Storytelling and Clustering for Cellular Signaling Pathways

M. Shahriar Hossain, Monika Akbar, and Nicholas F. Polys
Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, USA.

Abstract - *In this paper, we describe our recent work on graph mining as applied to the cellular signaling pathways in the Signal Transduction Knowledge Environment (STKE) [1]. We present new algorithms and a graphical tool that can help biologists discover relationships between pathways by looking at structural overlaps within the database. We address the problem of determining pathway relationships by two data mining approaches: clustering and storytelling. In the first approach, our tool brings similar pathways to the same cluster and in the second, our tool determines intermediate overlapping pathways that can lead biologists to new hypotheses and experiments regarding relationships between the pathways. We formulate the problem of discovering pathway relationships as a subgraph discovery problem and propose a new technique called Subgraph-Extension Generation (SEG) that outperforms the traditional Frequent Subgraph Discovery (FSG) approach [2] by magnitudes. Our developed tool also provides an interface to compare these two approaches with a variety of similarity measures and clustering techniques as well as in terms of computational performance measures including runtime and memory consumption.*

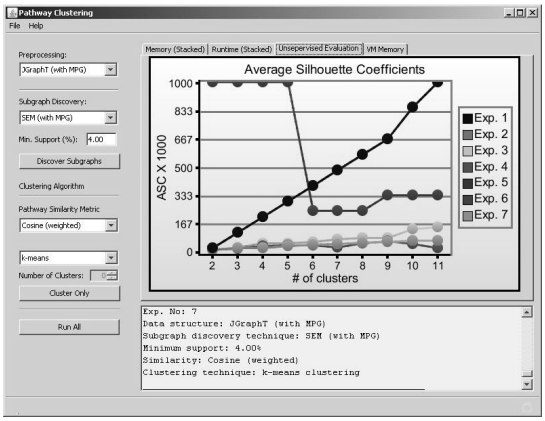
Keywords: Apriori, Cellular Signaling Pathway, Clustering, Storytelling, Subgraph-Extension Generation, FSG.

1 Introduction

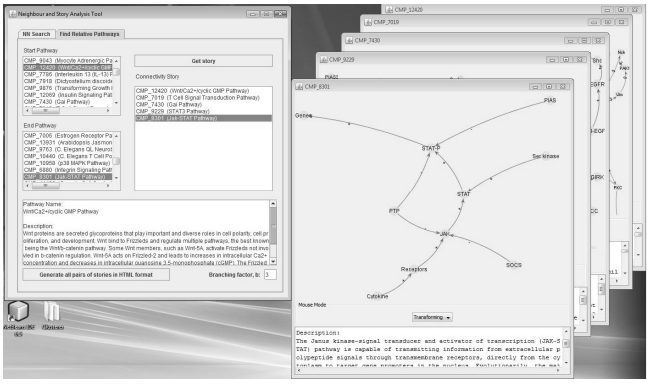
In this work we examine the algorithmic and computational costs for discovering relationships between a set of graphs. Understanding these costs and benefits is a crucial step to building visual analytic applications that perform interactively and in real-time. In order to illustrate our techniques, we apply and evaluate them using cellular signaling pathways organized as connection maps from the STKE dataset [1]. These pathways are essentially relationships between bio-molecular components such as proteins that transform cellular 'signals' to appropriate biological responses. Our observation is that a relation between two components of a signal can appear in more than one pathway that might aid the biologists to identify a new phenomenon.

A cellular signaling pathway contains a set of molecules interacting with each other through signals and conveying information, generally from the outside of the cell to inside [3]. The Signal Transduction Knowledge Environment (STKE) dataset covers signal transduction in biology, allowing a study of how cells interact with each other through chemical signals. Scientists all over the world documented different cell signaling pathways over time. Still, there can be some undiscovered relationships between various pathway components due to the lack of capable tools to mine and analyze the existing database.

The ultimate goal of our project is to build a tool that can discover relationships between pathways using graph mining approaches. The resultant pathway relationships could then help biologists to analyze the biochemistry and discover new relationships between them. In this work, we examine different algorithms to mine for frequent or common subgraphs among the signaling pathways. These frequent subgraphs are then used to calculate the similarities between every pair of pathways. We use the discovered subgraphs to cluster pathways or to discover a 'story', or connection,



A screenshot of the interface (clusters' unsupervised evaluation)



A screenshot of the storytelling interface.

Figure 1. Visual analytic interface of the developed tool.

between a pair of pathways. We have developed an interactive tool by which users can control the parameters for different phases of the pipeline. Additionally, we provide a runtime analysis for each of the algorithms implemented. Figure 1 shows two screenshots illustrating the interactive tool and the interface for storytelling.

In this work, we propose a graph-based storytelling approach that is similar to the text-based storytelling described by Kumar *et al.* [4]. The graph-based storytelling is more robust than the text-based storytelling approach considering the fact that texts are sometimes misleading and can generate meaningless stories. We insure that subsequent pathways in a generated story have overlapping signals between them- which text-based storytelling algorithms do not guarantee. As a result, chances that our algorithm generates misleading or meaningless stories are lower than the text-based storytelling.

The rest of the paper is organized as follows. Section 2 describes some of the related works. We describe the overall design in Section 3. Some illustrative experimental results are described in Section 4. We conclude this paper in Section 5.

2 Literature review

There are some existing tools that help biologists to visualize and analyze signaling pathways. PathCase [5] presents a way to visualize signaling pathways as nested graphs and employs four abstraction levels to counter for the visual complexity of signaling pathways. Xu *et al.* [6] design a model for the WNT signaling pathway, a gene regulation route of living cells of various organisms. Schreiber [7] presents a different approach using a constraint graph drawing algorithm for visually comparing metabolic pathways of different species. In this system, similar parts of the similar pathways of different species are placed side by side thereby making it easier to identify similarities and dissimilarities between the pathways. However, none of these tools provides any automatic mechanism to discover frequent subgraphs from the pathways with the aim of graph-similarity-based discovery of pathway relations. We aim to discover relations between pathways using clustering and storytelling algorithms. Our proposed approaches are automated and depend on discovered frequent subgraphs offering a graph-similarity-based relation discovery tool.

There has been some previous research related to clustering pathways. For example, ASK-GraphView [8] is a large scale graph visualization tool that is used for navigating large graphs. It provides a way to cluster the graphs using structural clustering for visualizing the graphs in a more compact form. We aim to find clusters automatically using discovered knowledge on the appearance of frequent structures in the pathways. Miyake *et al.* [9] present a comparison technique based on clustering between pathways. It introduces a scoring system to measure the similarity between pathways. Although [9] conveys significant importance to their corresponding biological datasets, it does not properly fit to the STKE dataset of our work. Moreover, their generated clusters are not evaluated using standard measures of clusters' validity.

Our approach depends on graphs for modeling the STKE dataset and takes advantage of graph-based data mining techniques. There are some well-known subgraph discovery techniques like FSG (Frequent Subgraph Discovery) [2], gSpan (graph-based Substructure pattern mining) [10], DSPM (Diagonally Subgraph Pattern Mining) [11], and SUBDUE [12]. Most of these systems have been tested on real and artificial datasets of chemical compounds. In our work, we discuss a system that can perform frequent subgraph discovery on large pathway-graphs. We propose a novel Subgraph-Extension Generation (SEG) mechanism that significantly outperforms the original FSG [2] approach.

To the best of our knowledge, we introduce the first automated graph-based technique to find relationships between pathways. We also propose the use of Master Pathway Graph (MPG) that helps our Subgraph-Extension Generation (SEG) mechanism to discover subgraphs in an efficient manner. We offer a fully automated system that utilizes enhanced subgraph discovery techniques and provides interactive feedback as to their effectiveness and performance.

3 Design

In this section, we give an overview of our proposed system. Figure 2 shows the design pipeline. It contains five major modules: (1) Preprocessor, (2) Frequent Subgraph Discovery Module, (3) Clustering Module, (4) Nearest Neighbours (NN) Module and (5) Storytelling Module.

(1) Preprocessor: This module takes inputs directly from the STKE dataset and converts each pathway to a graph object. The aim of this module is to provide an efficient data structure to store the pathways effectively for the subgraph discovery process. Additionally, it generates the Master Pathway Graph (*see* Definition 1) which assists Subgraph-Extension Generator (SEG) used by the frequent subgraph discovery module.

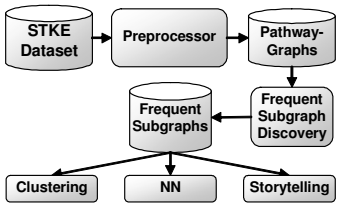


Figure 2. Design pipeline.

Definition 1. A Master Pathway Graph (MPG) is a graph that contains all the relations (edges) of all the pathways but contains them only once. Example: let P_1, P_2, P_3 and P_4 be the pathways in the dataset. If their corresponding edge-sets are defined by $P_1(e_1, e_2, e_5, e_6, e_7)$, $P_2(e_3, e_4, e_5, e_8)$, $P_3(e_4, e_5, e_6, e_7)$ and $P_4(e_3, e_4, e_5, e_6, e_8)$, then the edge-set of their MPG would be $MPG(e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8)$.

(2) Frequent Subgraph Discovery Module: This module uses our SEG approach to discover frequent subgraphs from

the pathways. The developed tool also provides the traditional FSG approach for comparison. The details of the FSG and the SEG approaches are described in section 3.1.1.

(3) Clustering Module: The clustering module generates clusters of pathways and provides comparison between two popular clustering techniques: Hierarchical Agglomerative Clustering (HAC [13]) and k-means [14]. It also provides an evaluation of the generated clusters with a graphical interface using Average Silhouette Coefficient (ASC [15]) at different numbers of clusters.

(4) NN Module: Given an input pathway P and an integer n (where $n<N$ and N=total number of pathways), the NN module provides n number of most similar pathways in their descending order of similarity values calculated with respect to pathway P. We use cover tree [16], an indexing mechanism for efficient nearest neighbor search.

(5) Storytelling module: Given two pathways, the storytelling module finds a story (*see* Definition 2) between these two pathways. Users can visually investigate the graph-structures of the intermediate pathways of a story using our tool. We generate stories between all possible pairs of the pathways, some of which might reveal insights to the biologists in developing a new pathway. Consider the scenario where a biologist wants to uncover the relationship between a gene and a phenotype. The pathways related to this gene and the phenotypes are not enough to establish a logical relationship. Given two pathways (and the database of all known pathways), our tool can provide a set of possible relationships in the form of a story with visual representations that would help the biologist to analyze and discover new relationships.

Definition 2. A story from pathway P_1 to pathway P_z is a sequence of some intermediate pathways $P_2, P_3, ..., P_{z-1}$ such that similarity $sim(P_i, P_j)>0$, $1 \le i < z$, and $j=i+1$. In this work, we denote the length of a story by z, i.e., the number of pathways involved in the story. A story of length *2* does not have any intermediaries.

3.1 Implementation details

In the following subsections, we describe all the algorithms we have used to develop the tool.

3.1.1 Apriori paradigm

One of the baseline approaches to mining subgraphs from the pathways database is the Apriori paradigm [17], which was originally developed to solve the association rule mining problem of market basket datasets [18]. The aim of the original Apriori algorithm is to find frequent itemsets from a list of transactions. The algorithm concentrates on the corresponding supports of the items and itemsets. In our work, we replace transactions with pathways, items with edges and item-sets with subgraphs (i.e., sets of connected edges). The association rule mining problem of market basket data analysis has an analog in our research area as the problem of frequent subgraph discovery. In this subsection, we illustrate the internal mechanism of our Apriori paradigm and propose an enhancement to generate candidate subgraphs efficiently.

Table 1. Apriori algorithm.

Input:
D: a database of pathway-graphs
min_sup: the minimum support threshold
Output:
L: frequent subgraphs in D
Method:

(1) L_1= find_frequent_1-edge_subgraphs(D);
(2) for ($k=2$; $L_{k-1} \neq \Phi$; k++){
(3) C_k=apriori_gen(L_{k-1});
(4) for each pathway-graph $g \in D$ {
(5) $C_g = C_k \bigcap g$;
(6) for each candidate subgraph $s \in C_g$
(7) s.count++;
(8) }
(9) L_k={ $s \in C_k \mid s.count \geq min\_sup$}
(10) }
(11) return $L = \bigcup_k L_k$

Table 1 portrays the modified high-level algorithm for frequent subgraph discovery using the Apriori paradigm. The `apriori_gen` procedure in the algorithm joins and prunes the subgraphs. In the join operation, a k-edge candidate subgraph is generated by combining two *(k-1)*-edge subgraphs of L_{k-1}. This k-edge subgraph becomes a member of L_k only if it passes the `min_sup` threshold. We use the FSG [2] approach to join two subgraphs to obtain a higher order subgraph. The details are given in section 3.1.1.1. We also propose a novel mechanism called Subgraph-Extension Generation (SEG) in section 3.1.1.2 which is more efficient than the FSG approach.

We define importance factor of a subgraph, *sfipf* (short for subgraph frequency × inverse pathway frequency) of a subgraph s_i of pathway j as follows:

$sf_j = \frac{1}{n_j}$, $ipf_i = \frac{|D|}{|\{p_j : s_i \in p_j\}|}$ where n_j is the total number of subgraphs in pathway j and $|\{p_j : s_i \in p_j\}|$ indicates the number of pathways where subgraph s_i appears. $|D|$ is the total number of pathways in the dataset. We say, $sfipf_{i,j} = sf_j \times ipf_i$. We use an sfipf threshold *min_sfipf* in the `find_frequent_1-edge_subgraphs` procedure to pick up only the important edges for subgraph discovery.

An edge of a pathway of the STKE dataset is directed (upstream/downstream) and contains one of the four signaling activities: stimulatory(+), inhibitory(-), undefined effect(?) or neutral(0). The directions and edge attributes are strictly considered during the comparison between two subgraphs in our implementation. For example, consider that there is a stimulatory relation between two vertices v_1 and v_2 in a subgraph X. Consider we have another subgraph Y that looks same as X, but the edge between v_1 and v_2 of Y is inhibitory. Our implementation considers that these two

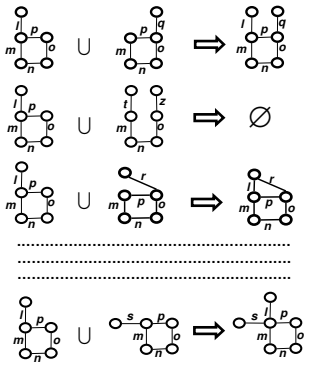


Figure 3. Attempts to combine *lmnop* with other *5*-edge subgraphs of (L_5).

subgraphs are different differing in one edge. Therefore, the implementation takes the edge attributes into consideration as well as the direction of the edges.

3.1.1.1 FSG

In the FSG approach, we generate a *(k+1)*-edge candidate subgraph by combining two *k*-edge subgraphs where these two *k*-edge subgraphs have a common core subgraph [2] of *(k-1)*-edges. This approach requires time-consuming comparisons between core subgraphs while generating a higher order subgraph. Although this approach is very fast for small graphs, it becomes inefficient for large graphs due to large number of blind attempts to combine *k*-edge subgraphs to generate *(k+1)*-edge subgraphs.

Consider an Apriori's iteration in which we have a total of 21 *5*-edge subgraphs in the candidate list L_5. We try to generate *6*-edge subgraphs from this list. Consider the situation of generating candidates using one *5*-edge subgraph (e.g., the subgraph *lmnop* of Figure 3) of L_5. The original FSG approach tries to combine all remaining 20 other subgraphs with *lmnop* but succeeds, let us assume, only in three cases. Figure 3 illustrates that *lmnop* is successfully combined with only *mnopq*, *mnopr* and *mnops*. All 17 other attempts to generate a *6*-edge subgraph with *lmnop* fail because the *4*-edge core-subgraphs, analyzed in this case, do not match. Figure 3 shows the attempts to generate good candidates for just one subgraph (*lmnop*). For all the subgraphs in L_5, there would be a total of *21×20* blind attempts to generate *6*-edge subgraphs. Some of these attempts would succeed, but most would fail to generate acceptable *6*-edge candidates. This algorithm cannot avoid comparing a large number of core subgraphs to generate all candidates. We have reduced the number of comparisons by a significant degree with our Subgraph-Extension Generation approach. The technique is described in the following sub-section.

3.1.1.2 Subgraph Extension Generation (SEG)

Rather than trying a brute-force strategy of comparing all possible combinations (e.g., FSG), we use the master pathway graph (MPG) as the source of background knowledge to entirely eliminate the unsuccessful attempts at generating *(k+1)*-edge candidate subgraphs from *k*-edge

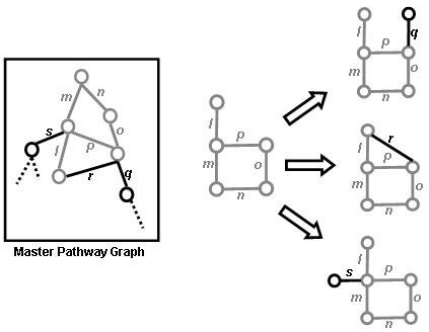


Figure 4. *6*-edge Subgraph-Extension Generation for the *5*-edge subgraph *lmnop*.

subgraphs. We maintain a neighboring-edges' list for each *k*-edge subgraph and generate candidates for frequent higher order subgraphs by taking edges only from this list.

Figure 4 shows the Subgraph-Extension Generation mechanism for subgraph *lmnop*, which can be compared with the FSG approach of Figure 3. The gray edges of Figure 4 are the edges of the *5*-edge subgraph which we want to extend to generate *6*-edge candidates. The black lines indicate the neighboring edges which extend the *5*-edge gray subgraph maintained in our MPG. The same instance is used for both Figure 3 and Figure 4 for easy comparison. The neighboring-edges' list of *lmnop* contains edges *{q, r, s}*. Unlike Figure 3, the example presented in Figure 4 uses the Subgraph-Extension Generation technique and does not try to blindly generate higher order subgraphs 20 times. Rather, it proceeds only three times, using the constraints coming from knowledge about the neighboring edges of *lmnop* in the MPG. It results in only three attempts to generate higher-order candidate subgraphs. None of these attempts fails at step 3 of the Apriori algorithm (Table 1) because the mechanism depends on the physical evidence of possible extension. Therefore, SEG offers a novel knowledge-based mechanism that eliminates unnecessary attempts to combine subgraphs. All the generated subgraphs that pass the *min_sup* threshold are stored in a subgraph-pathway matrix which is used in the clustering or storytelling phase later.

3.1.2 Pathway clustering and cluster's evaluation

We use commonly recognized methods to group pathways: Hierarchical Agglomerative Clustering (HAC [13]) and k-means [14] clustering. The tool provides evaluation in a broad range of generated clusters. Our tool also provides a selection of different types of similarity measures for the resulting clusters: Cosine, *sfipf* weighted Cosine, Jaccard, Dice, Overlap and Matching coefficients.

To evaluate the clustering results, we used unsupervised measure of cluster validity: Average Silhouette Coefficient (ASC) [15]. An overall measure of goodness of a clustering can be obtained by computing the average silhouette coefficient of all data points [19].

3.1.3 Storytelling

We use bidirectional search to find a story between two pathways P_1 and P_z. In a bidirectional search algorithm two

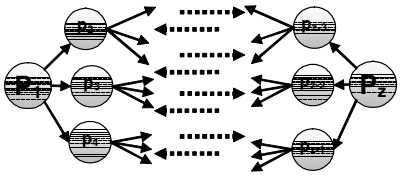


Figure 5. Bidirectional search for storytelling.

simultaneous searches proceed in two directions: one forward from the start pathway (P_1), and one backward from the goal (P_z). The search stops when the two searches meet in the middle. A path from P_1 to P_z forms a story. Figure 5 portrays a sketch of the bidirectional search used in our tool. The search proceeds from one pathway P_i to its *n* most nearest pathways. The length of a story and the search time depend on the branching factor *n*. We use bidirectional search for every pair of pathways to generate all possible stories.

4 Experimental results

To assess the power and performance of these algorithms we conducted an experiment on the STKE dataset. All the results of this paper are generated in a machine with Intel quad core processor and 8GB memory. The tool is implemented in Java and the JVM was running under Windows Vista 64-bit platform during all the experiments.

4.1 Dataset

The STKE dataset [1] used in this work contained 50 pathways. Figure 6 shows that larger pathways are scarce in the dataset compared to smaller pathways. The largest pathway has 101 edges and the smallest pathway has 2 edges. Since there are a total of 1376 unique relations in the dataset, our MPG contains a total of 1376 edges. Figure 7 provides the number of edges and pathways left as functions of *sfipf* threshold *min_sfipf*. It shows that large *min_sfipf* results in filtration of large amount of edges as well as pathways.

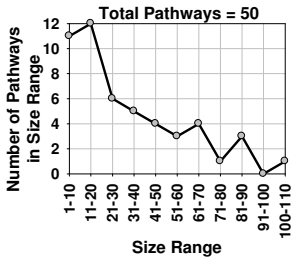


Figure 6. Number of pathways as function of size-range.

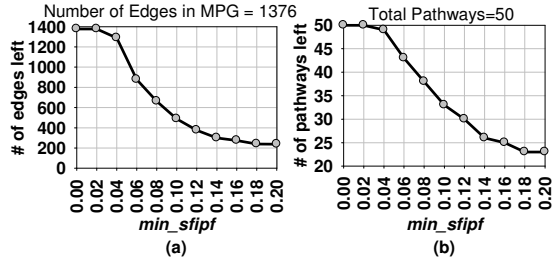


Figure 7. (a) Number of edges left in MPG as function of *min_sfipf*, (b) Number of pathways left as function of *min_sfipf*.

4.2 Subgraph discovery

The number of generated subgraphs in our subgraph discovery process depends on the *min_sfipf* and *min_sup* thresholds. Figure 8 shows experimental results with varying *min_sfipf* and *min_sup*. It shows that the subgraph discovery module discovers higher amount of frequent subgraphs with lower *min_sfipf* and *min_sup* thresholds. The clustering and the storytelling parts require large amount of subgraphs for better results. This means, we can find better relationships and clusters among pathways with lower *min_sfipf* and *min_sup* values. On the other hand, lower *min_sfipf* and *min_sup* values results in longer runtime in the subgraph discovery process. Additionally, lower thresholds require more memory due to large amount of generated candidate subgraphs during each iteration of the Apriori paradigm. The tradeoff depends on the machine where our tool is running.

A performance comparison between the FSG and our SEG approach is given in Figure 9. Due to the significant speed of SEG, the gray line drawn for it appears to be linear and flat in comparison to the black line of the FSG approach, although the actual behavior of SEG is not really linear. The curve maintains its hat-like shape, typical of the Apriori approach, but due to the scale necessary to show the FSG results it is not clearly visible in Figure 9(a). We changed the scale in Figure 9(b) to show the actual behavior of SEG.

Figure 10 depicts the reason why the SEG approach is more efficient than the FSG approach. The figure compares numbers of attempts to generate *k*-edge subgraphs by the FSG and SEG approaches. The SEG approach outperforms the FSG approach by a high magnitude due to the lower number of attempts used to generate higher order subgraphs by avoiding blind attempts. Table 2 shows that in every case except for *k=21*, SEG saved a huge percentage of blind attempts generated by the FSG approach. The SEG approach saved 90.26% of the attempts while generating *16*-edge subgraphs from frequent *15*-edge subgraphs. Since *15*-edge subgraphs are the most frequent ones (1117 *15*-edge subgraphs were discovered), obviously the number of attempts to construct *16*-edge subgraphs from *15*-edge subgraphs reaches the maximum for the FSG approach in Figure 10. Also, Figure 9 shows that both the curves reach their peaks near *16*-edge subgraphs. Overall attempts saved

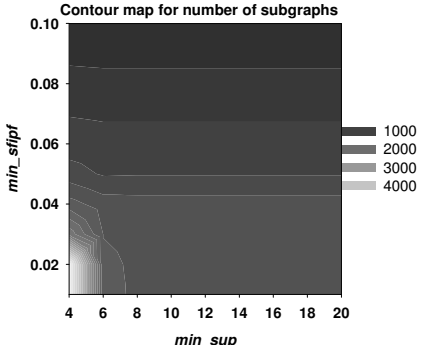


Figure 8. Contour map illustrating number of generated subgraphs.

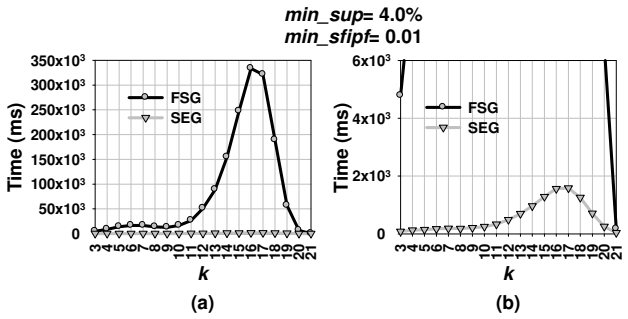


Figure 9. Runtime comparison between FSG and SEG.

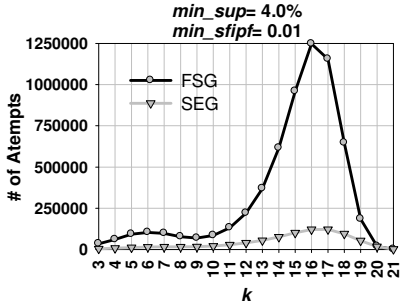


Figure 10. Numbers of attempts to discover *k*-edge subgraphs by FSG and SEG.

Table 2. Reduction of blind attempts by the SEG.

k	# of discovered *k*-edge Subgraphs	Time Saved (%)	Attempts Saved (%)
2	186	99.83	98.98
3	246	98.33	86.15
4	305	98.57	86.38
5	323	98.95	86.91
6	313	98.96	85.64
7	279	98.88	83.25
8	263	98.67	78.91
9	292	98.38	74.76
10	364	98.58	74.75
11	470	98.76	78.08
12	608	99.04	81.84
13	785	99.22	85.02
14	980	99.38	87.63
15	***1117***	99.48	89.48
16	1075	99.53	90.26
17	804	99.51	89.4
18	430	99.34	85.22
19	141	98.76	71.22
20	20	96.15	9.19
21	1	75.74	-574.47

by the SEG approach in this experiment was 89.52% which saves 99.39% runtime used by the FSG approach.

4.3 Clustering

We used HAC and k-means to automatically cluster the pathways. Figure 11 provides an unsupervised evaluations at different numbers of clusters both for HAC and k-means using five different similarity measures. The plot shows that HAC provides higher ASC than the k-means clustering using any similarity measure for the STKE dataset.

The clusters' accuracy also varies depending on the *min_sup* and *min_sfipf* thresholds. Figure 12(a) and (b) provides two ASC contour maps at 10 clusters for HAC and

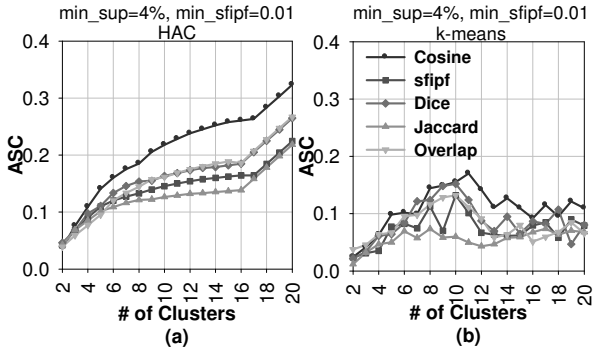


Figure 11. ASC at different numbers of clusters using (a) HAC and (b) k-means.

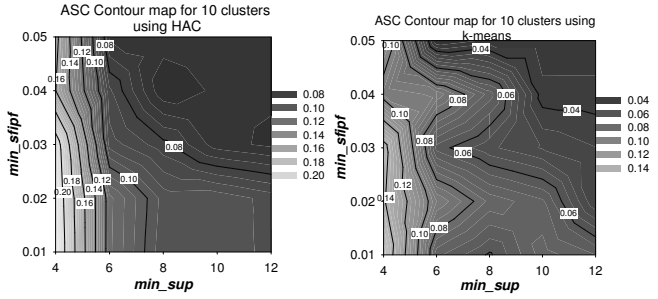


Figure 12. Contour map for ASC at 10 clusters (a) Hierarchical Agglomerative (b) k-means Clustering.

k-means respectively. Figure 12 (a) shows that HAC produces best results in the bottom left corner of the plot with ASC=0.20. Similarly, Figure 12 (b) shows that k-means generates its best clusters in the region with ASC=0.14 of the plot where *min_sup* and *min_sfipf* thresholds are small. Therefore, both HAC and k-means generate better clusters with lower *min_sup* and lower *min_sfips* thresholds. Earlier in Figure 8, we showed that lower *min_sup* and *min_sfipf* thresholds results in higher amount of subgraphs. Since, lower *min_sup* and *min_sfipf* results in larger amount of subgraphs, any clustering algorithm can better separate the groups which results in high quality clusters. Figure 12 also shows that HAC performs better than the k-means clustering since HAC has higher ASC (0.20) than the ASC (0.14) of k-means in corresponding bottom-left corners.

4.4 Storytelling

We attempted to generate stories for all possible pairs of pathways. But, not all pairs have stories. Some pairs do not have any relationship between them and with other pathways. Number of stories and story lengths vary with branching factor *b* used in the bidirectional search algorithm for storytelling. Figure 13 (a) shows that a small branching factor has the tendency to generate few stories, and large branching factor generally generates lots of stories. Figure 13 (b) also shows that total number of generated stories is larger with high branching factor. Figure 13 (c) depicts that large branching factor provides longer stories. Finally, Figure 13 (d) shows time to generate all possible stories using different branching factors. Although the bidirectional storytelling

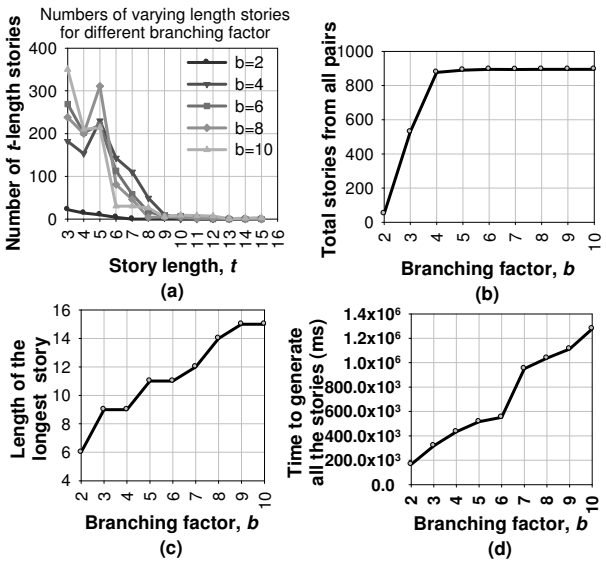


Figure 13. (a) Length of stories as function of story length, (b) Total stories generated from all pairs as function of *b*, (c) Longest story found as function of *b*, (d) Time to generate all possible stories as function of *b*.

algorithm generates longer stories with higher branching factor, the process becomes slower.

5 Conclusions

In this work, we developed a tool that would help biologists to find relationships between cellular signaling pathways. We have used enhanced algorithms in different phases of the work and given comparisons with traditional data mining approaches. In this work, we propose our novel Subgraph-Extension Generation approach that outperforms the FSG approach by high magnitude. We addressed the problem of determining pathway relationships by clustering and storytelling. We decomposed the problem of pathway-relation search into frequent subgraph discovery problem and provided a graph-based solution that depends on the structural overlaps between the pathways. The tool would help biologists as well as data mining researchers to analyze performance of different algorithms. It provides a visualization of the efficiency of different algorithms and evaluations of clusters. [20] contains all our source codes, necessary libraries, the dataset and samples of the generated stories. In future, we want to incorporate text mining features in our tool so that we can find relations between pathways using their descriptions. We could then compare the traditional text-based approach with our graph-based model.

6 References

[1] Science Signaling, The signal Transduction Knowledge Environment (STKE), "The Database of Cell Signaling", http://stke.sciencemag.org/cm/

[2] Kuramochi, M. and Karypis, G., "An efficient algorithm for discovering frequent subgraphs", IEEE Transactions on KDE, Vol. 16(9), September 2004, pp. 1038-1051.

[3] Breslin, T., Krogh, M., Peterson, C., and Troein, C., "Signal transduction pathway profiling of individual tumor samples", BMC Bioinformatics, June 29, 2005.

[4] Kumar, D., Ramakrishnan, N., Helm, R. F., and Potts, M., "Algorithms for Storytelling", IEEE Transactions on KDE, Vol. 20(6), June 2008, pp. 736-751.

[5] Ratprasartporn, N., Cakmak, A., and Ozsoyoglu, G., "On Data and Visualization Models for Signaling Pathways", 18th SSDBM, 2006, pp. 133-142.

[6] Xu, X., and Yu, Y., "Modeling and Verifying WNT Signaling Pathway", 3rd Intl. Conf. on ICNC. 2007, Vol. 2, pp. 319-323.

[7] Schreiber, F., "Comparison of metabolic pathways using constraint graph drawing", 1st Asia-Pacific bioinformatics Conf. on Bioinfo., Australia, Vol. 19, 2003, pp. 105-110.

[8] Abello, J., van Ham, F., and Krishnan, N., "ASK-GraphView: A Large Scale Graph Visualization System", IEEE Transactions on Visualization and Computer Graphics, Vol. 12(5), 2006, pp. 669-676.

[9] Miyake, S., Tohsato, A., Takenaka, Y., and Matsuda, H. "A clustering method for comparative analysis between genomes and pathways", 8th Intl. Conf. on Database Systems for Advanced Applications, March 2003, pp. 327-334.

[10] Yan, X., and Han, J. "gSpan: graph-based substructure pattern mining", IEEE ICDM, 2002, pp. 721-724.

[11] Moti, C., and Ehud, G. "Diagonally Subgraphs Pattern Mining", 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, 2004, pp. 51-58.

[12] Ketkar, N., Holder, L., Cook, D., Shah, R., and Coble, J. "Subdue: Compression-based Frequent Pattern Discovery in Graph Data", ACM KDD Workshop on Open-Source Data Mining, August 2005, pp. 71-76.

[13] Zhang, T., Ramakrishnan, R., and Livny, M., "BIRCH: An Efficient Data Clustering Method for Very Large Databases", ACM SIGMOD Intl. Conf. on Management of Data, Canada, 1996, pp. 103-114.

[14] Wagsta, K., Cardie, C., Rogers, S., and Schroedl, S., "Constrained K-means Clustering with Background Knowledge", ICML 2001, pp. 577-584.

[15] Lin, F., and Hsueh, C. M., "Knowledge map creation and maintenance for virtual communities of practice", Intl. Journal of Information Processing and Management, ACM, Vol. 42(2), 2006, pp. 551-568.

[16] Beygelzimer, A., Kakade, S., Langford, J., "Cover trees for nearest neighbor", ICML 2006, pp. 97-104.

[17] Agrawal, R., and Srikant, R. "Fast Algorithms for Mining Association Rules", Intl. Conf. on Very Large Data Bases, Santiago, Chile, September 1994, pp. 487-499.

[18] Agrawal, R., Mehta, M., Shafer, J., Srikant, R., Arning, A. and Bollinger, T. "The Quest Data Mining System", ACM KDD, USA, 1996, pp. 244-249.

[19] Tan, P. N., Steinbachm, M., and Kumar, V., "Introduction to Data Mining", Addison-Wesley, ISBN: 0321321367, April 2005, pp. 539-547.

[20] http://people.cs.vt.edu/amonika/infoviz/