s)
Drugs	1,543	Drugbank [198, 106]
Genes	1,622	Drugbank [198, 106]
Diseases	3,407	Comparative Toxicogenomics Database(CTD) [39], OMIM [76], MeSH [119]
Drug-gene associations	11,802	Drugbank [198, 106]
Drug-disease associations	271,169	CTD [39], Drugbank [198, 106]
Gene-disease associations	1,535,509	CTD [39], OMIM[76]

when evaluating the robustness of our strategy. CTD identifies diseases with various names and heterogeneous identifiers, mainly from Online Mendelian Inheritance In Man (OMIM) [76] and MeSH [119]. In order to handle the inconsistency caused by this heterogeneity, we map disease terms to CUIs in the UMLS (version 2015AA). Because MeSH is one of the vocabularies that are integrated into the UMLS, we retrieve UMLS CUIs for MeSH terms (MeSH Unique IDs for Main Heading and Supplementary Concept terms) accessing the UMLS terminology services (UTS) through their JAVA API.

8.2.5 Gene-Disease Pairs

We extracted $1,535,509$ gene-disease associations for the genes and diseases of this study (see above) from CTD and OMIM (released on April 2015).

8.3 Methods

8.3.1 Triangulation

All the above described data sets have been integrated and combined into an 3-partite graph, with nodes typed as drugs, genes and disease, respectively, and edges between them referring to drugs targeting genes, gene-disease associations, and drugs being registered to treat diseases. See Fig. 8.1 for an illustration. We refer to this network

either as "3-partite graph" or as "integrated network", dependent on the context. It is built upon the pre-clusters described in the above sections, where the edge weights are defined as the Jaccard Index of the two connected pre-clusters. Note that edges only exist between pre-clusters coming from different entity sets. For arbitrary two pre-clusters $C_1 = \{a_1, a_2, \ldots, a_m\}$,$C_2 = \{b_1, b_2, \ldots, b_n\}$, where a_i and b_i refer to the members in the pre-clusters, the edge weight w_{C_1,C_2} is defined as:

$$w_{C_1,C_2} = \frac{|E_{C_1,C_2}|}{mn} \tag{8.2}$$

where,

$$E_{C_1,C_2} = \{(a_i, b_j)|(a_i, b_j) \in E\} \tag{8.3}$$

Here, E stands for the union of the sets of drug-gene, drug-disease, and gene-disease pairs. Then the n-Force algorithm in n-CluE (see Chapter 5 for details) was utilized for computing the n-clusters in the integrated drug-gene-disease network.

The resulting n-clusters are collected and the inserted edges between drug pre-clusters and disease pre-clusters are extracted. If drug pre-cluster $D = \{d_1, d_2, \ldots, d_n\}$ and disease pre-cluster $K = \{k_1, k_2, \ldots, k_m\}$ were connected, where d_i and k_j are the drugs and the diseases within the pre-clusters, then we regarded all drug-disease pairs of (d_i, k_j) in D and K are associated. These pairs are reported as putative novel drug-disease associations, and we refer to this whole prediction process as "triangulation". Suggested drug-disease associations are then post-processed by using our literature mining engine (see corresponding sections below).

8.3.2 Parameter Optimization

As choosing the optimal parameters is crucial, we have tested a range of different configurations. For each of the three pre-clustering (drug, gene and disease networks) as well as the clustering of the integrated network, n-CluE requires one density param-

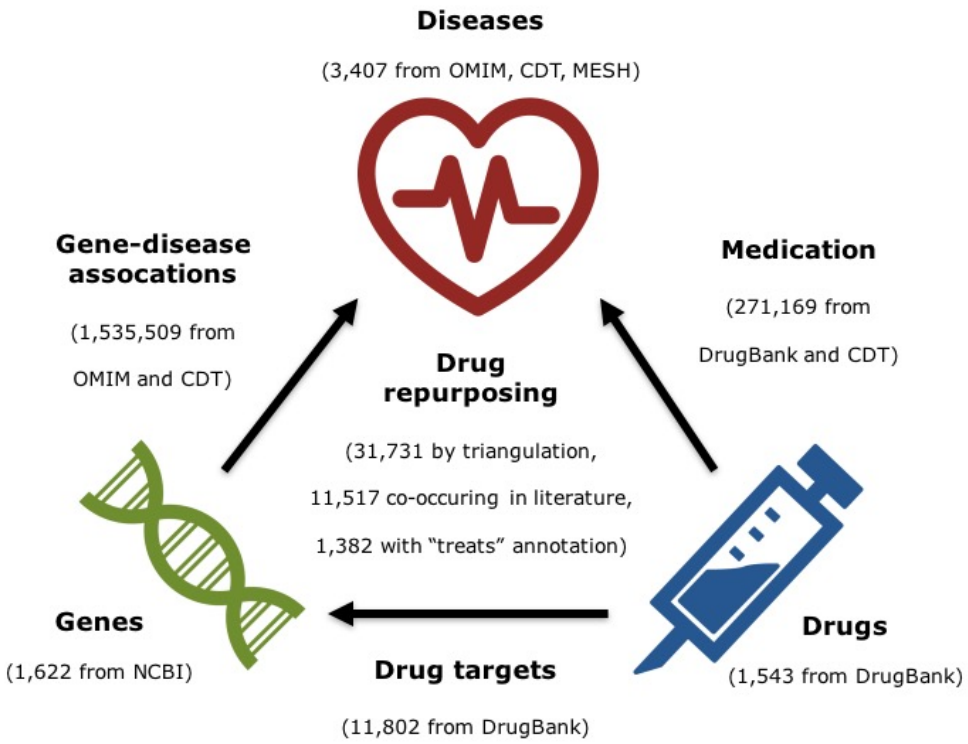

Figure 8.1: Overview about the input data, the main prediction principle and the results.

eter, resulting in four density parameters. Finding optimized density parameter for gene networks has been thoroughly discussed in Röttger *et al.* [156] and in our case, the threshold was accordingly set to the optimized value of 37. To optimize the two density parameters for drug and disease pre-clustering a grid search was performed. For each density parameter in drug and disease pre-clustering, we searched among the seven thresholds $d \in \{0.5, 0.6, 0.7, 0.8, 0.85, 0.9\}$. A 10-fold cross-validation was performed by removing 10% of the drug-disease associations in the input graph and by applying afterwards the n-CluE on the resulting graphs. We tried to recover the removed drug-disease associations by n-clustering. For each combination of drug and disease network density parameters, with varying Jaccard Index thresholds, the Receiver Operating Characteristic curve (ROC curve) was drawn and the Area Under Curve (AUC) was computed, as shown in Table 8.2. The best AUC (0.864) occurred

Table 8.2: The AUCs obtained by varying the thresholds for drug and disease pre-clusterings. A grid search is conducted with pre-clustering thresholds selected from {0.5,0.6,0.7,0.8,0.85,0.9,0.95}, generating 49 combinations. The largest 3 AUCs are bolded and colored in red. 0.9 is chosen for both drug and disease pre-clustering.

Disease thresholds \ Drug thresholds	**0.5**	**0.6**	**0.7**	**0.8**	**0.85**	**0.9**	**0.95**
0.5	0.686	0.680	0.686	0.725	0.733	0.739	0.768
0.6	0.688	0.687	0.702	0.720	0.729	0.740	0.768
0.7	0.707	0.715	0.705	0.733	0.735	0.740	0.769
0.8	0.717	0.726	0.734	0.762	0.768	**0.792**	0.767
0.85	0.741	0.750	0.757	0.762	0.765	0.772	0.768
0.9	0.726	0.738	0.747	0.749	**0.781**	**0.864**	0.766
0.95	0.734	0.722	0.713	0.722	0.725	0.772	0.766

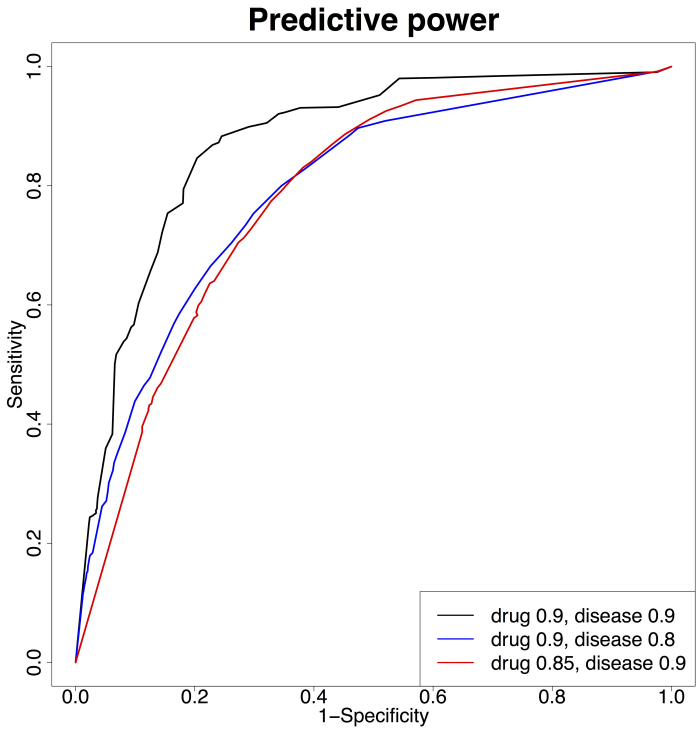

Figure 8.2: The predictive powers of n-CluE for the three threshold combinations with best AUCs.

when the density parameters were 0.9 for both drugs and disease pre-clustering. The Jaccard Index threshold in the integrated clustering was then fixed to 0.4 which optimizes the F1 score (the harmonic mean of precision and sensitivity, data not shown).

8.4 Results

8.4.1 Robustness Analysis

The classical graph cluster editing model has shown excellent robustness against missing data [205]. Here, we carry out two strategies to assess the robustness of the extended n-cluster editing model (evaluated using n-CluE) in different aspects. First, 10%, 20% and 30% of the edges in the integrated network (i.e. associations between drugs, genes and disease) are removed. Second, 10%, 20% and 30% of the drug-disease associations are removed. The known drug-disease pairs serve as positive test set. The negative test set is generated by the following procedure: first, we randomly selected a set of drug-disease pairs (detracted of the positive pairs) ten times as large as the positive set. Second, based on the assumption that extremely dissimilar diseases are much less likely to be treatable by the same drug, we generate an "unlikely" drug-disease pair set as follows: if a drug d is related to disease k_1 and in the disease set $\exists\ k_2$, such that the similarity between the two diseases $s(k_1, k_2) < t_k$, then we regarded the drug-disease pair (d, k_2) as "unlikely". Here, we set $t_k = 0.3$ (i.e. very un-similar disease [65]) and extract all corresponding pairs. Together with the random drug-disease pairs from the first step, this results in a negative set of 2,428,400 pairs. We than use these sets to compute ROC plots for varying n-cluster editing thresholds, displayed in Fig. 8.3, illustrating the performance of n-CluE for predicting the known set of drug-disease pairs (positive test set) and not too many of the (much bigger) negative control set. The AUC summarizes ROC plots into one value and remains at comparably high levels even when we remove up to one third of the input data indicating the robustness of n-CluE to missing input data.

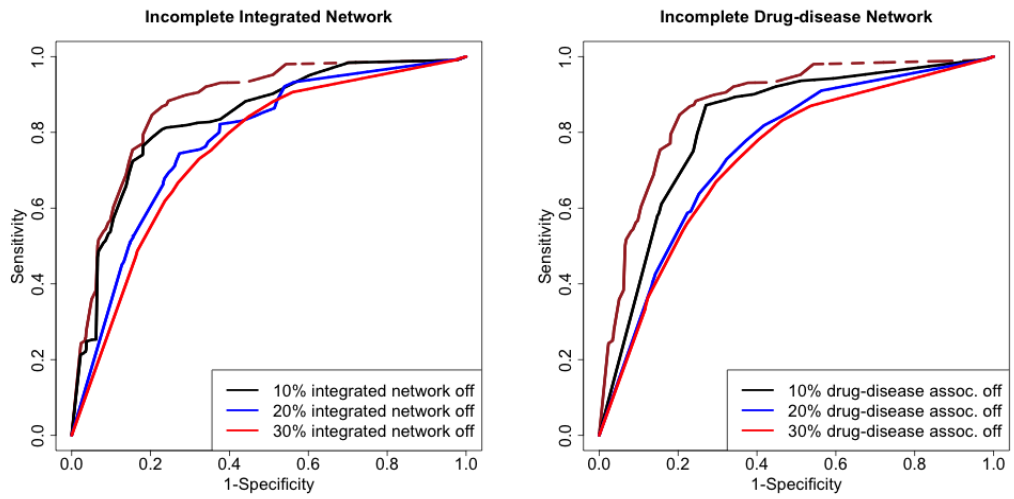

Figure 8.3: Robustness evaluation using receiver-operator characteristic (ROC) plots for varying n-cluster editing thresholds and with known drug-disease pairs as gold standard. We removed different amounts of edges from the "integrated network" (left) and only from the known set of drug-disease pairs (right), respectively. The brown dashed curve is the ROC achieved by n-CluE on the complete data set. Table 8.2 gives the AUCs of the ROCs.

Table 8.3: Robustness shown by the AUCs on the incomplete data sets.

Descriptions	**AUC**
10% of the integrated networks removed	0.836
20% of the integrated networks removed	0.778
30% of the integrated networks removed	0.752
10% of the drug-disease associations removed	0.824
20% of the drug-disease associations removed	0.752
30% of the drug-disease associations removed	0.734

8.4.2 Repositioning

The above pipeline yields 31,731 drug-disease pairs in the "novel prediction set" emerging from n-CluE. To further assess their relevance for drug repositioning and to filter for highly confident candidates we followed the assessment pipeline suggested by Rastegar-Mojarad *et al.* [152] and measure the co-occurrences of the drug and disease terms in MEDLINE, the U.S. National Library of Medicine bibliographic database

that contains more than 22 million references to journal articles in life sciences with a concentration on biomedicine. We implemented several assessment strategies providing different degrees of confidence. For the first strategy, we downloaded the titles and abstracts of all articles in MEDLINE (as of 11/2015) and built dictionaries for drug and disease synonyms that were retrieved from Drugbank and CTD, respectively. We used Apache Lucene (version 5.3.1) to index our literature set with drug and disease concepts from our dictionaries. We then retrieved evidence from the literature for the predicted novel drug targets based on the co-occurrence of these terms in a given title or abstract. To account for random drug-disease pair co-occurrences we only include pairs co-occurring in more than five articles as "high confidence" pairs. In addition, to overcome the limitation from co-occurrence based retrieval, i.e., the disregard of any contextual information that would help determine, for example, whether a disease is a potential indication or an adverse event for a given drug, we then developed a semantic literature search strategy based on the information in the Semantic Medline Database (SemMedDB) [99]. SemMedDB extracts pairs of biomedical concepts from MEDLINE titles and abstracts and characterize their relationships by assigning them one or several of 30 predicates, largely relating to clinical medicine (e.g. TREATS, DIAGNOSES, ADMINISTERED_TO, PROCESS_OF). We follow two strategies to retrieve drug-disease pairs from SemMedDB: retrieving drug and diseases that have (1) any and (2) TREATS relationships. Based on this strategy we hope to provide even more confidence for a given set of drug repositioning candidates, beyond term co-occurrence, which enables us to further prioritize our predictions. We call this the "treats annotation set".

Altogether, we sort the new drug reposition candidate pairs derived by n-CluE into three categories:

- (1) "novel prediction set": The 31,731 new drug-disease pairs predicted by n-CluE.

- (2) "high confidence set": A subset of 11,517 drug-disease pair identified by the MEDLINE-based search engine to co-occur more than five times in the literature pool.

- (3) "treats annotation set": SemMedDB extracted 1,382 case where the drug is mentioned to specifically TREAT the disease.

To further consolidate the significance of the "high confidence set" we permuted the "novel prediction set" of 31,731 novel drug-disease pairs randomly 1,000 times and repeated the literature mining pipeline that retrieved our 11,517 "high confidence set" drug disease pairs on the unpermuted data. Now, on average only 1,745 (+/- 39) pairs (5.5% of the 11,517 original pairs) have been identified as "high confidence set", suggesting a significant enrichment of our triangulated drug-disease associations ("novel prediction set") through (co-occurrence) literature support.

N-CluE triangulation suggests the drug molecule chlorpromazine (Drugbank ID: DB00477) to be associated with tuberculosis (Concept Unique Identifier: C0041327). We found this to be co-occurred in the literature several times (hence the "high confidence" level). Chlorpromazine has long been used for the therapy of psychotic disorders such as schizophrenia [37]. Our pipeline indicates an additional purpose for chlorpromazine, which is supported by several scientific articles. In the review of [216], it is suggested that the antibacterial properties of chlorpromazine could be used for anti-tubercular purpose. Another review discussed about the drug-resistance of pathogenic bacteria and suggested chlorpromazine to be promising as an effective anti-tubercular compound [4]. Likewise, n-CluE triangulation suggest the drug molecule dasatnib (Drugbank ID: DB01254) to be associated with thyroid cancer (Concept Unique Identifier: C0238463). We found this to co-occur in the literature serveral times (hence "high confidence" level). Dasatinib is an oral Src family kinase inhibitor approved by U.S. Food and Drug Administration (FDA) for the treatment of lymphoblastic leukemia and chronic myelogenous leukemia [25]. Experi-

ments demonstrated, both *in vitro* and *in vivo* the efficacy of dasatinib controlling the growth of thyroid cancer by inhibiting the Src family kinases, which is up-regulated in thyroid cancer cells [26]. An additional example for literature-supported repositioning is penicillamine (Drugbank ID: DB00859) and pulmonary hypertension (Concept Unique Identifier: C0020542), which has been suggested by Oroszlán *et al.* [141]. Furthermore, Xu *et al.* [206] reported that idiopathic pulmonary arterial hypertension is related to low levels of vasodilator nitric oxide (NO), and that molecules like S-nitroso-N-acetyl-D,L-penicillamine (SNAP), which provide NO in biochemical reactions, may serve as treatment. Additional (*in vitro*) experiments by Xu *et al.* give further evidence of potential effectiveness of SNAP as NO donor [206]. Eleven thousand additional such candidates are found for further laboratory validations and clinical studies. Research groups studying tuberculosis, for instance, will find 10 interesting repositioning records, including thalidomide, which has been suggested as an adjuvant treatment for tuberculosis [60]. Likewise, for pancreatic cancer investigators, we have identified 33 drug candidates with direct literature support, including salbutamol, ifosfamide, capecitabine and phenylephrine. Note that both prediction sets have not been filtered for potential side effects.

8.5 Related Methods

In Table 8.4 we summarize previously published methods integrating different data types for drug repositioning, most of which follow either a supervised or semi-supervised strategy to identify new drug targets or to predict new drug indications. No method exists integrating all relevant information (drug-disease associations, drug-gene associations, gene-disease associations as well as drug, gene and disease similarity networks) into one single model to predict new potential drug indications. With n-cluster editing we provide the first model for drug-gene-disease triangulation.

Table 8.4: Summary of methods for integrative analysis in drug repositioning.

Method name	Description	Integrated data	Learning strategy	Ref.
PREDICT	Drug respositioning based on integrated chemical similarity	Drug similarities (chemical)	Supervised	[68]
Network propagation	Infer drug-disease network from network propagation.	Drug similarity (chemical), Disease similarity (semantic)	Unsupervised	[89]
PreDr	Inference of novel drug-disease associations by matrix factorization	Drug and protein structures (chemical), side effects	Semi-supervised	[194]
DDR	Drug-disease prediction from known associations integrated with multiple drug similarities	Drug similarities (chemical), Target similarities (chemical)	Supervised	[217]

And with n-CluE, we have developed the first method to compute the model on data sets of real-world size. Finally, the literature mining post-processing identifies two kinds of high confidence targets for future clinical trials.

8.6 Conclusion

We developed the first tri-cluster editing approach, applied it to drug-disease-gene triangulation, integrated it with a literature mining pipeline, and applied it to several databases for computational drug repurposing yielding over thirty thousand new tricks for known drugs of which approximately eleven thousand significantly co-occur in literature over $1,300$ have a semantic "treats" annotation. The utilized n-CluE algorithm solves the longstanding weighted n-cluster graph editing computer science problem. A side-effect filter based on according databases, such SIDER [112], may further strengthen the confidence of our predictions. We anticipate that our methodology will be applied to other biomedical data processing problems in the future.

Table 8.5: Discovered drug repositioning examples (see text for discussion). Four of them have literature support while three are novel predictions.

Drugbank ID	**Drug name**	**Concept Unique Identifier**	**Disease name**	**Literature Support**
DB00118	S-adenosylmethionine (SAM)	C0010068	Coronary disease	Yes
DB00959	Methylprednisolone	C0003864	Arthritis	Yes
DB00571	Propranolol	C0677886	Ovarian epithelial cancer	Yes
DB01181	Ifosfamide	C2931037	Pancreatic cancer, adult	Yes
DB00762	Irinotecan	C2931037	Pancreatic cancer, adult	Novel
DB00635	Prednisone	C0030567	Parkinson's disease	Novel
DB04942	Tamibarotene	C1863051	Alzheimer disease type 2	Novel

In addition, we believe that our repositioning lists provide hot candidates for future screening efforts and will prove highly useful as a starting point for future clinical trials.

Chapter 9

Conclusion

To overcome the challenges of clustering n-dimensional ($n >= 2$) heterogeneous data, we have: (1) developed a software package n-CluE with exact and heuristic algorithms based on the bi-/n-cluster editing model and, (2) applied n-CluE on three real-world biomedical data sets to demonstrate its strength. We tackled the weighted bicluster editing problem by developing the first exact algorithm based on fixed-parameter tractability (FPT) with a kernelization procedure and a branching strategy to find the best set of edge modifications by traversing all possible solutions. The edge deletions heuristic improves the performance of the exact algorithm from exponential growth of running times to $O(|E|(|E| + |V|^2) + |V|^3)$ by searching for the most beneficial edge deletions to convert the input graph into disjoint bicliques. N-CluE's second heuristic, n-Force, is designed to perform n-clustering by finding an arrangement in u-dimensional ($u > 1$) space such that similar nodes are located closer to each other than the dissimilar ones. A systematic evaluation on artificially generated graphs was carried out to assess the performance of the three algorithms. Our results demonstrate the robustness of n-CluE dealing with graph of various sizes. The fixed-parameter algorithm, though with highest accuracy, is unfit to perform biclustering on graphs larger than 50 vertices due to explosion of running time. Our edge deletion heuristic

improves the running time and works fine for larger graphs, while n-Force shows the best performance in dealing with n-clustering on large scale data sets.

We then applied two of the three n-CluE's algorithms, the fixed-parameter algorithms and the edge deletion heuristic to two Genome-Wide Association Study (GWAS) to predict novel associations between genotypes and phenotypes. Two GWAS data sets were used in the biclustering analysis. Totally 86 new associations were identified, of which the associated phenotypes were found to be related by clinical studies.

To evaluate the performance of n-CluE on gene expression data, n-Force was systematically compared with other eight existing biclustering tools, both on artificial and real-world data sets. The results that n-Force outperformed other tools in both scenarios have proven the strength of the underlying bicluster editing model for gene expression data mining.

In the last application case, we suggest the utilization of drug-gene-disease triangulation to discover drug repositioning. We first summarize the latest progress of computational drug repositioning strategies from different aspects and afterwards present our model based on n-cluster editing to integrate drug-gene-disease networks. Combined with a semantic literature mining approach, n-CluE managed to predict 31,731 ("novel prediction set") new drug-disease associations of which 11,517 ("high confidence set") co-occur in literature and 1,382 cases were marked as "treats annotation set" where the drug is identified to treat the disease. The robustness of n-CluE in the context of drug repositioning was also assessed. We anticipate that our results and n-CluE to be valuable in identifying alternative indications for existing drugs and cost reduction.

Appendix A

Appendix

A.1 Appendix A: Additional Proofs

A.1.1 Proof of Theorem 4.1:

Proof. "⇒" Let $\mathcal{I} \subseteq \mathcal{C}$ be an exact cover of A, define: $F_1 = \{\{u_i, v_{S_j}\} |\ u_i \in U_1 \wedge v_{S_j} \in V_S^m \ \wedge\ \{u_i, v_{S_j}\} \in E \ \wedge\ S \notin \mathcal{I}\}$, $F_2 = \{\{v_i, u_{S_j}\} |\ v_i \in V_1 \ \wedge\ u_{S_j} \in U_S^m \ \wedge\ \{u_i, v_{S_j}\} \in E \ \wedge\ S \notin \mathcal{I}\}$, $F_3 = E_2 \backslash \{\{u_i, v_j\} |\ \ S = \{a_x, a_y, a_z\} \in \mathcal{I} \ \wedge\ i, j \in \{x, y, z\} \ \wedge\ i \neq j\}$.

Let F be the union of F_1, F_2 and F_3. Since in $\mathcal{I}$, each element must be contained in exactly one triplet, then in the graph $G - F$, for all vertices $u \in U_1 \cup V_1$, u is connected with exactly one W_S^m. Hence $G - F$ is a bicluster graph containing $2|\mathcal{C}|$ bicliques and clearly gives a solution to the s-BIPLEX EDITING instance.

Since $|\mathcal{I}| = n$, then $|\mathcal{C}| - n$ is equal to the number of "surplus triplets", i.e., the triplets that are not in $\mathcal{I}$. Thus, we have $|F_1| = |F_2| = 3(|\mathcal{C}| - n) \cdot m = M/2$. Moreover, in $G - F$, we have $3n + 6|\mathcal{I}| = 9n$ edges between U_1 and V_1, thus we can easily compute that $|F_3| = |E_1| + |E_2| - 9n = |E_2| - 6n$. Denote F as the set of edited edges, i.e., the set of deleted edges and inserted missing edges. We have $|F| = |F_1| + |F_2| + |F_3| = M + N$.

"⇐" In the reverse direction, if there exists an optimal solution for the s-BIPLEX EDITING instance, let F be the set of edited edges. We prove that $|F| \geq M + N$.

Let $F' = F_1 \cup F_2 \cup F_3$ where F_1, F_2 and F_3 are defined in the "⇒" direction, and thus $|F'| = M + N$. Clearly F' gives a solution to the s-BIPLEX EDITING instance. Before proving the lower-bound on $|F|$, we first show $|F'| < m(\frac{m+1}{6})$.

In 3X3C, each element can be covered by at most 3 triplets, thus $|\mathcal{C}| \leq 3n$. Hence $M = 2m(3|\mathcal{C}| - 3n) \leq 2m(9n - 3n) \leq 12mn$. Consider $|E_2|$, $|E_2| = 9|\mathcal{C}| - 3|\mathcal{C}| = 6|\mathcal{C}| \leq 18n$. Thus $|F'| = |F_1| + |F_2| + |F_3| = M + N = 2m(3|\mathcal{C}| - 3n) + |E_2| - 6n \leq 12mn + 18n - 6n = 12mn + 12n = 12m \cdot \frac{m}{72+s} + 12 \cdot \frac{m}{72+s} \leq \frac{m^2}{6} + \frac{m}{6} = m(\frac{m+1}{6})$

Let G' be the resulting graph of G after all edge insertions and edge deletions in F are performed. We next prove the claim: For every W_S and W_S^m, there is s-biplex B in G', such that $W_S^m \subseteq B \subseteq (W_S^m \cup W_S)$.

We proceed with the proof in two steps: In the first step, we show that for each W_S^m, there is always an s-biplex B in G', such that $|B \cap W_S^m| \geq \frac{3}{2}m + 3$.

By contradiction, if there is no such s-biplex in G'. Let $\mathcal{B} = \{B_1, B_2, ..., B_l\}$ be the set of s-biplexes in the optimal solution G', such that $|B_i \cap W_S^m| \neq \emptyset$ for $1 \leq i \leq l$. According to the assumption, we have $|B_i \cap W_S^m| \leq \frac{3}{2}m + 2$ for all $1 \leq i \leq l$. Let $x_i = |B_i \cap U_S^m|$, $y_i = |B_i \cap V_S^m|$, $x_i + y_i \leq \frac{3}{2}m + 2$. Obviously, we have to delete all the edges between $W_S^m \cap B_i$ and $W_S^m - B_i$. Hence the number of edge deletions C_D with respect to W_S^m is at least:

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

The inequality (*) holds since for all integers $x_i > 0$ and $y_i > 0$, we have $x_i y_i \leq \frac{1}{4}(x_i+y_i)^2$. The inequality (**) is correct because we have $(x_i+y_i) \leq \frac{3}{2}m+2$. The equality (***) is correct since we have $\sum_{i=1}^{l}(x_i+y_i) = |W_S^m| = |U_S^m| + |V_S^m| = 2m$. Thus we know that $|F| \geq \frac{1}{4}m^2 - m$. Consider $|F| - |F'|$, we have: $|F| - |F'| \geq \frac{1}{4}m^2 - m - m(\frac{m+1}{6}) = \frac{1}{4}m^2 - \frac{1}{6}m^2 - \frac{7}{6}m = \frac{m}{6}(\frac{m}{2}-7)$.

In our construction, $m = (72+s)n$, thus $\frac{m}{6}(\frac{m}{2}-7) > 0$. This contradicts with the assumption that F is optimal. Thereby we have proved that there exists an s-biplex B in G' with $|B \cap W_S^m| \geq \frac{3}{2}m+3$. For each W_S^m, denote B_S as the s-biplex in the optimal solution such that $|B_S \cap W_S^m| \geq \frac{3}{2}m+3$.

In the second step, we prove the claim that for each W_S^m, we have $W_S^m \subseteq B_S \subseteq (W_S^m \cup W_S)$. By contradiction, we assume there is one vertex $x \in W_S^m$, such that $x \notin B_S$. Without loss of generality, we assume $x \in U_S^m$. Since $0 \leq |B_S \cap U_S^m| \leq m$, the intersection between B_S and V_S^m must be at least $\frac{3}{2}m+3-m = \frac{m}{2}+3$, i.e.,

$|B_S \cap V_S^m| \geq \frac{1}{2}m + 3$. Thus to include x into B_S, we need at most: (1) Remove the edges between $V_s^m - B_S$ and x, and (2) remove the edges between V_S and x. This gives a cost $C_I \leq |V_S^m - B_S| + 3 \leq m - (\frac{m}{2} + 3) + 3 = \frac{m}{2}$. On the other hand, to have x not in B_S, it requires at least to delete all edges between x and $V_S^m \cap B$, which gives a cost $C_D = |V_S^m \cap B_S| = |W_S^m \cap B_S| - |U_S^m \cap B_S| \geq \frac{3}{2}m + 3 - m = \frac{m}{2} + 3 \geq C_I$. Hence we know that it is always better to include x as part of B_S than to leave x out.

The remaining part of the claim that $B_S \subseteq (W_S^m \cup W_S)$ is proved similarly by contradiction. Suppose there exists a vertex $y \in B_S$ with $y \notin (W_S^m \cup W_S)$. Without loss of generality, we assume $y \in U$. Denote C_I as the cost of including y as a part of B_S. Then we have to insert between y and V_S^m at least $(|V_S^m| - s)$ edges. $C_I \geq |V_S^m| - s = m - s$. Consider the cost C_D of removing y from B_S. C_D is at most the number of edges between y and V_S. Since $|V_S| = 3$, we have $C_D \leq 3 < C_I$, contradicting with the claim of optimal solution. Hence no vertex outside $(W_S \cup W_S^m)$ would end up in B_S in any optimal solution.

W know that for each W_S^m, there is an s-biplex B_S in the optimal solution, such that $W_S^m \subseteq B_S \subseteq (W_S^m \cup W_S)$. It remains to find out where the vertices in W_S end up. Examine an element $u_i \in U_1$, such that its corresponding element in A, a_i, is a member of (at least) two subsets $S_1, S_2 \in \mathcal{C}$. In the proof above, we have already shown that $W_{S_1}^m$ and $W_{S_2}^m$ are contained in distinct s-biplexes in any optimal solution. Hence we have to delete the edges either between u_i and $V_{S_1}^m$, between u_i and $V_{S_2}^m$ or both. Obviously, if each u_i only connects to one V_S^m, the total number of edge deletions is at least $3m(|\mathcal{C}| - n)$. For all such vertices in V_1, the same argument applies. Hence we have $|F \cap (E_3 \cup E_4 \cup E_5)| \geq M$.

Since for each s-biplex in G', we have $B_S \subseteq (W_S^m \cup W_S)$, we know that in the optimal solution, every $u_i \in U_1$ is a neighbor of at most three vertices in V_1. Since $|U_1| = |V_1| = 3n$, there are at most $9n$ edges between U_1 and V_1 in the optimal

solution. Thereby the edge deletions between U_1 and V_1 is at least $|F \cap (E_1 \cup E_2)| \geq |E_1| + |E_2| - 9n = |E_2| - 6n = N$.

In conclusion, we have that $|F \cap (E_3 \cup E_4 \cup E_5)| \geq M$ and $|F \cap (E_1 \cup E_2)| \geq N$. Hence $|F| \geq M + N$. Therefore, we know that $F' = F_1 \cup F_2 \cup F_3$ gives an optimal solution to s-BIPLEX EDITING. Finally, let $\mathcal{B}^*$ be a subset of all s-biplexes in the optimal solution, such that $\forall B \in \mathcal{B}^*$, we have $B = W_S^m \cup W_S$. Then $\{S \in \mathcal{C} | B_S \in \mathcal{B}^*\}$ gives a cover to the 3X3C instance. □ □

A.1.2 Proof of Lemma 4.2:

Proof. By contradiction, suppose we have a vertex $u^* \in R$ with $d(u^*) \leq |T| - s - 2$. Then we can find a non-cut vertex $v \in H$. Such vertex must exist, since it is well-known that in an arbitrary graph G, there are at least 2 non-cut vertices. Considering $H - v = (R', T', E'')$, we have 2 cases:

Case i: $v \in N(u^*)$. Then in $H - v$, $d_{H-v}(u^*) = d(u^*) - 1 \leq |T| - s - 2 - 1 = |T'| - s - 2$, u^* is still a forbidden vertex.

Case ii: $v \notin N(u^*)$. Then in $H - v$, $d_{H-v}(u^*) = d(u^*) \leq |T| - s - 2 = |T'| - s - i$, u^* is still a forbidden vertex.

In summary, either case gives another forbidden subgraph which is a subgraph of H, contradicting with H being minimal. □ □

A.1.3 Proof of Lemma 4.4:

Proof. For the first and second claims, the proofs are simple: If a vertex $u \in R_1$ is a non-cut vertex, we can remove u from H without affecting all vertices in R_0 being forbidden vertices, thus $H - u$ is still forbidden, contradicting with "minimal". If a vertex $v \in T_0$ is a non-cut vertex, we can remove v from H. In $H - v$, $min\{d_{H-v}(u)\} \leq |T'| - s - 1$, thus $H - v$ is still forbidden. For Claim 3, if there is more than one

forbidden vertex in R_0, all vertices in R_0 must be cut vertex, otherwise we could remove an arbitrary vertex in R_0 and the resulting graph is still forbidden.

Next we prove Claim 4. By contradiction, suppose $X_1 \cap R_1 = \emptyset$. We have $X_1 \subseteq R_0$. Then we have two cases:

Case i. $Y_1 = \emptyset$. Then we know that for $u_0 \in X_1$, in the original subgraph H, $d(u_0) = 1$. Since u_0 is in R_0, we know by Lemma 1 that in the original subgraph H, $\forall u \in R_0$, we have $d(u) = 1$. Therefore, $\forall u \in R_0$, u is a non-cut vertex. Hence we can only have $|R_0| = 1$, since otherwise R_0 would contain non-cut vertex and thus contradicts Claim 3. However, since $R_0 = \{u\}$ and $d(u) = 1$, we have $|T_0| = 1$, contradicting with $|T_0| > 1$ in Claim 4.

Case ii. $Y_1 \neq \emptyset$. Since $X_1 \cap R_0 \neq \emptyset$, we have $Y_1 \cap T_0 \neq \emptyset$. In subgraph H_1, there must be at least two non-cut vertices. Assume a vertex $v \in Y_1$ is a non-cut vertex. then in H, v must be a non-cut vertex as well, contradicting Claim 2. Thus we know that there must be at least two vertices $u_1, u_2 \in X_1$, such that u_1 and u_2 are non-cut vertices of H_1. Consider u_1 and u_2, we have three following properties:(1) In H, u_1 and u_2 must be connected to v^*. This is obvious since otherwise u_1 and u_2 would be non-cut vertices in H as well, contradicting with Claim 3. (2)$\forall x, y \in N(u_1) \backslash \{v^*\}$, there exists a path P in H_1 from x to y without passing u_1. This is true since otherwise u_1 would be a cut vertex in H_1. (3)$\forall x \in N(u_1) \backslash \{v^*\}$, there exists a path Q in H_1 from x to u_2 without passing u_1. This is also true since otherwise u_1 would be a cut vertex in H_1.

Consider two arbitrary neighbours of u_1, $x, y \in N(u_1)$. If $x \neq v^*$ and $y \neq v^*$, then we already know that there exists a path P from x to y without passing u_1. If $x = v^*$, we can also find a path from y to v^* without passing u_1. This is true, because we already know that there exists a path Q from y to u_2 and u_2 is connected to v^*, $Q' = Q \cup \{v^*\}$ is the path. In summary, $\forall x, y \in N(u_1)$ in H, there exists a path

from x to y without passing u_1. Thus u_1 is a non-cut vertex in H. Moreover, since $u_1 \in R_0$, this contradicting with Claim 3.

Finally, we prove Claim 5. For the purpose of contradiciton, suppose $\forall v \in T$, $d(v) \geq 2$. Consider an arbitrary vertex $u_1 \in R_1$. Since u_1 is a cut vertex, we can find a connected component $C_1 = (U_{C_1}, V_{C_1}, E_{C_1})$ in $H - u_1$, such that all vertices in $V_{C_1} \cap R$ are cut vertices. Such C_1 must exist when $|R_0| = 1$. In the case of $|R_1| > 1$, we have already proved that all vertices in R are cut vertices in H. By the assumption, we have $V_{C_1} \cap R \neq \emptyset$; let $u_2 \in V_{C_1} \cap R$. The fact that u_2 is a cut vertex in H leads to that there exists a connected component $C_2 = (U_{C_2}, V_{C_2}, E_{C_2})$ in $H - u_2$ which is also a connected component in $H - \{u_1, u_2\}$. Again, by the assumption, $V_{C_2} \cap R \neq \emptyset$; let $u_3 \in V_{C_2} \cap R$. The cut vertex u_3 implies that there is a connected component $C_3 = (U_{C_3}, V_{C_3}, E_{C_3})$ in $H - u_3$, which is a connected component in $H - \{u_1, u_2, u_3\}$. Since u_1 and u_2 are connected in $H - u_3$. Then $V_{C_3} \cap R \neq \emptyset$. Since u_1 and u_2 are connected in $H - u_3$, then $V_{C_3} \cap R \neq \emptyset$ follows our assumption. Note that $V_{C_3} \subset V_{C_2} \subset V_{C_1}$. Since H is finite, we will end up with a connected component $C_i = (U_{C_i}, V_{C_i}, E_{C_i}$ where $V_{C_i} \cap R = \emptyset$. However, this would imply that the vertices in $V_{C_i} \cap T$ are degree -1 vertices, contradicting to Claim 2 and our assumption. □ □

A.1.4 Proof of Theorem 4.8:

Proof. Obviously, the problem is in NP. Then the NP-hardness of the problem is shown by a reduction from MAXIMUM BALANCED BICLIQUE (MBB):

> **Input**: An undirected bipartite graph $G = (U, V, E)$, and and integer $k \geq 0$
>
> **Question**: Does there exist an induced biclique $C^* = (U_{C^*}, V_{C^*}, E_{C^*})$ in G, such that $|U_{C^*}| = |V_{C^*}| = k$?

Given an MBB instance $G = (U, V, E)$ and a nonnegative integer k, we construct an EQUAL-SIZE BICLUSTER EDITING (ESBE) instance as follows: (1) Add a set

of components $\mathcal{A}$ to G, such that

$\mathcal{A} = \{A_1, A_2, ..., A_{|U|-k}\}$, $A_i = (U_{A_i}, V_{A_i}, E_{A_i})$, $1 \leq i \leq |U| - k$. For each A_i, we have $|U_{A_i}| = k-1$, $|V_{A_i}| = k$. We first connect all vertices in U_{A_i} to all vertices in V_{A_i} for $1 \leq i \leq |U| - k$, and then connect all the vertices in U to all the vertices in $\bigcup_{A_i \in \mathcal{A}} V_{A_i}$.

(2) Add a set of components $\mathcal{B}$ to G, such that

$\mathcal{B} = \{B_1, B_2, ..., B_{|V|-k}\}$, $B_j = (U_{B_j}, V_{B_j}, E_{B_j})$, $1 \leq j \leq |V| - k$. For each B_j, we have $|U_{B_j}| = k$, $|V_{B_j}| = k-1$. We first connect all vertices in U_{B_j} to all vertices in V_{B_j} for $1 \leq j \leq |V| - k$, and then connect all the vertices in V to all the vertices in $\bigcup_{B_j \in \mathcal{B}} U_{B_j}$.

Hence, we have the new graph for ESBE $G' = (U', V', E')$: $U' = \bigcup_{A_i \in \mathcal{A}} U_{A_i} \cup \bigcup_{B_i \in \mathcal{B}} U_{B_i} \cup U$, $V' = \bigcup_{A_i \in \mathcal{A}} V_{A_i} \cup \bigcup_{B_i \in \mathcal{B}} V_{B_i} \cup V$. The edges of the new graph is: $E' = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$, with $E_1 = E$, $E_2 = \{\{u, v\} \mid u \in U_{A_i} \ \wedge \ v \in V_{A_i}\}$, $E_3 = \{\{u, v\} \mid u \in U_{B_i} \ \wedge \ v \in V_{B_i}\}$, $E_4 = \{\{u, v\} \mid u \in U \ \wedge \ v \in V_{A_i} \ \wedge \ 1 \leq i \leq |U| - k\}$ and $E_5 = \{\{u, v\} \mid u \in V \ \wedge \ v \in U_{B_i} \ \wedge \ 1 \leq i \leq |V| - k\}$.

The sizes of U' and V' are: $|U'| = k(|U| - k) + k(|V| - k) + k$ and $|V'| = k(|V| - k) + k(|U| - k) + k$. The number of edges is: $|E'| = |E| + k(|U| - k)(|U| + k - 1) + k(|V| - k)(|V| + k - 1)$

Let k' and d be the parameters of the new instance: $k' = k(|U| - k)(|U| - 1) + k(|V| - k)(|V| - 1) + |E| - k^2$ and $d = |U| + |V| - 2k + 1$.

"$\Rightarrow$ " Suppose we have found a biclique $C^* = (U_{C^*}, V_{C^*}, E_{C^*})$ in G with $|U_{C^*}| = |V_{C^*}| = k$. Then by the following steps, we can have a solution for the ESBE problem:

(1) For every vertex $u_i \in U \backslash U_{C^*} = \{u_1, u_2, ...u_{|U|-k}\}$, we delete the edges between u_i and all vertices in A_j, $i \neq j$. For every vertex $u'_i \in U_{C^*} = \{u_{|U|-k+1}, u_{|U|-k+2}, ..., u_{|U|}\}$, we delete all the edges between u'_i and all vertices in $\bigcup_{A_i \in \mathcal{A}} V_{A_i}$.

(2) For every vertex $v_i \in V \backslash V_{C^*} = \{v_1, v_2, ...v_{|V|-k}\}$, we delete the edges between v_i and all vertices in B_j, $i \neq j$. For every vertex $v'_i \in V_{C^*} = \{v_{|V|-k+1}, v_{|V|-k+2}, ..., v_{|V|}\}$,

we delete all the edges between v'_i and all vertices in $\bigcup_{B_j \in \mathcal{B}} U_{B_j}$.

(3) Delete the edges between C^* and $G - C^*$.

Let $k'_1 = k(|U| - k)(|U| - 1)$, $k'_2 = k(|V| - k)(|V| - 1)$ and $k'_3 = |E| - k^2$. Clearly, $k'_1 + k'_2 + k'_3 = k'$. More specifically, step (1) requires $k(|U| - k)(|U| - k - 1) + k^2(|U| - k) = k(|U| - k)(|U| - 1) = k'_1$ edge deletions. Step (2) requires $k(|V| - k)(|V| - k - 1) + k^2(|V| - k) = k(|V| - k)(|V| - 1) = k'_2$ edge deletions. Step (3) requires $|E| - k^2 = k'_3$ edges deletions. Thus altogether the 3 steps above need $k'_1 + k'_2 + k'_3 = k'$ edge deletions. Obviously the resulting graph contains $|U| + |V| - 2k + 1$ bicliques. This gives a solution to ESBE instance.

"$\Leftarrow$" Given the ESBE instance, we know $|U'| = |V'| = k(|U| + |V| - 2k + 1)$. First, we prove that each A_i (or B_i) will end up in a separated biclique in the optimal solution. That is, let $\mathcal{D} = \mathcal{A} \cup \mathcal{B}$, for each $D_i \in \mathcal{D}$, in the optimal solution, there exists a biclique C_i such that $D_i \subseteq C_i$ and $D_j \cap C_i = \emptyset$ for all $1 \leq i \neq j \leq |U| + |V| - 2k$.

Denote $\mathcal{C} = \{C_1, C_2, ..., C_d\}$ as the d bicliques in the optimal solution and let $C_i = (U_i, V_i, E_i)$. Clearly, the bicliques in $\mathcal{C}$ are d equally-sized balanced bicliques, thus we have $|U_i| = |V_i| = |U'|/d = |U'|/(|U| + |V| + 2k + 1) = k$. Thus, the total number of edges in the optimal solution is dk^2. Compute the difference between $|E'|$ and the number of edges in the optimal solution, we have:

$$
\begin{aligned}
|E'| - dk^2 &= |E| + k(|U| - k)(|U| + k - 1) + \\
&\quad k(|V| - k)(|V| + k - 1) - dk^2 \\
&= k(|U| - k)(|U| + k - 1 - k) + \\
&\quad k(|V| - k)(|V| + k - 1 - k) + \\
&\quad |E| - k^2 \\
&= k'
\end{aligned}
$$

Since $|E'| - dk^2 = k'$, this indicates that only edge deletions are allowed, otherwise we would have to convert G' into d equally-sized bicliques with more than k' edge modifications. To show that each D_i ends up in a separated biclique, we first show that for every D_i, there is a biclique C_i in the optimal solution, such that $D_i \subseteq C_i$. The proof is as follows:

For each A_i, we can always find a biclique C_i such that $U_{A_i} \cap U_i \neq \emptyset$ in the optimal solution. Let $u_0 \in (U_{A_i} \cap U_i)$. First, we show that $\forall v \in V_{A_i}$, we have $v \in C_i$. By contradiction, if there exists a vertex $v_0 \in V_{A_i}$ and $v_0 \notin C_i$, since $|V_i| = k$, there must exists a $v_0' \in V_i$ such that $v_0' \notin V_{A_i}$. Then we have to insert at least an edge between u_0 and v_0' and thus contradicts with no edge insertion proved above. Second, we prove that $\forall u_0' \in U_{A_i}$, $u_0' \neq u_0$, we have $u_0' \in C_i$. By contradiction, if $u_0' \in C_j$, $i \neq j$, then clearly, we have to insert the edges between u_0' and V_j thus contradicts with no edge insertion. The same proof applies on B_i as well. Therefore, we have proved that, for every D_i, there exists a C_i such that $D_i \subseteq C_i$.

Next, we show that $C_i \cap D_j = \emptyset$ for all $i \neq j$. By contradiction, if there exists an integer j, such that $C_i \cap D_j \neq \emptyset$, $i \neq j$, then we have to insert edges in C_i between

D_i and $D_j \cap C_i$. This contradicts with no edge insertion. Thereby, we have proved that every D_i ends up in a separated biclique, $1 \leq i \leq |U| + |V| - 2k$.

Finally, we prove an optimal solution to ESBE will give a solution to MBB. Without loss of generality, we assume $A_1 \subseteq C_1$. Since $|U_1| = k$ and $|U_{A_1}| = k - 1$, clearly there is one vertex $u_1 \in C_1$ but $u_1 \notin A_1$. We next discuss where this u_1 comes from. As proved above, u_1 cannot be from A_i, $i > 1$ or any B_j, $1 \leq j \leq |V| - k$. Thus we have $u_1 \in U$. Therefore, for every A_i, $1 \leq i \leq |U| - k$, there exists one vertex u_i in U, such that u_i ends up in the same biclique with A_i. Similarly, we can show that for every B_i, $1 \leq i \leq |V| - k$, there exists one vertex v_i in V, such that v_i ends up in the same biclique with B_i. That leaves us $|U| - (|U| - k) = k$ vertices in U and $|V| - (|V| - k) = k$ in V. If these $2k$ vertices cannot form a balanced biclique in G', then we have to insert edges between them. This would lead to more than k' edge modifications and thus contradicts with the assumption. Therefore, if we have a solution to ESBE instance, we must be able to find a biclique with k vertices in each vertex set in G, which gives a solution to the MBB instance. □ □

A.1.5 Proof of Theorem 4.9:

Proof. We reduce ESBE to AVERAGE-s-BIPLEX EDITING. First, given an ESBE instance $G = (U, V, E)$, with k as the parameter of maximum number of edge modifications and d as the number of bicliques in the solution, we can safely assume that $|U| = |V|$, an AVERAGE-s-BIPLEX EDITING instance $G' = (U', V', E')$ is constructed by adding d components to each of U and V, respectively:

$U' = U \cup U_1 \cup U_2 \cup ... \cup U_d$,

$V' = V \cup V_1 \cup V_2 \cup ... \cup V_d$.

Let $l = |U|/d$. Each of U_i and V_i contains $d^4 l^4 s^4$ vertices, i.e.:

$|U_i| = |V_i| = d^4 l^4 s^4$.

We denote $W_i = U_i \cup V_i$. For each U_i, we divide the vertices in U_i into $d^4l^4s^3$ vertex groups: $U_i = U_i^1 \cup U_i^2 \cup ... \cup U_i^{d^4l^4s^3}$. $|U_i^j| = |U_i^{j'}|$ and $U_i^j \cap U_i^{j'} = \emptyset$ for all $j \neq j'$. Hence we have $|U_i^j| = s$ for all $1 \leq j \leq d^4l^4s^3$. For each V_i, we divide them in the same way.

$U_i = \bigcup_{j=1}^{d^4l^4s^3} U_i^j$ and $V_i = \bigcup_{j=1}^{d^4l^4s^3} V_i^j$.

Then we connect: (1) all vertices in U_i to all vertices in V, and (2) all vertices in U_i to all vertices in U. Next, inside each W_i, we connect all vertices in U_i^j to all vertices in $\bigcup_{j \neq j'} V_i^{j'}$, leaving the vertex pairs $\{u, v\}$ with $u \in U_i^j, v \in V_i^j$) unconnected. Finally, within each W_i, we further remove $s \cdot l$ arbitrary edges, which leaves $|U_i| \cdot |V_i| - s \cdot |U_i| - sl = d^8l^8s^8 - s(d^4l^4s^4 + l)$ edges in each W_i. The new parameter $k' = k + 2(d-1)d^5l^5s^4$. Obviously, G' is not an average-s-biplex cluster graph and edge modifications are required. Note here we can assume that $1 \leq s < d \cdot l$.

"$\Rightarrow$" If we have a solution for ESBE, i.e., a group of d bicliques with equal size l which costs at most k modifications. Let $\mathcal{B} = \{B_1, B_2, ..., B_d\}$ be the set of d bicliques in the solution. Then a solution for AVERAGE-s-BIPLEX EDITING is constructed by disconnecting B_i with all constructed components W_j, $1 \leq i \neq j \leq d$. Thus each B_i is only connected to W_i, $1 \leq i \leq d$. This asks for $2(d-1)d^5l^5s^4$ edge deletions. Denote $C_i = B_i \cup W_i$. Since $|C_i \cap U'| = |C_i \cap V'|$, then it is sufficient just to consider the average degree of the vertices in $|C_i \cap U'|$. The average degree in C_i is:

$$\overline{d_{C_i}} = \frac{(|E(U_i, V_i)| + |E(U_i, B_i)| + |E(V_i, B_i)| + |E(B_i)|)}{(|U_i| + |U' \cap B_i|)}$$

Here, $E(U_i, V_i)$ refers to the edges between U_i and V_i; $E(U_i, B_i)$ and $E(V_i, B_i)$ refer to the edges between B_i and U_i, B_i and V_i, respectively; $E(B_i)$ is the set of edges within the biclique B_i. According to the construction of G', $|E(U_i, V_i)| =$

$d^8l^8s^8 - s(d^4l^4s^4 + l)$, $|E(U_i, B_i)| = |E(V_i, B_i)| = s^4d^5l^5$. Since B_i is biclique, we have $|E(B_i)| = l^2$. Hence,:

$$\begin{aligned}
\overline{d_{C_i}} &= \frac{(|E(U_i, V_i)| + |E(U_i, B_i)| + |E(V_i, B_i)| + |E(B_i)|)}{(|U_i| + |U' \cap B_i|)} \\
&= \frac{d^8l^8s^8 - s(d^4l^4s^4 + l) + 2d^4l^5s^4 + l^2}{l + d^4l^4s^4} \\
&= \frac{(d^4l^4s^4 + l)^2 - s(d^4l^4s^4 + l)}{d^4l^4s^4 + l} \\
&= d^4l^4s^4 - s \\
&= |V_i| + |V' \cap B_i| - s
\end{aligned}$$

This average degree satisfies the average-s-biplex. This will give a solution to AVERAGE-s-BIPLEX EDITING instance.

"$\Leftarrow$". We firstly prove that for each W_i there is an average-s-biplex D_i in the optimal solution, such that $W_i \subseteq D_i$, $1 \leq i \leq d$ and $W_j \cap D_i = \emptyset$ for $1 \leq j \leq d$, $i \neq j$. In order to prove this claim, we proceed in several steps. First, we prove that, for each W_i, there exists an average-s-biplex D_i and $|D_i \cap W_i| \geq d^4l^4s^4$.

By contradiction, if there does not exist such an average-s-biplex in the optimal solution. Denote $\mathcal{D} = \{D_1, D_2, ..., D_r\}$ as the set of average-s-biplexes in the optimal solution, $r \geq 1$. Then we have $\forall D_j \in \mathcal{D}$, $|W_i \cap D_i| < d^4l^4s^4$, $1 \leq j \leq r$, $1 \leq i \leq d$. Specifically, consider W_1, without loss of generality, let $\mathcal{D}' = \{D_1, D_2, ..., D_t\}$, $1 \leq t \leq r$ be the set of average-s-biplexes in the optimal solution that intersect with W_1. Let $X_i = U_1 \cap D_i$, and $Y_i = V_1 \cap D_i$, $1 \leq i \leq t$. Thus we have to delete the edges between

different average-s-biplexes. The corresponding number of edge deletions C_D is:

$$
\begin{aligned}
C_D &\geq \frac{1}{2}\sum_{i=1}^{t}(|X_i|(|V_1|-|Y_i|)+|Y_i|(|U_1|-|X_i|))-s(d^4l^4s^4+l)\\
&=\frac{1}{2}\sum_{i=1}^{t}(|X_i|(d^4l^4s^4-|Y_i|)+|Y_i|(d^4l^4s^4-|X_i|))\\
&-s(d^4l^4s^4+l)\\
&=\frac{1}{2}\sum_{i=1}^{t}(d^4l^4s^4(|X_i|+|Y_i|)-2|X_i||Y_i|)-s(d^4l^4s^4+l)\\
&\geq\frac{1}{2}\sum_{i=1}^{t}(d^4l^4s^4(|X_i|+|Y_i|)-\frac{1}{2}(|X_i|+|Y_i|)^2)\\
&-s(d^4l^4s^4+l)\quad(*)\\
&\geq\frac{1}{2}\sum_{i=1}^{t}(d^4l^4s^4(|X_i|+|Y_i|-\frac{1}{2}|X_i|-\frac{1}{2}|Y_i|))\\
&-s(d^4l^4s^4+l)\quad(**)\\
&=d^4l^4s^4\cdot\frac{1}{2}\sum_{i=1}^{t}(\frac{1}{2}(|X_i|+|Y_i|))-s(d^4l^4s^4+l)\\
&=\frac{1}{2}d^8l^8s^8-s(d^4l^4s^4+l)\quad(***)
\end{aligned}
$$

Inequality (*) holds because for an arbitrary pair of integers (a,b) that are greater than 0, we have $ab\leq\frac{1}{4}(a+b)^2$. Inequality (**) holds because we assume $|X_i|+|Y_i|<d^4l^4s^4$ for all i, thus $d^4l^4s^4(|X_i|+|Y_i|)-\frac{1}{2}(|X_i|+|Y_i|)^2<d^4l^4s^4(|X_i|+|Y_i|)-\frac{1}{2}\cdot d^4l^4s^4(|X_i|+|Y_i|)$. Equality (***) holds since we have $\sum_{j=1}^{t}(|X_i|+|Y_i|)=|U_1|+|V_1|=2d^4l^4s^4$. Obviously, when $d\geq 2$, $l\geq 1$, $s\geq 1$, we have $C_D>k'$, contradicting with the assumption that D is a solution.

Denote D_1 as the average-s-biplex such that $|D_1\cap W_1|\geq d^4l^4s^4$. Let $X_1=D_1\cap U_1$ and $Y_1=D_1\cap V_1$. Next, we prove that $|X_1|\geq\frac{1}{3}d^4l^4s^4$ and $|Y_1|\geq\frac{1}{3}d^4l^4s^4$.

By contradiction, assume $|X_1| < \frac{1}{3}d^4l^4s^4$. Then we have to delete all the edges between $W_1 \cap D_1$ and $W_1 - D_1$. The cost for edge deletions C_D is lower-bounded by:

$$\begin{aligned} C_D &\geq |X_1|(|V_1| - |Y_1|) + |Y_1|(|U_1| - |X_1|) - s(d^4l^4s^4 + l) \\ &\geq d^4l^4s^4(|X_1| + |Y_1|) - \frac{1}{2}(|X_1| + |Y_1|)^2 - s(d^4l^4s^4 + 1) \\ &\geq d^4l^4s^4 \cdot \frac{4}{3}d^4l^4s^4 - \frac{1}{2} \cdot \frac{16}{9}d^8l^8s^8 - s(d^4l^4s^4 + 1) \quad (*) \\ &= \frac{4}{9}d^8l^8s^8 - s(d^4l^4s^4 + 1) \geq k' \end{aligned}$$

Inequality (*) holds, since $d^4l^4s^4(|X_1| + |Y_1|) - \frac{1}{2}(|X_1| + |Y_1|)^2$ reaches the maximum value within the range of $0 \leq |X_1| \leq \frac{1}{3}d^4l^4s^4$ and $\frac{2}{3}d^4l^4s^4 \leq |Y_1| \leq d^4l^4s^4$, when $|X_1| = \frac{1}{3}d^4l^4s^4$, $|Y_1| = d^4l^4s^4$. Thus we have a cost of edge deletion greater than k', contradicting with the assumption. Hence it is proved that $|X_1|, |Y_1| \geq \frac{1}{3}d^4l^4s^4$.

Next, consider D_1, let $P = (\bigcup_{j=2}^{d} U_j \cap D_1)$ and $Q = (\bigcup_{j=2}^{d} V_j \cap D_1)$. That is, P and Q together represent the set of vertices in D_1 and in all other W_js, $1 < j \leq d$. Next, we prove: $|P| \leq 4s$ and $|Q| \leq 4s$.

Since the vertices in P and Q belong to other W_js, $1 < j \leq d$, thus to include them in D_1, we have to insert a certain number of edges between P, Q and W_1. Denote $X_1^* = U \cap D_1$ and $Y_1^* = V \cap D_1$. We compare the cost of insertions to include these vertices into D_1 and the cost of deletions to remove them out from D_1. Note in the following computation, we compute only the average degree for the vertices in $U' \cap W_1$, since the average degree for $V' \cap W_1$ gives the same result. To meet the

condition of average degree, the insertion cost C_I is lower-bounded as follows:

$$\begin{aligned} C_I &\geq (|D_1 \cap U'| - s)|D_1 \cap V'| - \max\{|E(D_1 \cap U', D_1 \cap V')|\} \\ &\geq ((|Q| + |Y_1| + |Y_1^*|) - s)(|P| + |X_1| + |X_1^*|) - \\ &\quad ((|Q| + |Y_1| + |Y_1^*|)(|P| + |X_1| + |X_1^*|) - |P||Y_1| - |Q||X_1|) \quad (*) \\ &\geq (|Y_1| - s)|P| + (|Q| - s)|X_1| - s|X_i^*| \end{aligned}$$

Inequality (*) holds because to lower-bound C_I, we use the maximum value of $|E(D_1 \cap U', D_1 \cap V')|$. However, for edge deletions, we only need:

$$C_D = |P||Y_1^*| + |Q||X_1^*|$$

Thus, we have:

$$\begin{aligned} C_I - C_D \geq (|Y_1| - |Y_1^*| - s)|P| + (|Q| - s)(|X_1| - |X_1^*|) - \\ 2s|X_1^*| \end{aligned}$$

By contradiction, we assume $|P| \geq 4s$, then:

$$\begin{aligned}
C_I - C_D &\geq 4s|Y_1| - 4s^2 - 4s|Y_1^*| + (|Q| - s)(|X_1| - |X_1^*|) - \\
&\quad 2s|X_1^*| \\
&\geq 4s|Y_1| - 4s^2 - 4s|Y_1^*| - s|X_1| - 2s|X_1^*| \quad (*) \\
&\geq 4s \cdot \frac{1}{3} d^4 l^4 s^4 - 4s^2 - 4s|Y_1^*| - s \cdot d^4 l^4 s^4 \\
&\quad - 2s \cdot dl \quad (**) \\
&\geq \frac{4}{3} d^4 l^4 s^5 - 4s^2 - 4sdl - d^4 l^4 s^5 - 2sdl \\
&\geq \frac{1}{3} d^4 l^4 s^5 - 4s^2 - 5sdl > 0
\end{aligned}$$

In inequality (*), we have $(|Q|-s)(|X_1|-|X_1^*|) = -s|X_1|+|Q|(|X_1|-|X_1^*|)+s|X_1^*| \geq -s|X_1|$. In inequality (**), we have $|Y_1| \geq \frac{1}{3} d^4 l^4 s^4$, $|X_1| \leq d^4 l^4 s^4$, $|X_i^*| \leq dl$ and $|Y_i^*| \leq dl$. This $C_I - C_D$ is clearly greater than 0, indicating it is better to remove all vertices in P and Q. Thus we know that $|P| \geq 4s$, we could remove all vertices in P and Q out from D_1, contradicting with $\mathcal{D}$ is an optimal solution.

By contradiction, if $|Q| \geq 4s$, then:

$$\begin{aligned}
C_I - C_D &\geq (|Y_1| - |Y_1^*| - s)|P| + 3s \cdot (|X_1| - |X_1^*|) - 2s|X_1^*| \\
&\geq 3s \cdot (|X_1| - |X_1^*|) - 2s|X_1^*| \\
&\geq 3s(\frac{1}{3} d^4 l^4 s^4 - dl) - 2sdl \quad (*) \\
&= d^4 l^4 s^5 - 5sdl > 0
\end{aligned}$$

Inequality (*) is correct since $|X_1| \geq \frac{1}{3} d^4 l^4 s^4$ and $|X_1^*| \leq dl$. Thus we know that $|Q|$ cannot be greater than $4s$. Thus we proved that both $|P|$, $|Q|$ are smaller than $4s$, otherwise it would need less cost to delete these vertices from D_1 than to include them in D_1.

Now based on the claims above, we prove that $\forall u \in W_1$, we have $u \in D_1$. We prove this by comparing the cost of including an arbitrary vertex $u \in W_1$ into D_1 (C_I) and the cost of removing u from D_1 (C_D). Without loss of generality, we assume $u \in U_1$. For u, the cost of removing u from D_1 is at least the edges between u and Y_1: $C_D \geq |Y_1| - s - sl \geq \frac{1}{3}d^4l^4s^4 - s - sl$.

To include the u into D_1, we need to delete the edges between u and the vertices outside D_1, insert all the missing edges incident to u and insert the missing edges between u and Q: $C_I \leq |Y_1^*| + s + sl + 4s \leq dl + 5s + sl$. Clearly, $C_D > C_I$, then we conclude for each $u \in W_1$, we have $u \in D_1$ in the optimal solution, i.e., $W_1 \subseteq D_1$. Next we prove that for each vertex $v \in \bigcup_{1 \leq i \leq d} W_i \backslash W_1$, we have $v \notin D_1$ in any optimal solution. In order to show this claim, we compare the cost of including v into D_1 (C_I) and the cost of deleting v from D_1 (C_D). Without loss of generality, we assume $v \in U'$. To have v in D_1, we have to keep the average degree satisfying the criteria:

$$
\begin{aligned}
C_I &\geq (1 + |X_1| + |X_1^*| + |P|)(|Y_1| + |Y_1^*| + |Q|) \\
&\quad - ((1 + |P| + |X_1| + |X_1^*|)(|Q| + |Y_1| + |Y_1^*|) - s(|Y_1| + l) - |Y_1|) \\
&= |Y_1| + sl - s|V_1^*| - 4s \\
&\geq d^4l^4s^4 + sl - sdl - 4s
\end{aligned}
$$

And to remove v from D_1, we just need to delete at most the edges between v and Y_1^* and the edges between v and Q: $C_D \leq dl + 4s$. Obviously, we have $C_I > C_D$, thus it is better to remove every $u \notin W_1$ from D_1. In summary, we have proved that there are d disjoint components in the optimal solution, each of which contains one W_i. For each vertex $u \in G$, in the optimal solution u must be connected to one of the d components. This asks for $k' - k = 2(d-1)d^5l^5s^4$ edge deletions.

Finally, we show that for all D_i, we have $|D_i \cap U| = |D_i \cap V| = l$. First, we prove that $|D_i \cap U| \geq l$ and $|D_i \cap V| \geq l$, for all $1 \leq i \leq d$. By contradiction, if $|D_i \cap U| < l$, then the average degree of the vertices in $D_i \cap U'$ is upper-bounded by:

$$\begin{aligned}
\overline{d_{D_i \cap U'}} &\leq \frac{|D_i \cap V'| \cdot |D_i \cap U'| - s(d^4 l^4 s^4 + l)}{|D_i \cap U'|} \\
&= |D_i \cap V'| - \frac{s(d^4 l^4 s^4 + l)}{|D_i \cap U'|} \\
&< |D_i \cap V'| - s \quad (*)
\end{aligned}$$

Inequality (*) holds since $|D_i \cap U| < l$ and thus we have $|D_i \cap U'| \leq (d^4 l^4 s^4 + l)$. This contradicts with optimal solution. Similarly, we can prove that $|D_i \cap V| \geq l$ for all $1 \leq i \leq d$. Since $|U| = |V| = dl$ and there are d bicliques in the optimal solution, we know that $|D_i \cap U| = |D_i \cap V| = l$. Moreover, in the optimal solution, the induced subgraph $G'[D_i \cap (U \cup V)]$ must form a biclique, otherwise D_i would not be an average-s-biplex. Thus, this requires that G be transformed into d equally-sized balanced bicliques within k edge modifications. Therefore, in summary, in the optimal solution, G must be converted into d disjoint equal-size bicliques and thus gives a solution to ESBE instance. □ □

A.1.6 Proof of Lemma 4.10:

Proof. Obviously, to separate u and v, we must not allow u and v to have any common neighbors. Thus at least $d = \min\{\omega(u, S(u,v)), \omega(v, S(u,v))\}$ deletions are required. If $d > k$, then we cannot afford the cost of deletions. Hence u and v must end up in the same average-s-biplex. □ □

A.1.7 Proof of Lemma 4.11:

Proof. Without loss of generality, let $L \subseteq N^*(u)$ be the subset of $N^*(u)$ that end up in the same average-s-biplex as u, and $M = N^*(u) \backslash L$. Thus we have $|M| \leq k$, since otherwise we would have to delete more than k edges between M and u. Thus $|L| \geq |N^*(u)| - k$. Note that the vertices in M and L are "*interchangeable*", i.e. for any vertex $m \in M$ and $l \in L$, we can exchange the locations of m and l, by putting m in L and l in M. Such location-exchange requires no further edge modification. Suppose in an optimal solution, $m \in M$ is connected to a set of vertices Z. Obviously, $u \notin Z$. Thus to reconnect m to u, we have to re-insert the deleted edge between u and m, and remove all the inserted edges between m and Z. That saves $|Z| + 1$ edge modifications. Next, to move l out and connect l with Z, we have to delete the edge between l and u, and insert the edges between l and Z. This procedure requires $|Z| + 1$ edge modifications. Therefore, M and L are interchangeable. Let $N^*(u) = \{v_1, v_2, ..., v_r\}$, $r \geq k+1$. We can claim that the vertex set $L' = \{v_{k+1}, ..., v_r\}$ is a subset of L and all vertices in L' end up in the same average-s-biplex with u. This claim is true because we cannot afford deleting the edges between u and more than k vertices in $N^*(u)$. Since all vertices in L' end up in the same average-s-biplex, we can merge them together. Thus, Rule 3. is correct. □ □

A.1.8 Proof of Theorem 4.12:

Proof. Let G be a connected component in the optimal solution after applying the reduction rules exhaustively. Since we have removed all existing average-s-biplexes, G must be incident to edge modifications. Thus there exist at most $2k$ connected components. Let $G = (U, V, E)$ and without loss of generality, we assume $|U| \leq |V|$. First we prove $|U| \leq 4k + 6s$.

By contradiction, if $|U| > 4k + 6s$, then obviously we have $|V| > 4k + 6s$. Let u^* be the vertex in U that has the largest degree. Obviously, $d'(u^*) \geq \sigma(V) - s \geq |V| - s$. We distinguish in two cases:

Case i: $\sigma(u^*) > \frac{1}{2}\sigma(U) > 2k + 3s$. Note that for all edges incident to u^*, we can have at most one edge with weight larger than k. This is true because by contradiction, if we have v_1, $v_2 \in V$ and $\omega(u^*, v_1) > k$, $\omega(u^*, v_2) > k$, then we could merge v_1 and v_2 by Rule 2. Since u has at least $|V| - s$ neighbors, we can compute the number of missing edges m_e incident to u as:

$$\begin{aligned} m_e &\geq (|V| - s - 1)(\sigma(u^*) - k) \quad (*) \\ &> (4k + 5s - 1) \cdot \sigma(u^*)/2 \\ &> 4s \cdot \sigma(u^*)/2 \\ &\geq 4s \cdot \sigma(U)/4 \\ &= s \cdot \sigma(U) \end{aligned}$$

Thus the average degree of U is :

$$\begin{aligned} \overline{d(U)} &< \frac{\sigma(U) \cdot \sigma(V) - s \cdot \sigma(U)}{\sigma(U)} \\ &= \sigma(V) - s \end{aligned}$$

Inequality (*) holds because there can be at most one edge incident to u^* with edge weight larger than k and then for the rest of the neighbors (at least $|V| - s - 1$ vertices), each of them is incident to at least $(\sigma(u^*) - k)$ missing edges. Hence G does not satisfy average-s-biplex, we have a contradiction.

Case ii: $\sigma(u^*) \leq \frac{1}{2}\sigma(U)$. Then we first prove that there must exist another vertex u', such that $\omega(u', V) \geq |V| - 2s$. Suppose there does not exist such a vertex, then

we compute the number of missing edges m_e:

$$\begin{aligned} m_e &> (\sigma(V) - (|V| - 2s))(\sigma(U) - \sigma(u^*)) \quad (*) \\ &> 2s \cdot \sigma(U)/2 \\ &> s \cdot \sigma(U) \end{aligned}$$

Thus the average degree of U is smaller than $\sigma(V) - s$. Inequality (*) holds because the degrees of the vertices other than u^* in U are all smaller than $|V| - 2s$. Then for each of such vertices, there are at least $\sigma(V) - (|V| - 2s)$ missing edges incident to it. Hence we proved that there must be a vertex u' with $d'(u') \geq |V| - 2s$. Consider u^* and u', the number of shared neighbors between them is at least:

$$\begin{aligned} &\geq |V| - s + |V| - 2s - |V| \\ &\geq |V| - 3s \\ &\geq 4k + 3s > k \end{aligned}$$

This contradicts with Rule 2., since we can merge u^* and u' together. So far, we have proved that $|U| \leq 4k+6s$. Next we show $|V|$ cannot be larger than $(4k+6s) \cdot s+k$ vertices.

By contradiction, assume $|V| > (4k + 6s) \cdot s + k$. Since G is already an average-s-biplex, there can be at most $\sigma(U) \cdot s \leq s(4k+6s)$ missing edges in G. Hence in V, there can be at most $s(4k+6s)$ vertices incident to missing edge(s). By $|V| > (4k+6s) \cdot s+k$, there are at least k vertices that are connected to all vertices in U. If $|U| \geq 2$, then for every pair of vertices in U, they share more than k common neighbors in V, contradicting with Rule 2. That leaves the only case of $|U| = 1$. Let $U = \{u\}$, We consider two sub-cases:

Case i: If $\sigma(u) > k$, then there are at least k vertices in $v \in V$, such that $\omega(v, U) > k$. This contradicts with Rule 2.

Case ii: If $\sigma(u) \leq k$, then we prove the claim that: $\forall v \in V$, $\sigma(v) = 1$. By contradiction, assume if there exists a vertex v' such that $\sigma(v') > 1$. Then consider two vertices v_1, $v_2 \in \delta(v')$, $\sigma(v_1) = 1$, $\sigma(v_2) = 1$. We must have the common neighbors of v_1 and v_2 be a subset of $\delta(u)$, i.e., $S(v_1, v_2) \subseteq \delta(u)$. Thus $|S(v_1, v_2)| < k$. Then we have $\omega(v_1, S(v_1, v_2)) < k$ and $\omega(v_2, S(v_1, v_2)) < k$. Thus v_1 and v_2 would not be merged, contradicting with Rule 2. Since we proved that $\forall v \in V$, $\sigma(v) = 1$, then $|V|$ cannot be bigger than $k + 1$; otherwise we could apply Rule 3. to reduce it.

In summary, we proved that $|U| \leq 4k + 6s$ and $|V| \leq (4k + 6s) \cdot s + k$. Then the total number of vertices in G is at most $2k(((s + 1)(4k + 6s)) + k)$. □ □

A.1.9 Proof of Theorem 4.13:

Proof. For $s = 0$, the problem is equivalent to BICLUSTER EDITING problem and thus is NP-complete. For any s>1, we give a reduction from BICLUSTER EDITING. The same construction also works for s-DEFECTIVE BICLUSTER DELETION.

Given a BICLUSTER EDITING instance: a bipartite graph $G = (U, V, E)$ and a nonnegative integer k, we construct a new graph G' by adding n components ($n = |U| + |V|$) to G: $\{W_1, W_2, ..., W_n\}$. $W_i = (U_i, V_i, E_i)$, $|U_i| = |V_i| = n^4 + s$. We connect all vertices $u \in U_i$ to all vertices $v \in V$, all vertices $v \in V_i$ to all vertices $u \in U$. Inside each W_i, we have $|E_i| = |U_i||V_i| - s$ edges, i.e. insert all but s edges. The parameter k' is equal to $k + n(n - 1)(n^4 + s)$.

"⇒" Let $B_1, B_2, ..., B_l$ be the l ($l < n$) bicliques in the optimal solution for the BICLUSTER EDITING instance. Then for all $v \in B_i$ ($1 \leq i \leq l$), remove all the edges between v and W_j for all $i \neq j$, $1 \leq j \leq n$. Denote G^* as the result graph from G' after the edge deletions. Clearly, G^* is an s-defective bicluster. Moreover, the total number of edge modifications is equal to the edge modification used for bicluster

editing in G plus the edge deletions afterwards: $k' = k + n(n-1)(n^4+s)$, satisfying the edge modification upper-bound. Hence we have a solution for s-DEFECTIVE BICLUSTER EDITING instance.

"$\Leftarrow$" Let $\{D_1, D_2, ..., D_r\}$, $r \geq 1$ be the connected components in an optimal solution of G'. Denote the resulting graph as G^*. Clearly, each D_i, $1 \leq i \leq r$, is an s-defective biclique.

In order to prove solution for s-DEFECTIVE BICLUSTER EDITING is also optimal solution for BICLUSTER EDITING, we first show for every W_i with $1 \leq i \leq n$, there is a D_i such that $W_i \subseteq D_i$ and $D_i \cap W_j = \emptyset$ for all $1 \leq j \neq i \leq n$. By contradiction, we assume this is not the case. Then there must exist at least one W_i, we assume it to be W_1, such that there is no D_i completely containing W_1 in G^*. Without loss of generality, we assume that $\mathcal{D} = \{D_1, D_2, ..., D_t\}$, $1 \leq t \leq r$, is the set of components that intersect with W_1. We then claim that there exists a D_j, such that $|D_j \cap W_1| \geq (n^4+s)$, $1 \leq j \leq t$. Denote $X_j = D_j \cap U_1$ and $Y_j = D_j \cap V_1$. If we have $|D_j \cap W_1| < (n^4+s)$, for all $1 \leq j \leq t$, then the edges between $W_1 \cap D_j$ and $W_1 \backslash D_j$ must be deleted. Thus the number of edge deletions C_D that we need:

$$
\begin{aligned}
C_D &\geq \frac{1}{2}\sum_{j=1}^{t}(|X_j|(|V_1| - |Y_j|) + |Y_j|(|U_1| - |X_j|)) - s \\
&= \frac{1}{2}\sum_{j=1}^{t}((n^4+s)(|X_j + |Y_j|) - 2|X_j||Y_j|) - s \\
&\geq \frac{1}{2}\sum_{j=1}^{t}((n^4+s)(|X_j + |Y_j|) - \frac{1}{2}(|X_j| + |Y_j|)^2) - s \\
&> \frac{1}{2}\sum_{j=1}^{t}((n^4+s)(|X_j| + |Y_j| - \frac{1}{2}|X_j| - \frac{1}{2}|Y_j|)) - s \quad (*) \\
&= \frac{1}{4}(n^4+s)\sum_{j=1}^{t}(|X_j| + |Y_j|) - s \\
&= \frac{1}{4}(n^4+s)^2 - s \geq k'
\end{aligned}
$$

Inequality (*) is correct since we assume $|D_j \cap W_1| < n^4 + s$, for all $1 \leq j \leq t$. Thus we have $\frac{1}{2}(|X_j| + |Y_j|)^2 < \frac{1}{2}(n^4 + s)(|X_j| + |Y_j|)$. The above inequality shows that if we have $|D_j \cap W_1| < n^4 + s$ for all $1 \leq j \leq t$, then we need more than k' edge modifications, contradicting with the assumption of optimal solution. Thus we know that, for each W_i, there exists a D_j such that $|W_i \cap D_j| \geq (n^4 + s)$. Consider W_1. Let D_1 be the component such that $|W_1 \cap D_1| \geq (n^4 + s)$. We next prove that $|X_1| > \frac{1}{3}(n^4 + s)$ and $|Y_1| > \frac{1}{3}(n^4 + s)$.

By contradiction, without loss of generality, we assume $|X_1| \leq \frac{1}{3}(n^4 + s)$. Then we compute the cost of deletions within W_1:

$$\begin{aligned} C_D &\geq |X_1|(|V_1| - |Y_1|) + |Y_1|(|U_1| - |X_1|) - s \\ &= (n^4 + s)(|X_1| + |Y_1|) - 2|X_1||Y_1| - s \\ &\geq (n^4 + s)(|X_1| + |Y_1|) - \frac{1}{2}(|X_1| + |Y_1|)^2 - s \\ &\geq \frac{4}{3}(n^4 + s)^2 - \frac{8}{9}(n^4 + s)^2 - s \quad (*) \\ &= \frac{4}{9}(n^4 + s)^2 - s \geq k' \end{aligned}$$

In the inequality (*), we compute the minimum value for $(n^4 + s)(|X_1| + |Y_1|) - \frac{1}{2}(|X_1| + |Y_1|)^2$, within the range of $0 \leq |X_1| \leq \frac{1}{3}(n^4 + s)$, $\frac{2}{3}(n^4 + s) \leq |Y_1| \leq n^4 + s$. The result shows that we must have more than k' edge modifications if $|X_1|$ or $|Y_1|$ is at most $\frac{1}{3}(n^4 + s)$, thus contradicting with the assumption of optimal solution.

Next, based on the claims proved above, we show that: (1) $W_1 \subseteq D_1$ and (2) for all j that $j \neq 1$, we have $W_j \cap D_1 = \emptyset$, $1 < j \leq n$. First, for each $u \in W_j$ $1 < j \leq n$, we prove $u \notin D_1$. We compute the C_I as the cost of including u into D_1 and the cost C_D as the cost of removing u from D_1. Without loss of generality, we assume $u \in U_j$:

$$
\begin{aligned}
C_I &\geq |D_1| - s \\
&\geq \frac{1}{3}(n^4 + s) - s \\
C_D &\leq |V| \leq n
\end{aligned}
$$

Obviously, $C_I > C_D$, thus it is better to remove such u from D_1. Similarly, for each $v \in W_1$ we compute the cost C_I of including it into D_1 (to delete edges connecting v and the vertices outside D_1), and the cost C_D of removing v out from D_1. Without loss of generality, we assume $v \in U_1$:

$$
\begin{aligned}
C_I &\leq |V| \leq n \\
C_D &\geq |D_1| - s \\
&\geq \frac{1}{3}(n^4 + s) - s
\end{aligned}
$$

Clearly, it is better to include these vertices as part of D_1. So far, we have proved that for each W_i, there is one D_i such that $W_i \subseteq D_i$. Hence in the optimal solution, we have at least n components, which requires $k' - k$ deletions. If G cannot be edited within k edge modifications into disjoint set of bicliques, there must be more than s missing edges in some component in the result graph G^*. Thus we must be able to modify G into a set of disjoint bicliques, in order to have an optimal solution for s-DEFECTIVE BICLUSTER EDITING. In conclusion, if we have an optimal solution for s-DEFECTIVE BICLUSTER EDITING, then the induced subgraph of the result graph $G^*[U \cup V]$ gives a solution to BICLUSTER EDITING. □ □

A.2 Appendix B: Pseudo-code for Edge Deletion Heuristics

Algorithm 1 EDGE_DEL_MAIN(G)

1: $cost = cost \leftarrow$TRANSITIVE_CLOSURE_COST(G);
2: **if** ($cost == 0$) **then**
3: $return(null, 0)$;
4: **end if**
5: $actions \leftarrow null; delcost \leftarrow 0$;
6: **while** (G is still connected) **do**
7: $uv \leftarrow$ REMOVE_CULPRIT(G);
8: $actions.add(uv)$;
9: $delcost+ = s(uv)$;
10: **end while**
11: // Adjust actions such that it only contains the edge removals contributing to the separation of two subgraphs of G
12: //Assume G is cut into G_1 and G_2
13: **while** (uv in actions) **do**
14: **if** (both u, v are in G_1 or G_2) **then**
15: $actions.remove(uv)$;
16: **end if**
17: **end while**
18: //Solve the problem in a recursive manner for G_1 and G_2, until no better solution can be found
19: **if** (delcost >= cost) **then**
20: $return(null, cost)$;
21: **end if**
22: $(list1, cost1) \leftarrow$ EDGE_DEL_HEURISTICS(G_1);
23: **if** ($delcost + cost1 >= cost$) **then**
24: $return(null, cost)$;
25: $(list2, cost2) \leftarrow$ EDGE_DEL_HEURISTICS(G_2);
26: **end if**
27: **if** ($delcost + cost1 + cost2 >= cost$) **then**
28: $return(null, cost)$;
29: **end if**
30: $actions.add(list1)$;
31: $actions.add(list2)$;
32: //add all the edge insertions required for the closure of transitivity
33: $actions.add(insertions)$;
34: $return(actions, delcost + cost1 + cost2)$;

Bibliography

[1] Ash A Alizadeh, Michael B Eisen, R Eric Davis, Chi Ma, Izidore S Lossos, Andreas Rosenwald, Jennifer C Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[2] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[3] Uri Alon, Naama Barkai, Daniel A Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

[4] Leonard Amaral, Marta Martins, and Miguel Viveiros. Enhanced killing of intracellular multidrug-resistant mycobacterium tuberculosis by compounds that affect the activity of efflux pumps. *Journal of antimicrobial chemotherapy*, 59(6):1237–1246, 2007.

[5] Noga Amit. *The bicluster graph editing problem.* PhD thesis, Tel Aviv University, 2004.

[6] Eric Anderson, Gilman D Veith, and David Weininger. *SMILES, a line notation and computerized interpreter for chemical structures.* US Environmental Protection Agency, Environmental Research Laboratory, 1987.

[7] Marcia Angell. *The truth about the drug companies: How they deceive us and what to do about it.* Random House Trade Paperbacks New York, 2005.

[8] Scott A Armstrong, Jane E Staunton, Lewis B Silverman, Rob Pieters, Monique L den Boer, Mark D Minden, Stephen E Sallan, Eric S Lander, Todd R Golub, and Stanley J Korsmeyer. Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature genetics*, 30(1):41–47, 2002.

[9] Pierre Baldi and G Wesley Hatfield. *DNA microarrays and gene expression: from experiments to data analysis and modeling.* Cambridge university press, 2002.

[10] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[11] Tanya Barrett, Stephen E Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F Kim, Maxim Tomashevsky, Kimberly A Marshall, Katherine H Phillippy, Patti M Sherman, Michelle Holko, et al. Ncbi geo: archive for functional genomics data setsâĂŤupdate. *Nucleic acids research*, 41(D1):D991–D995, 2013.

[12] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of computational biology*, 10(3-4):373–384, 2003.

[13] D. A. Benson, M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. GenBank. *Nucleic Acids Res.*, 41(Database issue):36–42, Jan 2013.

[14] Dennis A Benson, Karen Clark, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic acids research*, 43(Database issue):D30, 2015.

[15] Sven Bergmann, Jan Ihmels, and Naama Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical review E*, 67(3):031902, 2003.

[16] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[17] Pavel Berkhin and Jonathan D Becher. Learning simple relations: Theory and applications. In *SDM*, pages 420–436. SIAM, 2002.

[18] S Böcker, S Briesemeister, QBA Bui, and A Truß. Peace: Parameterized and exact algorithms for cluster editing. *Manuscript, Lehrstuhl für Bioinformatik, Friedrich-Schiller-Universität Jena (September 2007)*, 2007.

[19] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[20] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[21] Stanislav Busygin, Gerrit Jacobsen, Ewald Krämer, and Contentsoft Ag. Double conjugated clustering applied to leukemia microarray data. In *In 2nd SIAM ICDM, Workshop on clustering high dimensional data*. Citeseer, 2002.

[22] Stanislav Busygin, Oleg Prokopyev, and Panos M Pardalos. Biclustering in data mining. *Computers & Operations Research*, 35(9):2964–2987, 2008.

[23] Andrea Califano, Gustavo Stolovitzky, Yuhai Tu, et al. Analysis of gene expression microarrays for phenotype classification. In *Ismb*, volume 8, pages 75–85, 2000.

[24] Chuck Chakrapani. *Statistics in market research*. Oxford University Press, 2004.

[25] Christine M Chan, Xia Jing, Laura A Pike, Qiong Zhou, Dong-Jun Lim, Sharon B Sams, Gregory S Lund, Vibha Sharma, Bryan R Haugen, and Rebecca E Schweppe. Targeted inhibition of src kinase with dasatinib blocks thyroid cancer growth and metastasis. *Clinical Cancer Research*, 18(13):3580–3591, 2012.

[26] D Chan, JW Tyner, WJ Chng, C Bi, R Okamoto, J Said, BD Ngan, GD Braunstein, and HP Koeffler. Effect of dasatinib against thyroid cancer cell lines in vitro and a xenograft model in vivo. *Oncol Lett*, 3(4):807–815, 2012.

[27] Bernard Chen, Phang C Tai, Robert Harrison, and Yi Pan. Novel hybrid hierarchical-k-means clustering method (hk-means) for microarray analysis. In *Computational Systems Bioinformatics Conference, 2005. Workshops and Poster Abstracts. IEEE*, pages 105–108. IEEE, 2005.

[28] Tung-Shou Chen, Tzu-Hsin Tsai, Yi-Tzu Chen, Chin-Chiang Lin, Rong-Chang Chen, Shuan-Yow Li, and Hsin-Yi Chen. A combined k-means and hierarchical clustering method for improving the clustering efficiency of microarray. In *Intelligent Signal Processing and Communication Systems, 2005. ISPACS 2005. Proceedings of 2005 International Symposium on*, pages 405–408. IEEE, 2005.

[29] Y. Cheng and G. M. Church. Biclustering of expression data. *Proc Int Conf Intell Syst Mol Biol*, 8:93–103, 2000.

[30] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[31] Annie P Chiang and Atul J Butte. Systematic evaluation of drug–disease relationships to identify leads for novel drug uses. *Clinical Pharmacology & Therapeutics*, 86(5):507–510, 2009.

[32] Hyuk Cho, Inderjit S Dhillon, Yuqiang Guan, and Suvrit Sra. Minimum sum-squared residue co-clustering of gene expression data. In *SDM*, volume 3, page 3. SIAM, 2004.

[33] Raymond J Cho, Michael J Campbell, Elizabeth A Winzeler, Lars Steinmetz, Andrew Conway, Lisa Wodicka, Tyra G Wolfsberg, Andrei E Gabrielian, David Landsman, David J Lockhart, et al. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular cell*, 2(1):65–73, 1998.

[34] CG Chute. Classification and retrieval of patient records using natural language: an experimental application of latent semantic analysis. In *Engineering in Medicine and Biology Society, 1991. Vol. 13: 1991., Proceedings of the Annual International Conference of the IEEE*, pages 1162–1163. IEEE, 1991.

[35] Christopher G Chute and Yiming Yang. An evaluation of concept based latent semantic indexing for clinical information retrieval. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 639. American Medical Informatics Association, 1992.

[36] UniProt Consortium et al. Uniprot: a hub for protein information. *Nucleic acids research*, page gku989, 2014.

[37] AJ Crowle, GS Douvas, and MH May. Chlorpromazine: a drug potentially useful for treating mycobacterial infections. *Chemotherapy*, 38(6):410–419, 1992.

[38] Simone Daminelli, V Joachim Haupt, Matthias Reimann, and Michael Schroeder. Drug repositioning through incomplete bi-cliques in an integrated drug–target–disease network. *Integrative Biology*, 4(7):778–788, 2012.

[39] Allan Peter Davis, Cynthia J Grondin, Kelley Lennon-Hopkins, Cynthia Saraceni-Richards, Daniela Sciaky, Benjamin L King, Thomas C Wiegers, and Carolyn J Mattingly. The comparative toxicogenomics database's 10th year anniversary: update 2015. *Nucleic acids research*, page gku935, 2014.

[40] Allan Peter Davis, Cynthia J Grondin, Kelley Lennon-Hopkins, Cynthia Saraceni-Richards, Daniela Sciaky, Benjamin L King, Thomas C Wiegers, and Carolyn J Mattingly. The comparative toxicogenomics database's 10th year anniversary: update 2015. *Nucleic acids research*, 43(D1):D914–D920, 2015.

[41] Milind Dawande, Pinar Keskinocak, Jayashankar M Swaminathan, and Sridhar Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41(2):388–403, 2001.

[42] Milind Dawande, Pinar Keskinocak, and Sridhar Tayur. On the biclique problem in bipartite graphs. Technical report, GSIA Working Paper 1996-04, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 1996.

[43] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[44] Frank Dehne, Michael A Langston, Xuemei Luo, Sylvain Pitre, Peter Shaw, and Yun Zhang. The cluster editing problem: Implementations and experiments. In *Parameterized and Exact Computation*, pages 13–24. Springer, 2006.

[45] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[46] Chris Ding, Ya Zhang, Tao Li, and Stephen R Holbrook. Biclustering protein complex interactions with a biclique finding algorithm. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 178–187. IEEE, 2006.

[47] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

[48] Joel T Dudley, Tarangini Deshpande, and Atul J Butte. Exploiting drug–disease relationships for computational drug repositioning. *Briefings in bioinformatics*, page bbr013, 2011.

[49] Joel T Dudley, Marina Sirota, Mohan Shenoy, Reetesh K Pai, Silke Roedder, Annie P Chiang, Alex A Morgan, Minnie M Sarwal, Pankaj Jay Pasricha, and Atul J Butte. Computational repositioning of the anticonvulsant topiramate for inflammatory bowel disease. *Science translational medicine*, 3(96):96ra76–96ra76, 2011.

[50] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.

[51] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[52] S Elvidge. Getting the drug repositioning genie out of the bottle. *Life Science Leader*, 2010.

[53] Kemal Eren, Mehmet Deveci, Onur Küçüktunç, and Ümit V Çatalyürek. A comparative analysis of biclustering algorithms for gene expression data. *Briefings in bioinformatics*, 14(3):279–292, 2013.

[54] Seth Falcon and Robert Gentleman. Using gostats to test gene lists for go term association. *Bioinformatics*, 23(2):257–258, 2007.

[55] Michael R Fellows and RG Downey. Parameterized complexity, 1999.

[56] Jörg Flum and Martin Grohe. Parameterized complexity theory, volume xiv of texts in theoretical computer science. an eatcs series, 2006.

[57] Nir Friedman and Moises Goldszmidt. Learning bayesian networks with local structure. In *Learning in graphical models*, pages 421–459. Springer, 1998.

[58] Nir Friedman, Iftach Nachman, and Dana Peér. Learning bayesian network structure from massive datasets: the «sparse candidate «algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc., 1999.

[59] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[60] L Fu, C Fu-Liu, et al. Thalidomide and tuberculosis. *The International Journal of Tuberculosis and Lung Disease*, 6(7):569–572, 2002.

[61] Michael R Garey and David S Johnson. Computers and intractability: a guide to the theory of np-completeness. 1979. *San Francisco, LA: Freeman*, 1979.

[62] Audrey P Gasch, Mingxia Huang, Sandra Metzner, David Botstein, Stephen J Elledge, and Patrick O Brown. Genomic expression responses to dna-damaging agents and the regulatory role of the yeast atr homolog mec1p. *Molecular biology of the cell*, 12(10):2987–3003, 2001.

[63] Audrey P Gasch, Paul T Spellman, Camilla M Kao, Orna Carmel-Harel, Michael B Eisen, Gisela Storz, David Botstein, and Patrick O Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular biology of the cell*, 11(12):4241–4257, 2000.

[64] Gad Getz, Erel Levine, and Eytan Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences*, 97(22):12079–12084, 2000.

[65] Jeffrey W Godden, Ling Xue, and Jürgen Bajorath. Combinatorial preferences affect molecular similarity/diversity calculations using binary fingerprints and tanimoto coefficients. *Journal of Chemical Information and Computer Sciences*, 40(1):163–166, 2000.

[66] Kwang-Il Goh, Michael E Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-Laszlo Barabasi. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

[67] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[68] Assaf Gottlieb, Gideon Y Stein, Eytan Ruppin, and Roded Sharan. Predict: a method for inferring novel drug indications with application to personalized medicine. *Molecular systems biology*, 7(1):496, 2011.

[69] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica*, 39(4):321–347, 2004.

[70] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

[71] Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8):718–726, 2009.

[72] Jiong Guo, Falk Hüffner, Christian Komusiewicz, and Yong Zhang. Improved algorithms for bicluster editing. In *Theory and Applications of Models of Computation*, pages 445–456. Springer, 2008.

[73] Jiong Guo, Iyad A Kanj, Christian Komusiewicz, and Johannes Uhlmann. Editing graphs into disjoint unions of dense clusters. *Algorithmica*, 61(4):949–970, 2011.

[74] Jiong Guo, Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. A more relaxed model for graph-based data clustering: s-plex cluster editing. *SIAM Journal on Discrete Mathematics*, 24(4):1662–1683, 2010.

[75] Soyang Ha, Young-Ju Seo, Min-Seok Kwon, Byung-Ha Chang, Cheol-Kyu Han, and Jeong-Hyeok Yoon. Idmap: facilitating the detection of potential leads with therapeutic targets. *Bioinformatics*, 24(11):1413–1415, 2008.

[76] Ada Hamosh, Alan F Scott, Joanna S Amberger, Carol A Bocchini, and Victor A McKusick. Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(suppl 1):D514–D517, 2005.

[77] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[78] John A Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

[79] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[80] TO SUBSPACE CLUSTERING HAVE. A number of approaches to subspace clustering have been proposed in the past two decades. 2011.

[81] Scott J Hebbring. The challenges, advantages and future of phenome-wide association studies. *Immunology*, 141(2):157–165, 2014.

[82] Ingrid Hedenfalk, David Duggan, Yidong Chen, Michael Radmacher, Michael Bittner, Richard Simon, Paul Meltzer, Barry Gusterson, Manel Esteller, Mark Raffeld, et al. Gene-expression profiles in hereditary breast cancer. *New England Journal of Medicine*, 344(8):539–548, 2001.

[83] Kenneth Higbee. Mathematical classification and clustering. *Technometrics*, 40(1):80–80, 1998.

[84] Lucia A Hindorff, Praveen Sethupathy, Heather A Junkins, Erin M Ramos, Jayashri P Mehta, Francis S Collins, and Teri A Manolio. Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proceedings of the National Academy of Sciences*, 106(23):9362–9367, 2009.

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

[86] Yosef Hochberg and Yoav Benjamini. More powerful procedures for multiple significance testing. *Statistics in medicine*, 9(7):811–818, 1990.

[87] Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, Adetayo Kasim, Tatsiana Khamiakova, Suzy Van Sanden, Dan Lin, Willem Talloen, et al. Fabia: factor analysis for bicluster acquisition. *Bioinformatics*, 26(12):1520–1527, 2010.

[88] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *IJCAI*, volume 99, pages 688–693, 1999.

[89] Yu-Fen Huang, Hsiang-Yuan Yeh, and Von-Wun Soo. Inferring drug-disease associations from integration of chemical, genomic and phenotype data using network propagation. *BMC medical genomics*, 6(Suppl 3):S4, 2013.

[90] Timothy R Hughes, Matthew J Marton, Allan R Jones, Christopher J Roberts, Roland Stoughton, Christopher D Armour, Holly A Bennett, Ernest Coffey, Hongyue Dai, Yudong D He, et al. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

[91] Hanaa M Hussain, Khaled Benkrid, Huseyin Seker, and Ahmet T Erdogan. Fpga implementation of k-means algorithm for bioinformatics application: An accelerated approach to clustering microarray data. In *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on*, pages 248–255. IEEE, 2011.

[92] Trey Ideker, Vesteinn Thorsson, Jeffrey A Ranish, Rowan Christmas, Jeremy Buhler, Jimmy K Eng, Roger Bumgarner, David R Goodlett, Ruedi Aebersold, and Leroy Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 292(5518):929–934, 2001.

[93] Francesco Iorio, Roberta Bosotti, Emanuela Scacheri, Vincenzo Belcastro, Pratibha Mithbaokar, Rosa Ferriero, Loredana Murino, Roberto Tagliaferri, Nicola Brunetti-Pierri, Antonella Isacchi, et al. Discovery of drug mode of action and drug repositioning from transcriptional responses. *Proceedings of the National Academy of Sciences*, 107(33):14621–14626, 2010.

[94] Vishwanath R Iyer, Michael B Eisen, Douglas T Ross, Greg Schuler, Troy Moore, Jeffrey CF Lee, Jeffrey M Trent, Louis M Staudt, James Hudson, Mark S Boguski, et al. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283(5398):83–87, 1999.

[95] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[96] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[97] Andrew D Johnson and Christopher J O'Donnell. An open access database of genome-wide association results. *BMC medical genetics*, 10(1):1, 2009.

[98] Michael J Keiser, Vincent Setola, John J Irwin, Christian Laggner, Atheir I Abbas, Sandra J Hufeisen, Niels H Jensen, Michael B Kuijer, Roberto C Matos, Thuy B Tran, et al. Predicting new molecular targets for known drugs. *Nature*, 462(7270):175–181, 2009.

[99] Halil Kilicoglu, Dongwook Shin, Marcelo Fiszman, Graciela Rosemblat, and Thomas C Rindflesch. Semmeddb: a pubmed-scale repository of biomedical semantic predications. *Bioinformatics*, 28(23):3158–3160, 2012.

[100] Andrew D King, N Pržulj, and Igor Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, 2004.

[101] Sarah L Kinnings, Nina Liu, Nancy Buchmeier, Peter J Tonge, Lei Xie, and Philip E Bourne. Drug discovery using chemical systems biology: repositioning the safe medicine comtan to treat multi-drug and extensively drug resistant tuberculosis. *PLoS Comput Biol*, 5(7):e1000423, 2009.

[102] Robert J Klein, Caroline Zeiss, Emily Y Chew, Jen-Yue Tsai, Richard S Sackler, Chad Haynes, Alice K Henning, John Paul SanGiovanni, Shrikant M Mane, Susan T Mayne, et al. Complement factor h polymorphism in age-related macular degeneration. *Science*, 308(5720):385–389, 2005.

[103] Ulf Klein, Yuhai Tu, Gustavo A Stolovitzky, Michela Mattioli, Giorgio Cattoretti, Hervé Husson, Arnold Freedman, Giorgio Inghirami, Lilla Cro, Luca Baldini, et al. Gene expression profiling of b cell chronic lymphocytic leukemia reveals a homogeneous phenotype related to memory b cells. *The Journal of experimental medicine*, 194(11):1625–1638, 2001.

[104] H Kluger, B Kacinski, Y Kluger, O Mironenko, M Gilmore-Hebert, J Chang, AS Perkins, and E Sapi. Microarry analysis of invasive and metastatic phenotypes in a breast cancer model. In *Poster presented at the Gordon Conference on Cancer*, 2001.

[105] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.

[106] Craig Knox, Vivian Law, Timothy Jewison, Philip Liu, Son Ly, Alex Frolkis, Allison Pon, Kelly Banco, Christine Mak, Vanessa Neveu, et al. Drugbank 3.0: a comprehensive resource for âĂŸomicsâĂŹ research on drugs. *Nucleic acids research*, 39(suppl 1):D1035–D1041, 2011.

[107] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[108] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[109] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *AAAI/IAAI*, pages 580–587, 1998.

[110] Eugene V Koonin, Natalie D Fedorova, John D Jackson, Aviva R Jacobs, Dmitri M Krylov, Kira S Makarova, Raja Mazumder, Sergei L Mekhedov, Anastasia N Nikolskaya, B Sridhar Rao, et al. A comprehensive evolutionary classification of proteins encoded in complete eukaryotic genomes. *Genome biology*, 5(2):R7, 2004.

[111] Ekaterina Kotelnikova, Anton Yuryev, Ilya Mazo, and Nikolai Daraselia. Computational approaches for drug repositioning and combination therapy design. *Journal of bioinformatics and computational biology*, 8(03):593–606, 2010.

[112] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, page gkv1075, 2015.

[113] Justin Lamb, Emily D Crawford, David Peck, Joshua W Modell, Irene C Blat, Matthew J Wrobel, Jim Lerner, Jean-Philippe Brunet, Aravind Subramanian, Kenneth N Ross, et al. The connectivity map: using gene-expression signatures to connect small molecules, genes, and disease. *science*, 313(5795):1929–1935, 2006.

[114] Laura Lazzeroni, Art Owen, et al. Plaid models for gene expression data. *Statistica sinica*, 12(1):61–86, 2002.

[115] Laura Lazzeroni, Art Owen, et al. Plaid models for gene expression data. *Statistica sinica*, 12(1):61–86, 2002.

[116] Dennis Lee, Ronald Cornet, Francis Lau, and Nicolette De Keizer. A survey of snomed ct implementations. *Journal of biomedical informatics*, 46(1):87–96, 2013.

[117] Guojun Li, Qin Ma, Haibao Tang, Andrew H Paterson, and Ying Xu. Qubic: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic acids research*, 37(15):e101–e101, 2009.

[118] Jiao Li, Xiaoyan Zhu, and Jake Yue Chen. Building disease-specific drug-protein connectivity maps from molecular interaction networks and pubmed abstracts. *PLoS Comput Biol*, 5(7):e1000450, 2009.

[119] Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

[120] Jinze Liu and Wei Wang. Op-cluster: Clustering by tendency in high dimensional space. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 187–194. IEEE, 2003.

[121] Xiaowen Liu and Lusheng Wang. Computing the maximum similarity bi-clusters of gene expression data. *Bioinformatics*, 23(1):50–56, 2007.

[122] SP Lloyd. Least square quantization in pcm. bell telephone laboratories paper. published in journal much later: Lloyd, sp: Least squares quantization in pcm. *IEEE Trans. Inform. Theor.(1957/1982)*.

[123] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

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

[125] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *WALCOM: Algorithms and Computation*, pages 274–285. Springer, 2009.

[126] Kira S Makarova, Yuri I Wolf, and Eugene V Koonin. Archaeal clusters of orthologous genes (arcogs): An update and application for analysis of shared features between thermococcales, methanococcales, and methanobacteriales. *Life*, 5(1):818–840, 2015.

[127] Timothy Mather, Vaheh Oganessyan, P Hof, R Huber, S Foundling, C Esmon, and W Bode. The 2.8 a crystal structure of gla-domainless activated protein c. *The EMBO journal*, 15(24):6822, 1996.

[128] David W Matula. *Graph theoretic techniques for cluster analysis algorithms*. Department of Computer Science, Southern Methodist University, 1976.

[129] Michael L Metzker. Sequencing technologiesâĂŤthe next generation. *Nature reviews genetics*, 11(1):31–46, 2010.

[130] John H Morris, Leonard Apeltsin, Aaron M Newman, Jan Baumbach, Tobias Wittkop, Gang Su, Gary D Bader, and Thomas E Ferrin. clustermaker: a multi-algorithm clustering plugin for cytoscape. *BMC bioinformatics*, 12(1):436, 2011.

[131] Supratim Mukherjee, Dimitri Stamatis, Jon Bertsch, Galina Ovchinnikova, Olena Verezemska, Michelle Isbandi, Alex D Thomas, Rida Ali, Kaushal Sharma, Nikos C Kyrpides, et al. Genomes online database (gold) v. 6: data updates and feature enhancements. *Nucleic Acids Research*, page gkw992, 2016.

[132] Asher Mullard. 2015 fda drug approvals. *Nature Reviews Drug Discovery*, 15(2):73–76, 2016.

[133] Joseph Mullen, Simon J Cockell, Hannah Tipney, Peter M Woollard, and Anil Wipat. Mining integrated semantic networks for drug repositioning opportunities. *PeerJ*, 4:e1558, 2016.

[134] TM Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pacific Symposium on Biocomputing*, volume 8, pages 77–88, 2003.

[135] Takeru Nakazato, Hidemasa Bono, Hideo Matsuda, and Toshihisa Takagi. Gendoo: functional profiling of gene and disease features using mesh vocabulary. *Nucleic acids research*, page gkp483, 2009.

[136] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[137] Rolf Niedermeier. Invitation to fixed-parameter algorithms. 2006.

[138] Rolf Niedermeier and Peter Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47(2):63–77, 2003.

[139] Paul A Novick, Oscar F Ortiz, Jared Poelman, Amir Y Abdulhay, and Vijay S Pande. Sweetlead: an in silico database of approved drugs, regulated chemicals, and herbal isolates for computer-aided drug discovery. *PloS one*, 8(11):e79568, 2013.

[140] Ali Oghabian, Sami Kilpinen, Sampsa Hautaniemi, and Elena Czeizler. Biclustering methods: biological relevance and application in gene expression analysis. *PloS one*, 9(3):e90801, 2014.

[141] G Oroszlán, J Sanderud, OD Saugstad, and L Lakatos. [d-penicillamine: old drug, new indication? d-penicillamine reduced pulmonary hypertension induced by free radicals]. *Orvosi hetilap*, 133(44):2835–6, 1992.

[142] Jagadeesh Patchala and Anil G Jegga. Concept modeling-based drug repositioning. *AMIA Summits on Translational Science Proceedings*, 2015:222, 2015.

[143] Steven M Paul, Daniel S Mytelka, Christopher T Dunwiddie, Charles C Persinger, Bernard H Munos, Stacy R Lindborg, and Aaron L Schacht. How to improve r&d productivity: the pharmaceutical industry's grand challenge. *Nature reviews Drug discovery*, 9(3):203–214, 2010.

[144] Ted Pedersen, Serguei VS Pakhomov, Siddharth Patwardhan, and Christopher G Chute. Measures of semantic similarity and relatedness in the biomedical domain. *Journal of biomedical informatics*, 40(3):288–299, 2007.

[145] René Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[146] Robert M Plenge. Disciplined approach to drug discovery and early development. *Science Translational Medicine*, 8(349):349ps15–349ps15, 2016.

[147] Scott L Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M Sturla, Michael Angelo, Margaret E McLaughlin, John YH Kim, Liliana C Goumnerova, Peter M Black, Ching Lau, et al. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

[148] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[149] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[150] Fábio Protti, Maise Dantas da Silva, and Jayme Luiz Szwarcfiter. Applying modular decomposition to parameterized bicluster editing. In *Parameterized and Exact Computation*, pages 1–12. Springer, 2006.

[151] John Quackenbush. Computational analysis of microarray data. *Nature reviews genetics*, 2(6):418–427, 2001.

[152] Majid Rastegar-Mojarad, Zhan Ye, Jill M Kolesar, Scott J Hebbring, and Simon M Lin. Opportunities for drug repositioning from phenome-wide association studies. *Nature biotechnology*, 33(4):342–345, 2015.

[153] David J Reiss, Nitin S Baliga, and Richard Bonneau. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC bioinformatics*, 7(1):1, 2006.

[154] Ryan M Rich, Philip J Rosenfeld, Carmen A Puliafito, Sander R Dubovy, Janet L Davis, Harry W Flynn Jr, Serafin Gonzalez, William J Feuer, Richard C Lin, Geeta A Lalwani, et al. Short-term safety and efficacy of intravitreal bevacizumab (avastin) for neovascular age-related macular degeneration. *Retina*, 26(5):495–511, 2006.

[155] Peter W Rose, Chunxiao Bi, Wolfgang F Bluhm, Cole H Christie, Dimitris Dimitropoulos, Shuchismita Dutta, Rachel K Green, David S Goodsell, Andreas Prlić, Martha Quesada, et al. The rcsb protein data bank: new resources for research and education. *Nucleic acids research*, 41(D1):D475–D482, 2013.

[156] Richard Röttger, Prabhav Kalaghatgi, Peng Sun, Siomar de Castro Soares, Vasco Azevedo, Tobias Wittkop, and Jan Baumbach. Density parameter estimation for finding clusters of homologous proteins-tracing actinobacterial pathogenicity life styles. *Bioinformatics*, pages 215–222, 2012.

[157] Richard Röttger, Christoph Kreutzer, Thuy Duong Vu, Tobias Wittkop, and Jan Baumbach. Online transitivity clustering of biological data with missing values. In *GCB*, pages 57–68, 2012.

[158] Gerard Salton. Developments in automatic text retrieval. *Science*, 253(5023):974, 1991.

[159] Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael DiCuccio, Scott Federhen, et al. Database resources of the national center for biotechnology information. *Nucleic acids research*, 39(suppl 1):D38–D51, 2011.

[160] Eran Segal, Alexis Battle, and Daphne Koller. Decomposing gene expression into cellular processes. In *Pacific Symposium on Biocomputing*, volume 8, pages 89–100. World Scientific, 2003.

[161] Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(suppl 1):S243–S252, 2001.

[162] Stephen B Seidman and Brian L Foster. A graph-theoretic generalization of the clique concept*. *Journal of Mathematical Sociology*, 6(1):139–154, 1978.

[163] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1):173–182, 2004.

[164] Jay Shendure and Hanlee Ji. Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145, 2008.

[165] Qizheng Sheng, Yves Moreau, and Bart De Moor. Biclustering microarray data by gibbs sampling. *Bioinformatics*, 19(suppl 2):ii196–ii205, 2003.

[166] Xi-Nan Shi, Hongjian Li, Hong Yao, Xu Liu, Ling Li, Kwong-Sak Leung, Hsiang-fu Kung, Di Lu, Man-Hon Wong, and Marie Chia-mi Lin. In silico identification and in vitro and in vivo validation of anti-psychotic drug fluspirilene as a potential cdk2 inhibitor and a candidate anti-cancer drug. *PloS one*, 10(7):e0132072, 2015.

[167] Steven L Soignet, Peter Maslak, Zhu-Gang Wang, Suresh Jhanwar, Elizabeth Calleja, Laura J Dardashti, Diane Corso, Anthony DeBlasio, Janice Gabrilove, David A Scheinberg, et al. Complete remission after treatment of acute promyelocytic leukemia with arsenic trioxide. *New England Journal of Medicine*, 339(19):1341–1348, 1998.

[168] Robert R Sokal. Clustering and classification: Background and current directions. *Classification and clustering*, pages 1–15, 1977.

[169] Paul T Spellman, Gavin Sherlock, Michael Q Zhang, Vishwanath R Iyer, Kirk Anders, Michael B Eisen, Patrick O Brown, David Botstein, and Bruce Futcher. Comprehensive identification of cell cycle–regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular biology of the cell*, 9(12):3273–3297, 1998.

[170] Wolfram Stacklies, Henning Redestig, Matthias Scholz, Dirk Walther, and Joachim Selbig. pcamethodsâĂŤa bioconductor package providing pca methods for incomplete data. *Bioinformatics*, 23(9):1164–1167, 2007.

[171] Christoph Steinbeck, Yongquan Han, Stefan Kuhn, Oliver Horlacher, Edgar Luttmann, and Egon Willighagen. The chemistry development kit (cdk): An open-source java library for chemo-and bioinformatics. *Journal of chemical information and computer sciences*, 43(2):493–500, 2003.

[172] Alexander Sturn, John Quackenbush, and Zlatko Trajanoski. Genesis: cluster analysis of microarray data. *Bioinformatics*, 18(1):207–208, 2002.

[173] Peng Sun, Jan Baumbach, and Jiong Guo. Efficient large-scale bicluster editing. In *German Bioinformatics Conference 2014*, pages 54–60.

[174] Peng Sun, Jiong Guo, and Jan Baumbach. Integrated simultaneous analysis of different biomedical data types with exact weighted bi-cluster editing. *J Integr Bioinform*, 17, 2012.

[175] Peng Sun, Jiong Guo, and Jan Baumbach. Integrated simultaneous analysis of different biomedical data types with exact weighted bi-cluster editing. *Journal of Integrative Bioinformatics*, 9(2):197, 2012.

[176] Peng Sun, Jiong Guo, and Jan Baumbach. Biclue-exact and heuristic algorithms for weighted bi-cluster editing of biomedical data. In *BMC proceedings*, volume 7, page S9. BioMed Central Ltd, presented at GLBIO2013, 2013.

[177] Peng Sun, Jiong Guo, and Jan Baumbach. Complexity of dense bicluster editing problems. *Computing and Combinatorics*, pages 154–165, 2014.

[178] Peng Sun, Jiong Guo, Rainer Winnenburg, and Jan Baumbach. Integrated literature mining and drug-gene-disease triangulation reveals ten thousand new purposes for existing medication. *Drug Discovery Today*, (In press), 2016.

[179] Peng Sun, Nora K Speicher, Richard Röttger, Jiong Guo, and Jan Baumbach. Bi-force: large-scale bicluster editing and its application to gene expression data biclustering. *Nucleic Acids Research*, page gku201, 2014.

[180] Ryota Suzuki and Hidetoshi Shimodaira. Pvclust: an r package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, 22(12):1540–1542, 2006.

[181] S Joshua Swamidass. Mining small-molecule screens to repurpose drugs. *Briefings in bioinformatics*, 12(4):327–335, 2011.

[182] Fujian Tan, Ruizhi Yang, Xiaoxue Xu, Xiujie Chen, Yunfeng Wang, Hongzhe Ma, Xiangqiong Liu, Xin Wu, Yuelong Chen, Lei Liu, et al. Drug repositioning by applying âĂŸexpression profilesâĂŹ generated by integrating chemical structure similarity and gene semantic similarity. *Molecular BioSystems*, 10(5):1126–1138, 2014.

[183] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18 Suppl 1:S136–144, 2002.

[184] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl 1):S136–S144, 2002.

[185] Chun Tang, Li Zhang, Aidong Zhang, and Murali Ramanathan. Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In *Bioinformatics and Bioengineering Conference, 2001. Proceedings of the IEEE 2nd International Symposium on*, pages 41–48. IEEE, 2001.

[186] Roman L Tatusov, Eugene V Koonin, and David J Lipman. A genomic perspective on protein families. *Science*, 278(5338):631–637, 1997.

[187] Robert Tibshirani, Trevor Hastie, Mike Eisen, Doug Ross, David Botstein, Pat Brown, et al. Clustering methods for the analysis of dna microarray data. *Dept. Statist., Stanford Univ., Stanford, CA, Tech. Rep*, 1999.

[188] Heather Turner, Trevor Bailey, and Wojtek Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational statistics & data analysis*, 48(2):235–254, 2005.

[189] Lyle Ungar and Dean P Foster. A formal statistical approach to collaborative filtering. *CONALDâĂŹ98*, 1998.

[190] Jesse CJ van Dam, Peter J Schaap, Vitor AP Martins dos Santos, and María Suárez-Diez. Integration of heterogeneous molecular networks to unravel gene-regulation in mycobacterium tuberculosis. *BMC systems biology*, 8(1):111, 2014.

[191] Joachim Von Eichborn, Manuela S Murgueitio, Mathias Dunkel, Soeren Koerner, Philip E Bourne, and Robert Preissner. Promiscuous: a database for network-based drug-repositioning. *Nucleic acids research*, 39(suppl 1):D1060–D1066, 2011.

[192] Haixun Wang, Wei Wang, Jiong Yang, and Philip S Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 394–405. ACM, 2002.

[193] Tongsen Wang, Lei Wang, Zhijun Xie, and Ronghua Yang. Data compression algorithm based on hierarchical cluster model for sensor networks. In *2008 Second International Conference on Future Generation Communication and Networking*, volume 2, pages 319–323. IEEE, 2008.

[194] Yongcui Wang, Shilong Chen, Naiyang Deng, and Yong Wang. Drug repositioning by kernel-based integration of molecular structure, molecular activity, and phenotype data. *PloS one*, 8(11):e78518, 2013.

[195] John N Weinstein, Timothy G Myers, Patrick M O'Connor, Stephen H Friend, Albert J Fornace, Kurt W Kohn, Tito Fojo, Susan E Bates, Lawrence V Rubinstein, N Leigh Anderson, et al. An information-intensive approach to the molecular pharmacology of cancer. *Science*, 275(5298):343–349, 1997.

[196] Danielle Welter, Jacqueline MacArthur, Joannella Morales, Tony Burdett, Peggy Hall, Heather Junkins, Alan Klemm, Paul Flicek, Teri Manolio, Lucia Hindorff, et al. The nhgri gwas catalog, a curated resource of snp-trait associations. *Nucleic acids research*, 42(D1):D1001–D1006, 2014.

[197] Max Wintermark, Gregory W Albers, Andrei V Alexandrov, Jeffry R Alger, Roland Bammer, Jean-Claude Baron, Stephen Davis, Bart M Demaerschalk, Colin P Derdeyn, Geoffrey A Donnan, et al. Acute stroke imaging research roadmap. *American Journal of Neuroradiology*, 29(5):e23–e30, 2008.

[198] David S Wishart, Craig Knox, An Chi Guo, Savita Shrivastava, Murtaza Hassanali, Paul Stothard, Zhan Chang, and Jennifer Woolsey. Drugbank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic acids research*, 34(suppl 1):D668–D672, 2006.

[199] David S Wishart, Dan Tzur, Craig Knox, Roman Eisner, An Chi Guo, Nelson Young, Dean Cheng, Kevin Jewell, David Arndt, Summit Sawhney, et al. Hmdb: the human metabolome database. *Nucleic acids research*, 35(suppl 1):D521–D526, 2007.

[200] Tobias Wittkop, Dorothea Emig, Sita Lange, Sven Rahmann, Mario Albrecht, John H Morris, Sebastian Böcker, Jens Stoye, and Jan Baumbach. Partitioning biological data with transitivity clustering. *Nature methods*, 7(6):419–420, 2010.

[201] Tobias Wittkop, Dorothea Emig, Sita Lange, Sven Rahmann, Mario Albrecht, John H Morris, Sebastian Böcker, Jens Stoye, and Jan Baumbach. Partitioning biological data with transitivity clustering. *Nature methods*, 7(6):419–420, 2010.

[202] Tobias Wittkop, Dorothea Emig, Anke Truss, Mario Albrecht, Sebastian Böcker, and Jan Baumbach. Comprehensive cluster analysis with transitivity clustering. *Nature protocols*, 6(3):285–295, 2011.

[203] Tobias Wittkop, Dorothea Emig, Anke Truss, Mario Albrecht, Sebastian Böcker, and Jan Baumbach. Comprehensive cluster analysis with transitivity clustering. *Nature protocols*, 6(3):285–295, 2011.

[204] Tobias Wittkop, Sven Rahmann, Richard Röttger, Sebastian Böcker, and Jan Baumbach. Extension and robustness of transitivity clustering for protein–protein interaction network analysis. *Internet Mathematics*, 7(4):255–273, 2011.

[205] Christian Wiwie, Jan Baumbach, and Richard Röttger. Comparing the performance of biomedical clustering methods. *Nature methods*, 12(11):1033–1038, 2015.

[206] Weiling Xu, Thomas Koeck, Abigail R Lara, Donald Neumann, Frank P DiFilippo, Michelle Koo, Allison J Janocha, Fares A Masri, Alejandro C Arroliga, Constance Jennings, et al. Alterations of cellular bioenergetics in pulmonary artery endothelial cells. *Proceedings of the National Academy of Sciences*, 104(4):1342–1347, 2007.

[207] Jiong Yang, Haixun Wang, Wei Wang, and Philip Yu. Enhanced biclustering on expression data. In *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*, pages 321–327. IEEE, 2003.

[208] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 517–528. IEEE, 2002.

[209] Lun Yang and Pankaj Agarwal. Systematic drug repositioning based on clinical side-effects. *PloS one*, 6(12):e28025, 2011.

[210] Mihalis Yannakakis. Edge-deletion problems. *SIAM Journal on Computing*, 10(2):297–309, 1981.

[211] Hao Ye, Qi Liu, and Jia Wei. Construction of drug network based on side effects and its application for drug repositioning. *PloS one*, 9(2):e87864, 2014.

[212] V Wee Yong, Sophie Chabot, Olaf Stuve, and Gary Williams. Interferon beta in the treatment of multiple sclerosis mechanisms of action. *Neurology*, 51(3):682–689, 1998.

[213] Haiyuan Yu, Alberto Paccanaro, Valery Trifonov, and Mark Gerstein. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics*, 22(7):823–829, 2006.

[214] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Semi-supervised dimensionality reduction. In *SDM*, pages 629–634. SIAM, 2007.

[215] Jun Zhang, Rod Chiodini, Ahmed Badr, and Genfa Zhang. The impact of next-generation sequencing on genomics. *Journal of genetics and genomics*, 38(3):95–109, 2011.

[216] Lihong Zhang, Patrick F Byrne, and Elizabeth AH Pilon-Smits. Mapping quantitative trait loci associated with selenate tolerance in arabidopsis thaliana. *New Phytologist*, 170(1):33–42, 2006.

[217] Ping Zhang, Fei Wang, and Jianying Hu. Towards drug repositioning: a unified computational framework for integrating multiple aspects of drug similarity and disease similarity. In *AMIA Annual Symposium Proceedings*, volume 2014, page 1258. American Medical Informatics Association, 2014.