Integrated simultaneous analysis of different biomedical data types with exact weighted bi-cluster editing

Peng Sun[1,2,3,*], Jiong Guo[2,3], Jan Baumbach[1,2]

[1] Computational Systems Biology group, Max Planck Institute for Informatics, Campus E1.4, 66123 Saarbrücken, Germany

[2] Cluster of Excellence for Multimodal Computing and Interaction, Saarland University, Campus E1.7, 66123 Saarbrücken, Germany

[3] Saarland University, Campus E1.7, 66123 Saarbrücken, Germany

Summary

The explosion of biological data has largely influenced the focus of today's biology research. Integrating and analysing large quantity of data to provide meaningful insights has become the main challenge to biologists and bioinformaticians. One major problem is the combined data analysis of data from different types, such as phenotypes and genotypes. This data is modelled as bi-partite graphs where nodes correspond to the different data points, mutations and diseases for instance, and weighted edges relate to associations between them. Bi-clustering is a special case of clustering designed for partitioning two different types of data simultaneously. We present a bi-clustering approach that solves the NP-hard weighted bi-cluster editing problem by transforming a given bi-partite graph into a disjoint union of bi-cliques. Here we contribute with an exact algorithm that is based on fixed-parameter tractability. We evaluated its performance on artificial graphs first. Afterwards we exemplarily applied our Java implementation to data of genome-wide association studies (GWAS) data aiming for discovering new, previously unobserved geno-to-pheno associations. We believe that our results will serve as guidelines for further wet lab investigations. Generally our software can be applied to any kind of data that can be modelled as bi-partite graphs. To our knowledge it is the fastest exact method for weighted bi-cluster editing problem.

1 Introduction

1.1 Background

The focus of recent biologically motivated studies has shifted due to the explosive growth of available (sequential) data emerging from laboratories worldwide. For example, GenBank now stores over 197,000,000 sequences from more than 380,000 organisms [9]. UniProtKb/Swiss-Prot has incorporated ~53,000 annotated sequences, gathered from 205,244 (published) references and PDB (Protein Data Bank) provides approximately 78,400 molecule structures including proteins, nucleic acids (NA) and protein/NA complexes. Integrating, processing and analysing these large quantities of different types of data from various sources have become a main challenge in bioinformatics. Modern bioinformatics algorithms and computational approaches have been proven to have great potential. The application of carefully designed problem specific models and methodology allows for discovering novel interrelations and gaining further insights into the raw data, forming a better understanding. In

[*] To whom correspondence should be addressed. Email: psun@mpi-inf.mpg.de

this paper we concentrate on the simultaneous analysis of data from different types. We will exemplarily concentrate on so-called Genome-Wide Association Studies (GWAS).

GWAS is one of the fastest emerging areas of today's biological research, also known as Whole Genome Association Study. It is an examination of genetic variants (genotypes) to check if any of them is associated 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 bi-partite 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 [11], the number of GWAS publications is growing dramatically. Up to June 2011, there have been 951 publications on GWAS, according to National Human Genome Research Institute (NHGRI) Catalog of Published Genome-Wide Association Studies [10]. Although the discovered associations have revealed much insight on the mechanisms of common diseases/traits, 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 modelled as graphs, where vertices correspond to SNPs (genotypes) and traits (phenotypes) while edges symbolize significant associations between them (Figure 1). We proceed one step forward by associating a group of sequence variations (SNPs) to a group of traits/diseases, forming a "group to group" association, rather than the traditional "one to one". Therefore, we developed an exact algorithm for weighted bi-cluster editing. We applied it to different GWAS datasets and discovered new associations that have not been reported before. We believe such results, based on several associations instead of one pair-wise relation, are thereby with higher confidence.

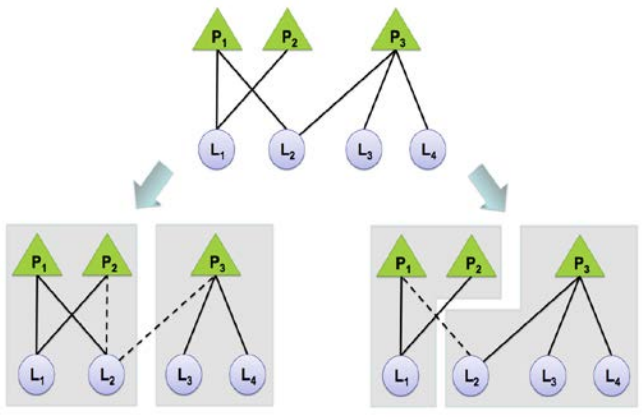


Figure 1: Bi-partite graph representation of GWAS data. Vertices P1, P2, P3 represent "phenotypes" and L1, L2, L3, L4 represent "SNP loci". Our bi-cluster editing algorithm converts the intransitive GWAS data graph into disjoint bi-cliques. Two putative solutions are presented at the bottom: the addition of edge P2-L2 and the deletion of edge P3-L2 (left) as well as the deletion of edge P1-L2 (right). In the unweighted bi-cluster editing problem we would prefer the right solution since we only need to modify one edge. However, in the more realistic

weighted scenario the preferred solution depends on the concrete edge weights: if the costs for adding P2-L2 and removing P3-L2 are lower than the costs for removing P1-L2, we would prefer the left solution.

1.2 Cluster Editing and Bi-cluster Editing

Data clustering is a classical task in computational biology. Its goal is to partition a data set into clusters such that elements within a cluster are more similar according to one or many specific characteristics than elements in different clusters. Clustering methods are extensively used in every area of biological studies (e.g. functional genomics, protein/DNA sequences analysis, almost all kinds of biological network analysis) [14]. However, in some other scenarios, the standard clustering model is not satisfactory. One of them is the clustering of gene expression data under different conditions, which can be modelled as a *bipartite graph* [1]. In such cases, clustering only genes or only conditions often does not yield sufficient insight. Instead, we would like to find subsets of genes and subsets of conditions that together behave in a consistent way. This type of clustering methods is called *bi-clustering*.

Clustering and bi-clustering are quite similar, so they share a number of similar strategies. A common strategy for clustering is to choose a similarity threshold and construct the corresponding graph according to the following rules: (1) the entities/objects refer to the vertices in the graph, and (2) an edge is drawn between two vertices if and only if the similarity between them exceeds a given threshold [15]. Under such setting, we call the end points u, v of an edge “similar”, written as $u \sim v$. However, the constructed graph is not necessarily “transitive”, which means $u \sim v$, $u \sim w$ does not necessarily imply $v \sim w$. We aim to convert the preliminarily constructed graph into a graph only consisting of disjoint clusters with minimal costs (minimal number of edge deletions/insertions, for instance). Such problems are named “cluster editing”. Formal definitions are given below:

V is denoted as the set of vertices (objects) to be clustered. $\binom{V}{k}$ is denoted as the set of k-element subsets of V. uv is an unordered pair of $\{u, v\} \in \binom{V}{2}$. The similarity between two vertices is a symmetric similarity function $s: \binom{V}{2} \to \mathbb{R}$. We call u and v similar, $u \sim v$, if and only if $s(u, v) > a\ given\ threshold$. The edge set of the similarity graph is $E := \{uv: u \sim v\}$. Self-loops are not permitted in our graphs.

If the graph satisfies any of the equivalent conditions below, then we call it “transitive”:

(1) For any three vertices $uvw \in \binom{V}{3}$, $uv \in E$ and $vw \in E \Longrightarrow uw \in E$ holds.

(2) No paths of length 2 in the graph, i.e., for each uvw, we have $|E \cap \{uv, vw, uw\}| \neq 2$.

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

Given a graph $G = (V, E)$, we convert G into a transitive graph $G' = (V, E')$ by inserting and deleting edges. Each insertion and deletion of uv incurs a certain cost of $s(uv)$. Define $cost(G \to G')$ to be the cost of conversion, $cost(G \to G') = s(E \backslash E') - s(E' \backslash E)$. Our goal is to find a G', such that $cost(G \longrightarrow G')$ is minimized.

For bi-cluster editing, we have a similar strategy. The graphs are constructed in the same way, where vertices refer to objects and edges represent the similarity between two vertices. The only difference is that the resulting graph is a *bipartite graph*. Bipartite graphs are special graphs satisfying the following conditions: (1) the vertices of the graph can be divided into two subsets V_1 and V_2, (2) edges can only be defined between the vertices from different subsets, i.e., $s(u,v) \to \mathbb{R}$ if and only if $u \in V_1, v \in V_2$ or $u \in V_2, v \in V_1$.

For a bipartite graph $G = (V_1, V_2, E)$, we have the following equivalent conditions characterizing it if G is transitive:

(1) For each subset of four vertices, $uvwx \in \binom{V}{4}$, where $\{u,w\} \in \binom{V_1}{2}$, $\{v,x\} \in \binom{V_2}{2}$, we have $uv \in E, wv \in E$ and $wx \in E \Longrightarrow ux \in E$.

(2) G does not contain a path of 4 vertices, i.e., for each $uvwx \in \binom{V}{4}$, where $\{u,w\} \in \binom{V_1}{2}$, $\{v,x\} \in \binom{V_2}{2}$, we have $|E \cap \{uv, wv, ux, wx\}| \neq 3$.

(3) G is a disjoint union of bi-cliques (i.e. complete bipartite graphs).

Bi-cluster editing is similar to its counterpart cluster editing: We convert a given bipartite graph into a set of disjoint union of bi-cliques by inserting/deleting edges with minimal costs. This problem is called “bi-cluster editing”. The definition of the $cost(G \to G')$ is the same.

1.3 Problem Statement

The *weighted bi-cluster editing problem* is formally defined as follows: Given an undirected bipartite graph $G = (V_1, V_2, E, s)$, where s is a similarity function $s: (\binom{V_1}{1}, \binom{V_2}{1}) \to \mathbb{R}$, compute $\delta(G) := \min\{cost(G \longrightarrow G')\}$ and find one or all G^*, such that $cost(G \longrightarrow G^*) = \delta(G)$.

1.4 Previous Studies and Results

The unweighted version of this problem, *unweighted bi-cluster editing*, has a similarity function of $s: \left(\binom{V_1}{1}, \binom{V_2}{1}\right) \to \{+1, -1\}$. The editing cost is defined as $cost(G \to G') = |E \backslash E'| + |E' \backslash E|$. Both the weighted and unweighted cases of bi-cluster editing problems are NP-hard, proven by N. Amit [2].

Unlike its counterpart, cluster editing, which has been extensively studied [13-15], the study of bi-cluster editing is far from complete. F. Protti *et al.* [3] developed an algorithm that finished in $O(4^k + |V| + |E|)$ for the unweighted version of bi-cluster editing. Later J. Guo *et al.* [4] improved the running time to $O(3.24^k + |E|)$, by developing an improved branching strategy. However, most real life graphs are weighted and to our knowledge no exact algorithm for weighted bi-partite graphs exists so far.

1.5 Our Contributions

Here, we present a fixed-parameter algorithm for the weighted bi-cluster editing problem. We assume $|s(uv)| > 1$ for all $u \in V_1, v \in V_2$. Our algorithm checks in $O(4^k)$ time if there is a set of insertions and deletions that converts the given graph into disjoint bi-cliques. We implemented the algorithm in Java and evaluated its performance on artificially generated graphs. Our experiments show that the algorithm can give exact solutions at least to medium-sized graphs within acceptable running times. To our knowledge, it is the fastest exact algorithm for weighted problem instances so far.

Furthermore, we applied our software on Genome-Wide Association Studies, searching for new associations between genotypes and different medical traits. Our algorithm successfully solved all but two of the GWAS data instances in reasonable time and thereby found 86 new associations.

1.6 Preliminaries

The vertex set is denoted as $V = \{v_1, v_2, \ldots, v_n\}$. The input to our algorithm is a graph $G = (V_1, V_2, E)$, with similarity function $s(uv) \rightarrow \mathbb{R}$ and a similarity threshold. E denotes the edge set, $E = \{v_1, v_2 \colon s(v_1, v_2) > threshold\}$. The output of the algorithm is a list of solutions and their corresponding costs (see 2.2 for details).

Without loss of generality, the input graph is assumed to consist of one single connected component. If not, we can treat each connected component separately. Obviously, an optimal solution will never join separate components, since we can always find a solution with less cost where the disjoint components remain separated, than the solutions linking the separated components together [4].

We use “P4” as the short form of “a path of 4 vertices”. A P4 is also the “*basic conflict element*” in our problem, i.e. the “conflict P4”. As mentioned above, a bipartite graph is a complete bipartite graph if and only if it contains no conflict P4. Therefore, our goal is to remove all these P4s by edge insertions and deletions. Let $B(G)$ be the set of all the basic conflict elements, i.e. $B(G) = \{uvwx \in (V, 4) \mid |E \cap (uv, wv, ux, wx)| = 3\}$ $u, w \in V_1, v, x \in V_2$. G is transitive if and only if $B(G) = \{\}$.

2 Fixed-Parameter Algorithm

2.1 Introduction to Fixed-Parameter Algorithm

Fixed-parameter algorithm and fixed-parameter tractability were concepts introduced by Downey and Fellows in 1990s [6]. They provided a possible way of solving NP-hard problems more efficiently. A problem is called “fixed-parameter tractable” regarding to a certain parameter, if it can be solved in a running time of $O(f(k) \cdot |I|^c)$, where f is a function that solely depends on the parameter k, $|I|$ is the input size and c is a constant. A more recently overview of the fixed-parameter algorithms can be found in [7].

Here, we present the first fixed-parameter algorithm for the weighted bi-cluster editing problem with the parameter k as the costs for edge modifications. Given a problem instance, i.e. an intransitive connected component, the algorithm finds the optimal solution with cost at most k if there is any solution. Our algorithm accepts a running time of $O(4^k)$ with real weighted edges. The minimum editing cost is required to be >1, in order to guarantee this running time, since for arbitrarily small edge weights, no fixed-parameter algorithm is able to solve the problem in a provable running time unless P = NP.

2.2 Algorithm

2.2.1 Data Reduction

We reduce the problem size by identifying existing disjoint bi-cliques since they do not need to be repaired and thereby are not considered in the following steps. Initially, all disjoint bi-cliques are removed from the original graph. Afterwards, the algorithm recognizes all the disjoint connected components as individual input graph. In the next step, for each component we check whether it is already a bi-clique or not. If this is the case, the algorithm deletes the whole bi-clique from the input and reports the corresponding component as one bi-cluster. This procedure can be carried out within $O(|V| + |E|)$ time.

2.2.2 Branching Strategy

In this section, we present a search tree algorithm for weighted bi-cluster editing in bipartite graphs. Our goal is to repair all the P4s using edge insertions and edge deletions. For each P4, basically we have 4 possibilities to convert it into a bi-clique/bi-cliques: we can remove either one of the three edges in the P4, resulting in two bi-cliques (one with three vertices and the other with only one isolated vertex, i.e. a singleton), or insert the missing edge such that four vertices form one bi-clique (refer to Figure 2). The details are elucidated as following:

Let $uvwx$ be a P4, where $u, w \in V_1$ and $v, x \in V_2$. We assume $(uv), (wv), (wx) \in E$ (Figure 2a). Afterwards, we recursively check the following four cases to repair the P4.

(1) Insert the missing edge ux and set ux to “permanent” (Figure 2b)

(2) Delete the edge uv and set uv to “forbidden” (Figure 2c)

(3) Delete the edge wv and set wv to “forbidden” (Figure 2d)

(4) Delete the edge wx and set wx to “forbidden” (Figure 2e)

Once a P4 is located in a connected component, the search tree algorithm starts. For each P4, four branches in the search tree are created; each of them represents one of the editing possibilities mentioned above. Then the four branches are visited one by one, performing the corresponding editing behaviour and updating k to $k' = k - (\textit{costs required for insertions and deletions})$. Afterwards, the program searches the new graph for more P4s. The whole procedure is implemented in a recursive manner. If the corresponding editing behaviour would lead to $k' < 0$, then the whole branch in the search tree is skipped. Solutions are identified and recorded by the algorithm when no P4 can be found in the graph and $k' > 0$. The algorithm stops when the entire search tree is visited, and returns all the solutions found, i.e. all bi-cliques. This branching strategy accepts a worst case running time of $O(4^k)$.

2.2.3 Algorithm Procedure

Our algorithm takes graphs as input and outputs a list of solutions. A *Solution* object is structured as a pair (*actions*, *cost*), where *actions* is a list of *action* objects (an *action* object represents an editing behaviour, either an edge insertion or edge deletion) and *cost* is the total cost of the edit behaviours. In addition to the main algorithm, we have four auxiliary functions: P4_FINDER(G), BI_CLIQUE_REMOVER(G), TAKE_ACTION(*action*) and ROLLBACK_ACTION(*action*). P4_FINDER(G) is responsible to find P4s in the graph. BI_CLIQUE_REMOVER(G) removes the existing bi-cliques. TAKE_ACTION(*action*) performs the edge insertion or deletion according to the *action* object and ROLLBACK_ACTION(*action*) removes the effect of the *action* and restores the graph to its previous condition. The subroutine BRANCHING(G, *action*, *solution_list*) creates the branches in the search tree algorithm and recursively visits them. It first carries out the given *action*, then checks the graph to see if there is any P4 unsolved. If any P4 found, then the subroutine continues to check the 4 possibilities to repair it, or if no P4 is found, a *solution* object will be created and put into the *solution_list*. The pseudo-code of the algorithm is described below:

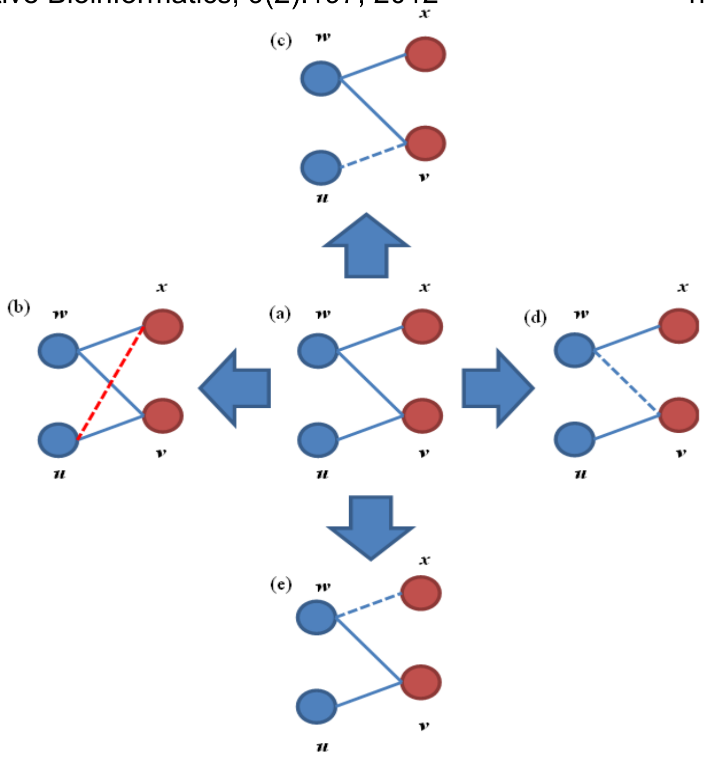
```
//1. MAIN ALGORITHM
solution_list ← empty list
//find the first P4
```

```
$uvwx$ ← P4_FINDER($G$)

if ($uvwx$ = = null)
    return (empty list)
//branching
$action$ ← insert $ux$
BRANCHING ($G$, *action*, *solution_list*)
*action* ← delete $uv$
BRANCHING ($G$, *action*, *solution_list*)
*action* ← delete $wv$
BRANCHING ($G$, *action*, *solution_list*)
*action* ← delete $wx$
//return the results
return *solution_list*

//2. SUBROUTINE BRANCHING($G$, *action*, *solution_list*)
BRANCHING($G$, *action*, *solution_list*)
    if ($k$ <= 0)
        ROLLBACK_ACTION(*action*)
        return;

    TAKE_ACTION(*action*)
    //find the first P4
    $uvwx$ ← P4_FINDER($G$)
    //if no P4 is found, then we have a solution
    if ($uvwx$ = = null)
        *solution* ← (*actions*, *cost*)
        *solution_list*.add(*solution*)
        ROLLBACK_ACTION(*action*)
        return (empty list)
    //Start branching
    $action$ ← insert $ux$
    BRANCHING ($G$, *action*, *solution_list*)
    *action* ← delete $uv$
    BRANCHING ($G$, *action*, *solution_list*)
    *action* ← delete $wv$, *cost* ← $s(wv)$
    BRANCHING ($G$, *action*, *solution_list*)
    *action* ← delete $wx$, *cost* ←$s(wx)$
    ROLLBACK_ACTION(*action*)
end subroutine
```

3 Results

We implemented our algorithm in JAVA 1.6 with support for parallel multi-core computing. For evaluation, we first applied our software on artificially generated data and later on real GWAS results from two different sources. All measurements were taken on Compute Clusters with 78 compute nodes consisting of 2×Intel XEON E5430 2.66 Ghz (Quad-core) CPUs and 16 GB RAM.

Figure 2. The graph cluster editing strategy based on P4-branching. Blue dashed lines relate to edge deletion and red dashed line corresponds to edge insertions. There are four options for



repairing a conflict P4 depicted in (a): Insertion of the missing edge *ux* (b), deletion of the edge *wx* (c), deletion of the edge *wv* (d), and deletion of the edge *uv* (e).

3.1 Artificial Graphs

We generated random artificial graphs as follows. Assume we have a graph consisting of n vertices, we randomly pick up k vertices ($k \in [1, n]$) and define them to be in one bi-clique. Then we carry out the procedure in the remaining $n - k$ vertices until there is no vertex left. This random graph generator gives us a graph consisting of random numbers of clusters of random sizes. The edge weights between vertices are obtained from Gaussian distributions $N(\mu, \sigma^2)$. Two Gaussian distributions were used to generate weights for edges: $N\ (\mu_{intra}, \sigma_{intra}^2)$ and $N\ (\mu_{inter}, \sigma_{inter}^2)$. The former was used to generate weights for edges between two vertices belonging to the same bi-clusters, and the latter for vertices connecting two different bi-cliques. If μ and σ are carefully chosen, then we are able to construct an "almost transitive" bipartite graph. In our case, we chose $\mu_{intra} = 21$, $\mu_{inter} = -21$, $\sigma_{intra} = \sigma_{inter} = 18$. The probability of finding an "inter-edge" (an edge between vertices in different bi-clusters) or an "intra-missing-edge" (missing edge between vertices in the same bi-cluster) is about 0.123 for each node pair.

Table 1 shows the performance of our bi-cluster editing algorithms on artificial graphs. Each running time and cost is averaged over 5 repeats of graphs of the same size but with different edge sets. We can see our algorithm works very fast on small-sized and medium-sized graphs. However, as the size grows, the running times grow drastically; the underlying algorithmic problem is still NP-hard. When the artificially generated graphs contain more than 40 vertices, our algorithm cannot finish within reasonable time. Figure 3 visualizes the running times of our algorithm against the graph component complexity (here we define graph complexity as $|V| \cdot |E|$, for all the artificial graphs we generated). The running times are comparably small at the beginning but hit higher levels very fast as the complexity increases.

No. of Vertices	No. of Edges	No. of Vertices		Running Times (s)		Cost	
		Vertex Set1	Vertex Set2	Ave.	Std.	Ave.	Std.
10	[10,19]	5	5	0.049	0.551	113.02	45.208
20	[35,75]	10	10	0.605	0.512	237.902	72.291
25	[64,130]	15	10	5.061	0.927	333.246	15.847
30	[86,201]	15	15	275.336	124.045	466.903	104.176
35	[104,258]	20	15	1141.572	104.606	1257.912	44.223
40	[161,319]	20	20	3053.877	498.884	2411.566	378.547

Table 1: Results on artificial graphs with different numbers of vertices, including the averages and standard deviations of cost and running times. Costs and running times are averaged over 5 repeats on 5 inputs.

3.2 Genome-Wide Association Studies

In order to demonstrate the applicability of our algorithm 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.*[8], 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 [10]. 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.

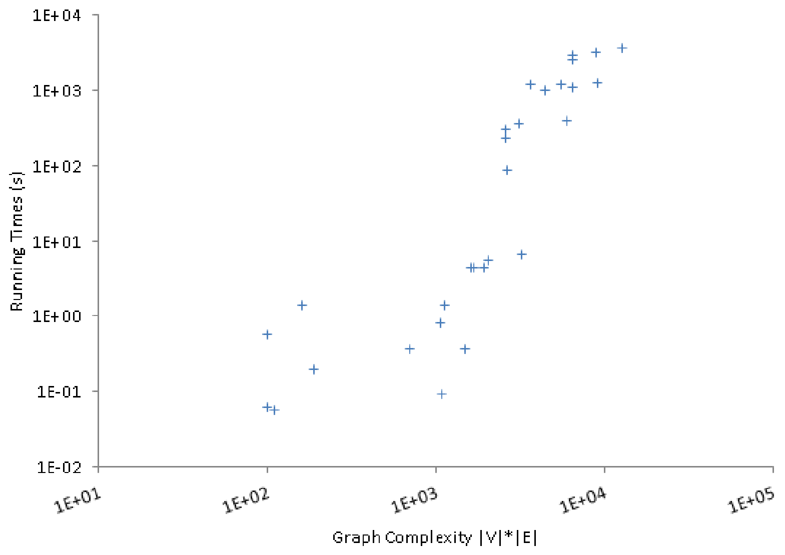


Figure 3. Running times of our fixed-parameter algorithm for varying graph complexities, i.e. $|V| \cdot |E|$, of artificial graphs.

Due to the incompatibility of terminologies utilized in these two data sources, we did not merge the two datasets. 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. Figure 4 shows a histogram of the initial distribution of component sizes $|V|$. Note that we excluded two graph components from Figure 4 because of their exceptionally large sizes (one from NHGRI dataset with size $|V|$ of 3,609 vertices, and one from Johnson's dataset with size $|V|$ of 50,161 vertices).

We applied the fixed-parameter algorithm separately on each disjoint connected component and identified exact solutions for 413 components (99.5% of all the components). We found in total 86 new associations that were not detected as significant in the two GWAS studies. Table 2 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 has 10 new associations. Note that our predictions are largely related to the user-given similarity threshold, i.e. 0.05 in our studies. Supplementary table 1 gives the details of the new associations.

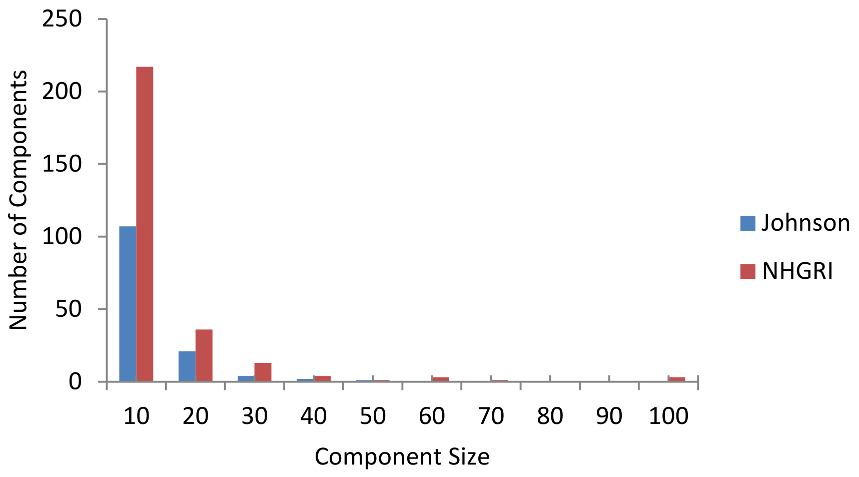


Figure 4. Distribution of the connected component sizes $|V|$ of the graphs generated from two GWAS data sources. The red bars represent the data from NHGRI and blue bars represent data from Johnson's online dataset. The figure does not include the two biggest connected components; one from NHGRI (3,609 vertices) and one from Johnson's online dataset (50,161 vertices).

4 Discussion and Conclusion

Here in this study we have brought forward the first exact algorithm based on fixed-parameter tractability for bi-cluster editing. The speed-up of our strategy is mainly based on the assumption that bi-partite graphs generated from real world data, such as GWAS, are not too far from transitivity. We showed that our algorithm is able to find exact solutions for small-sized and medium-sized components within reasonable time. When the sizes of the component exceed a certain value (around 40 vertices), the running times explode and become unreasonable, at least on standard desktop PCs.

We also applied our algorithm to two different GWAS datasets. Our results show that the algorithm works 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 associated with new phenotypes are 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 labelled with “Plasma levels of Protein C” previously (Protien C is one of the important plasma coagulation factors [17]). 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[16]. These results might imply the confidence of the newly discovered associations before any experiments performed for verification.

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

Table 2: New associations obtained from bi-clustering editing. The items with “*” come from NHGRI dataset while the remaining emerge from Johnson et al.’s online dataset.

Moreover, there are plenty of other potential applications for our algorithm, especially in the area of clustering different types of data: e.g. the identification of genetic variants that are responsible for certain bacterial life styles will require clustering on both genes and species. In the future we will investigate such applications.

To further understand and improve the performance of the fixed-parameter algorithm, more complex data reduction and branching procedures are required. The very similar problem, cluster editing, has been extensively studied. Therefore, it might be interesting to compare the two problems, making full use of the ideas and techniques for cluster editing problems to achieve better running times of bi-cluster editing problems as well.

Acknowledgements

All authors wish to thank the Cluster of Excellence for Multimodal Computing and Interaction of the German Research Foundation for financial support.

References

[1] A. Tanay, R. Sharan and R. Shamir. *Handbook of Computational Molecular Biology* In Handbook of Computational Molecular Biology (2004)

[2] N. Amit. The bicluster graph editing problem. Master's Thesis, Tel Aviv University, School of Mathematical Sciences (2004)

[3] F. Protti, M. D. da Silva and J. L. Szwarcfiter. Applying modular decomposition to parameterized bicluster editing. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 1-12. Springer, Heidelberg (2006)

[4] J Guo, F. Hüffner, C. Komusiewicz and Y. Zhang. Improved algorithms for bicluster editing. In TAMC'08: *Proceedings of the 5th international conference on Theory and applications of models of computation*, pp. 445-456, Berlin, Heidelberg, Springer Verlag. (2008)

[5] N. Ailon, N. Avigdor-Elgrabli and E. Liberty. An improved algorithm for bipartite correlation clustering. *CoRR*, abs/1012.3011. (2010)

[6] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer. (1999)

[7] R. Niedermeier. Invitation to Fixed-Parameter Algorithm. Oxford University Press. (2006)

[8] A. D. Johnson and D. J. O'Donnel. An open access database of genome-wide association results. *BMC Med Genet* 10:6 (2009)

[9] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell and E. W. Sayers. GenBank. *Nucleic Acids Res.*, 39, D32-D37 (2011)

[10] L. A. Hindorff, P. Sethupathy, H. A. Junkins, E. M. Ramos, J. P Mehta, F. S. Collins and T. A. Manolio. Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. Doi:10.1073 *PNAS* (2009)

[11] R. J. Klein, C. Zeiss, E. Y. Chew, J. Y. Tsai, R. S. Sackler, C. Haynes, A. K. Henning, J. P. SanGiovanni, S. M. Mane, S. T. Mayne, M. B. Bracken, F. L. Ferris, J. Ott, C. Barnstable, J. Hoh. Complement Factor H Polymorphism in Age-Related Macular Degeneration. *Science* 308(5720): 385-9. (2005)

[12] S. Böcker, S. Briesemeister, Q. B. A. Bui, A. Truss. Going weighted: Parameterized algorithms for cluster editing. In: Proc. 2nd COCA. Lecture Notes in Computer Science, vol. 5165, pp. 1-12 Springer, Berlin (2008)

[13] J. Guo. A more effective linear kernelization for cluster editing. *Theor. Comp. Sc.* 410, 718-726 (2009)

[14] T. Wittop, D. Emig, S. Lange, S. Rahmann, M. Albrecht, J. H. Morris, S. Böcker, J. Stoye and J. Baumbach. Partitioning biological data with transitivity clustering. *Nat. Methods* 7(6), 419-420(2010)

[15] T. Wittop, D. Emig, A. Trusss, M. Albrecht. S. Böcker, J. Baumbach. Comprehensive cluster analysis with Transitivity Clustering. Nature Protocols, 6(3) 285-295 (2011)

[16] G. A. Donnan, M. Fisher, M. Macleod, S. M. Davis. Stroke. *Lancet* 371. 9624. 1612-1623 (2008)

[17] T. Mather, V. Oganessyan, P. Hof *et al.* The 2.8 A crystal structure of Gladomainless activated protein C. *EMBO J.* 15:6822-6831(1996)

Supplementary Data

Traits/Disease	**Newly Found SNPs**
Alcoholism (alcohol dependence factor score)	rs2548145 rs3930234 rs4293630 rs933769
Atrial fibrillation	rs10033464
Atrial fibrillation/atrial flutter	rs13376333 rs7193343 rs958546 rs2106261 rs10501920 rs6843082 rs17042171 rs17375901 rs13038095 rs4776472
Cleft lip	rs9574565 rs7590268 rs17085106 rs227731 rs1258763 rs642961 rs7078160
Conduct disorder (case status)	rs6750486 rs7762160 rs1256531 rs12302829

	rs16891867
	rs4434872
	rs6031252
	rs8179116
	rs2122554
	rs3136202
	rs4792394
Conduct disorder (symptom count)	rs2184898
	rs17007017
	rs2720508
	rs1550057
	rs10776612
	rs7581919
	rs13398848
	rs1861046
	rs1861050
Ischemic stroke	rs13376333
	rs7193343
	rs958546
	rs2106261
	rs10501920
	rs6843082
	rs17042171
	rs17375901
	rs10033464
	rs13038095
	rs4776472
Nonsyndromic cleft lip with or without cleft palate	rs10863790
Permanent tooth development	rs9674544
	rs1956529
	rs6504340
	rs10506525
	rs8079702
	rs4844096
	rs5936487

	rs2817937 rs9386463 rs6435957
Plasma coagulation factors	rs17145713 rs1158867 rs6120849
Plasma levels of Protein C	rs9390459
Primary tooth development (number of teeth)	rs2281845 rs4491709 rs7924176 rs5936487 rs9386463
Primary tooth development (time to first tooth eruption)	rs2281845 rs1956529 rs6504340 rs4491709 rs7924176 rs4844096 rs2817937 rs6435957
Vitamin D insufficiency	rs1993116 rs2060793 rs3829251
Vitamin D levels	rs12785878 rs10741657

Supplement Table 1. Newly found traits/diseases, SNPs associations.