Efficient Large-scale Bicluster Editing

Peng Sun[1,2], Jan Baumbach[1,3] and Jiong Guo[2]
[1]Max Planck Institute for Informatics. Campus E1 4, 66123 Saarbrücken, Germany
[2]Cluster of Excellence for Multimodel Computing and Interaction.
Campus E1 7, Saarland University, 66123 Saarbrücken, Germany
[3]Institute for Mathematics and Computer Science,
University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
psun@mpi-inf.mpg.de

Abstract: The explosion of the biological data has dramatically reformed today's biological research. The need to integrate and analyze high-dimensional biological data on a large scale is driving the development of novel bioinformatics approaches. Biclustering, also known as *simultaneous clustering* or *co-clustering*, has been successfully utilized to discover local patterns in gene expression data and similar biomedical data types. Here, we contribute a new approach: *Bi-Force*. It is based on the weighted bicluster editing model, to perform biclustering on arbitrary sets of biological entities, given any kind of similarity function. We first evaluated the power of Bi-Force to solve dedicated bicluster editing problems by comparing Bi-Force with two existing algorithms in the BiCluE software package. We then followed a biclustering evaluation protocol from a recent review paper from Eren *et al.* and compared Bi-Force against eight existing tools: FABIA, QUBIC, Cheng and Church, Plaid, Bimax, Spectral, xMOTIFS and ISA. To this end, a suite of synthetic data sets as well as nine large gene expression data sets from Gene Expression Omnibus were analyzed. All resulting biclusters were subsequently investigated by Gene Ontology enrichment analysis to evaluate their biological relevance. The distinct theoretical foundation of Bi-Force (bicluster editing) is more powerful than strict biclustering. We thus outperformed existing tools with Bi-Force at least when following the evaluation protocols from Eren *et al.*. Bi-Force is implemented in Java and integrated into the open source software package of BiCluE. The software as well as all used data sets are publicly available at *http://biclue.mpi-inf.mpg.de*.

1 Introduction

Clustering is commonly accepted as a powerful approach to explore gene expression data sets [MAN+11]. Given a pairwise similarity function transformed into a similarity matrix, clustering algorithms seek to partition the data items into a list of disjoint groups, such that the similarities within each group are maximized and those between different groups are minimized. Traditional clustering approaches cluster only rows or columns in one run, which is not always beneficial [TSS02]. In contrast, biclustering allows to *simultaneously* partition both, rows and columns. If we are given, for instance, gene expression data sets for different cellular conditions, biclustering is more powerful in capturing biologically meaningful subsets of condition-specific genes. The major reason is that the expression

of gene subsets may correlated only under some conditions while being independent under other conditions. Biclustering approaches are generally capable of discovering such local patterns. They have proven particularly useful in various types of gene expression data analyses [GMO09] but should also work on other omics data sets, proteomics or metabolomics.

The first such "*biclustering*" tool was developed by Cheng and Church and applied to gene expression data [CC00]. Since then, many other biclustering tools have been reported (e.g. [MO04, BPP08, FBP10, BBP$^+$06, CLSL07]) and been suggested for applications to various biomedical problems [SGB12, HPC$^+$10]. Biclustering tools became increasingly popular due to their ability to simultaneously cluster biological data from different sources in order to discover local bi-correlations patterns. Several systematic comparisons have been published, using various measurements to evaluate a number of prevailing biclustering tools on both synthetic and real-world data sets [TBK05, PBZ$^+$06, EDKÇ13].

Here, we present a software implementing a novel heuristic algorithm that efficiently solves the biclustering problem: Bi-Force [SSR$^+$14]. It comes as an extension of the BiCluE software package. The Bi-Force extension is dedicated to solve the large-scale problem instances that we face in nowadays bioinformatics more and more frequently. Bi-Force was compared to eight existing biclustering software implementations on artificial and real biological data sets.

2 METHODS

The main methodological contribution of this paper is an algorithm that heuristically solves the weighted bicluster editing problem. Bi-Force is motivated by the well-known physics-inspired graph layout algorithm of Fruchterman and Reingold [FR91]. It mainly seeks to arrange all nodes of a graph in a two-dimensional plane such that "similar" nodes are located more close to each other than to others. Bi-Force, afterwards, assigned the nodes from 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.

2.1 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. Bi-Force computes pairwisely the "physical forces" between two nodes, i.e., the magnitudes that similar nodes attract each other, dragging them closer while dissimilar nodes repulse each other, pushing them farther away. The whole algorithm starts with an initial layout where nodes are "almost" evenly located on a two-dimensional circle with randomly permuted order. The radius R of the circle is a parameter of Bi-Force. The

strength of attracting/repulsing force depends 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 re-arrangement is performed in an iterative manner. In each round, the movement of each node is the cumulative effect of the attractions and repulsions from 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/repulsing 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/repulsing 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, they attract each other and if otherwise, they repulse. 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_0 affects the density/granularity of the bicluster editing model: the smaller t_0 is, the fewer biclusters there are and the larger of 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. This means in a certain iteration i, the movement magnitude cannot exceed the current cooling parameter M_i. Cooling parameter starts with an initial value M_0 as a parameter in Bi-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.

2.2 Bicluster partitioning

Based on the coordinates of the nodes obtained in the previous stage, we partition the graph into disjoint biclusters using two different geometric clustering methods: single-linkage clustering (SLC) and k-means. Both, SLC and k-means are standard methods in computational cluster analysis [WRR+11]. 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: $\delta = 0...M_0 + R$ in steps of σ, k-means: $k = 2...|V|/3$). For each clustering result we compute the necessary editing costs to create this solution. Finally, we keep the solution that has minimal editing costs before we proceed to post-processing.

2.3 Post-processing

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

To reduce the number of redundant biclusters, particularly the singletons, we try to merge biclusters. First, all the biclusters are ordered by size in an ascending order. Let $B = (b_1, b_2, ..., b_l)$ be the l ordered biclusters, where $|b_i| \leq |b_j|$, for all $i \leq j$. For all pairs of biclusters 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 biclusters 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 [KPJ04] 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 emerge 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 vertex move is beneficial.

The details of parameter training of Bi-Force could be found in [SSR+14].

2.4 Analysis

The worst-case running time of Bi-Force is dependent on the three steps mentioned above. Let $n = |U| + |V|$ for an input graph $G = (U, V, E)$. In the "layout generation" step, where Bi-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. The k-means problem is by its nature NP-hard [ADHP09, MNV09]. However, we limited the maximal number of iterations in k-means to be 200 and thus it finishes in $O(200 \cdot n)$ time. 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 Bi-Force did not reach this limit and we observed only small numbers of iterations before it terminated.

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

3 RESULTS

We applied all nine algorithms including Bi-Force and eight other prevalent biclustering tools mentioned above to real-world biological data: gene expression microarray data from the GEO database. Their performance was evaluated by means of GO Term Enrichment.

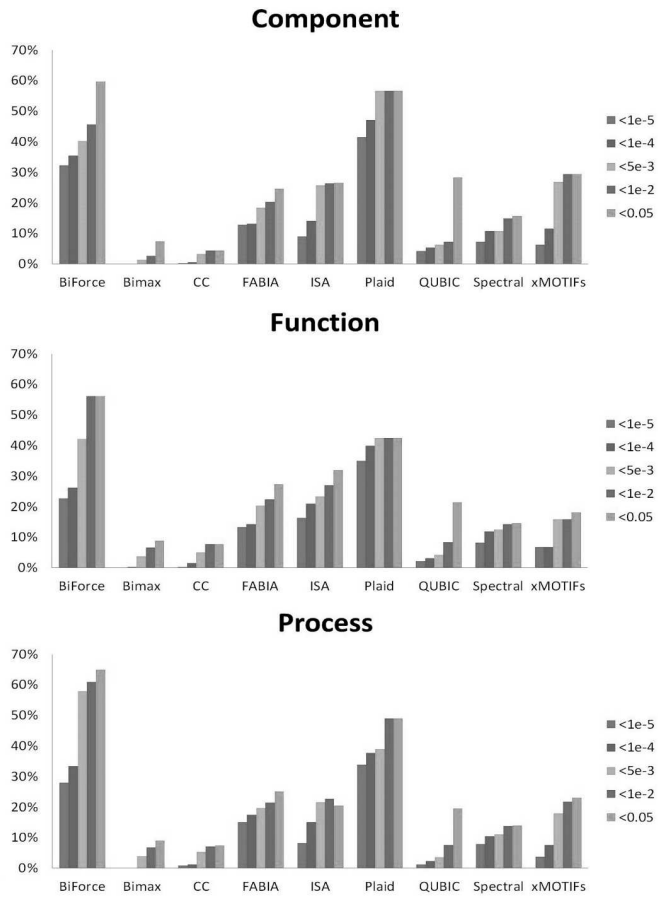

Figure 1: Proportions of GO-enriched biclusters for different biclustering tools on five significance levels.

Figure 1 gives the proportions for different significance levels of the biclusters found by all algorithms.

4 SUMMARY

We have presented Bi-Force and demonstrated its flexibility by applying it to biclustering, a restricted version of bicluster editing with many applications in gene expression data analysis. We compared it to eight existing tools by following an established evaluation protocol from Eren *et. al.*'s review paper. We show that Bi-Force outperformed the existing tools on synthetic data sets and on real-world gene expression data. Last but not the least,

we wish to emphasize that Bi-Force has the ability to perform simultaneous clustering of arbitrary multiple data sets. It is not restricted to gene expression scenarios. Instead any types of biological data that can be modeled as bipartite graph can be partitioned by using Bi-Force. It is now part of the BiCluE software package and publicly available at *http://biclue.mpi-inf.mpg.de*.

References

[ADHP09] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[BBP+06] Simon Barkow, Stefan Bleuler, Amela Prelić, Philip Zimmermann, and Eckart Zitzler. BicAT: a biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, 2006.

[BPP08] Stanislav Busygin, Oleg Prokopyev, and Panos M Pardalos. Biclustering in data mining. *Computers & Operations Research*, 35(9):2964–2987, 2008.

[CC00] Y. Cheng and G. M. Church. Biclustering of expression data. *Proc Int Conf Intell Syst Mol Biol*, 8:93–103, 2000.

[CLSL07] Kin-On Cheng, Ngai-Fong Law, Wan-Chi Siu, and TH Lau. BiVisu: software tool for bicluster detection and visualization. *Bioinformatics*, 23(17):2342–2344, 2007.

[EDKÇ13] 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.

[FBP10] Neng Fan, Nikita Boyko, and Panos M Pardalos. Recent advances of data biclustering with application in computational neuroscience. In *Computational Neuroscience*, pages 85–112. Springer, 2010.

[FR91] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[GMO09] Joana P Gonçalves, Sara C Madeira, and Arlindo L Oliveira. BiGGEsTS: integrated environment for biclustering analysis of time series gene expression data. *BMC Research Notes*, 2(1):124, 2009.

[HPC+10] Rave Harpaz, Hector Perez, Herbert S Chase, Raul Rabadan, George Hripcsak, and Carol Friedman. Biclustering of adverse drug events in the FDA's spontaneous reporting system. *Clinical Pharmacology & Therapeutics*, 89(2):243–250, 2010.

[KPJ04] Andrew D King, N Pržulj, and Igor Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, 2004.

[MAN+11] 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.

[MNV09] 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.

[MO04] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans Comput Biol Bioinform*, 1(1):24–45, 2004.

[PBZ+06] 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.

[SGB12] 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.

[SSR+14] 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.

[TBK05] 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.

[TSS02] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18 Suppl 1:S136–144, 2002.

[WRR+11] 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.