CSV: Visualizing and Mining Cohesive Subgraphs

Nan Wang[†] Srinivasan Parthasarathy [§] Kian-Lee Tan [†] Anthony K. H. Tung [†] [*]
[†]School of Computing
National University of Singapore, Singapore
{wangnan, tankl, atung}@comp.nus.edu.sg

[§]Dept. of Computer Science and Engineering and Department of Biomedical Informatics
The Ohio State University
srini@cse.ohio-state.edu

ABSTRACT

Extracting dense sub-components from graphs efficiently is an important objective in a wide range of application domains ranging from social network analysis to biological network analysis, from the World Wide Web to stock market analysis. Motivated by this need recently we have seen several new algorithms to tackle this problem based on the (frequent) pattern mining paradigm. A limitation of most of these methods is that they are highly sensitive to parameter settings, rely on exhaustive enumeration with exponential time complexity, and often fail to help the users understand the underlying distribution of components embedded within the host graph.

In this article we propose an approximate algorithm, to mine and visualize cohesive subgraphs (dense sub components) within a large graph. The approach, refereed to as Cohesive Subgraph Visualization (CSV) relies on a novel mapping strategy that maps edges and nodes to a multi-dimensional space wherein dense areas in the mapped space correspond to cohesive subgraphs. The algorithm then walks through the dense regions in the mapped space to output a visual plot that effectively captures the overall dense sub-component distribution of the graph. Unlike extant algorithms with exponential complexity, CSV has a complexity of O($V^2 logV$) when fixing the parameter mapping dimension, where V corresponds to the number of vertices in the graph, although for many real datasets the performance is typically sub-quadratic.

We demonstrate the utility of CSV as a stand-alone tool for visual graph exploration and as a pre-filtering step to significantly scale up exact subgraph mining algorithms such as CLAN [33].

[*]Support in part by NUS ARF Grant R-252-000-277-112 as part of the research project entitled "GEMINI: Gene Expression Mining" (http://nusdm.comp.nus.edu.sg/).

1. INTRODUCTION

Technological advances have made the collection of vast amounts of data possible in many domains. Computational simulations, astronomical observations, medical imaging, and World Wide Web often involve very large amounts of data. Extracting useful information from such data, often concisely represented as a graph, is vital.

A graph is an intuitive abstraction that naturally captures data entities as well as the relationship among those entities. Data entities are represented as vertices in the graph and edges capture the binary relationships between data atoms. Although on-going research in different domains may create the need to capture non-binary relationships (similar cases occur in traditional DBMS, where non-binary relationships are resolved via table joins), we can still use several graphs to decompose those non-binary relationships into binary ones. Graph representations are indeed extremely flexible and can be used to model data from a number of domains.

Graphical representations are widely used in domains such as proteomics, bibliometric and social relationship analysis, and the World Wide Web. For example, in the domain of social relations, a social network can be represented as a graph whose vertices represent people and an edge connects two vertices if two corresponding people have certain relationships. The Digital Bibliography & Library Project (DBLP) network is an instance of a bibliographic network in the form of an academic publication community in computer science. DBLP records relations such as co-authorship and article-reference for further citation and referencing purposes.

Interconnections in these domains, and the size of the graph representations grow larger and more complex with time and with advances in technology. Accordingly, it becomes more difficult to abstract information from those graphs. Researchers have thus invented various graph mining algorithms that aim to discover interesting information(eg. patterns or subgraphs) efficiently and accurately.

Among classes of interesting subgraphs, highly-cohesive subgraphs have often proven to be useful in different domains. For example, the identification of cohesive subgraphs in PPI networks is important since such subgraphs may aid in functional annotation [17, 5]. In the domain of stock market analysis, the phenomenon of strong correlations among a small group of stocks is fairly common [7]. Analysts often make trading decisions of one stock based on a group of highly correlated stocks, which can be modeled as a cohesive subgraph.

Despite the range of applications discussed above, to the best of of our knowledge an effective tool for visualizing the distribution of cohesiveness within a graph does not exist. Such visualization is important as it allows users to gain important insights about the task on hand. A key challenge in achieving efficient visualization lies in the inherent difficulty of cohesive subgraphs detection. The extreme case of this problem is clique detection, which is known to be NP-hard and very difficult to even approximate[30]. Moreover, existing cohesiveness measures in graph theory such as k-connectivity are computationally prohibitive (e.g., the exact algorithm to decide subgraph vertex-connectivity is #P-complete [11]). This makes research on cohesive subgraph mining and visualization a challenging topic. In this paper, we propose an algorithm called CSV[1] to visualize and mine cohesive subgraphs. The main contributions of our work are summarized below:

1. We propose a novel algorithm called CSV which can compute an ordering on the vertices of graph G and create a density plot based on the ordering such that cohesive subgraphs within G can be visualized based on the plot.

2. As CSV needs to detect cliques within the graph which will result in exponential running time, we propose a method to speed up the algorithm by computing an upper bound on the size of cliques within the graph instead. Our method makes use of a novel mapping that transforms graph elements (vertices and edges) into high-dimensional points and preserves graph elements' connectivity relations. The transformation embodies the desirable property that existing spatial indices such as the R-tree can be applied to the transformed data for more efficient mining of cohesive subgraphs.

3. We evaluate our algorithm on real datasets drawn from stocks correlation networks and DBLP co- authorship networks. The results show that our algorithm is especially suitable for locating subgraphs quickly. We also experimentally prove the algorithm's effectiveness and efficiency by comparing it with other state-of-the-arts algorithms.

4. In addition to using CSV as a stand-alone tool for visual exploration of dense sub-components within large graphs we find that it can also be effectively used as a pre-filtering step to significantly speed up exact clique finding algorithms such as CLAN[33].

2. RELATED WORK

Mining graphs for interesting patterns has been a subject of much study recently [20, 37, 5, 33, 17, 35, 25, 22, 36]. Below we briefly describe the work that is most relevant to the current work.

One approach to mining interesting patterns can be achieved by graph partitioning along a minimum cut. Ng and colleagues[25], use spectral methods to interrogate large complex networks in order to find cohesive subgraphs. Since the eigenvectors of a graph depict its topological information, their approach relies on spectral graph partitioning using the second eigenvector of a graph's Laplacian (also known as the Fiedler vector). The algorithm recursively bisects the graph until the desirable number of partitions is achieved. By its nature, this algorithm aims to discover highly connected subgraphs inside a graph. However, the use of graph eigenvectors during partitioning incurs high computational cost. METIS [22] is an algorithm for graph partitioning, a related but different problem to ours. The algorithm comprises three stages: a coarsening step where tight groups of vertices are replaced with super nodes; a partitioning step where the resulting coarsened graph is partitioned; and a refining step where the original vertices are added back to the graph and the partitions are refined accordingly. The approach is quite scalable but METIS targets a different task – graph partitioning, and moreover tends to favor balanced partitions and does not always effectively identify dense subgraphs of interest[5].

By applying special encoding techniques, some solutions convert the dense graph mining problem into a traditional data mining algorithm, so that an Aprior-style algorithm[2], can be directly applied [21, 24, 34, 17]. Of these the most germane to the current work is CODENSE[17]. CODENSE mines frequent coherent cohesive subgraphs across a collection of massive graphs. A coherent subgraph is a graph where all edges exhibit correlated occurrences in the collection of graphs. The density of a graph is defined by the ratio of graph edge count in the graph to the same-sized, complete graph's edge count. CODENSE performs clustering on two meta-graphs summarizing the graphs to be mined. CODENSE can discover overlapping clusters. This is important in biological applications since under different conditions, one gene may serve different roles and be involved in different functional groups [4]. Aside from density, in [35], studies are also done on efficiently finding interesting subgraphs through combining connectivity constraints with column or row enumeration methods [26, 14, 12, 27, 13].

The above algorithms do not provide any visual cue for graph exploration. Osprey [9] is a system for visualization and analysis of complex protein interaction networks with no mining ability embedded. Visant [36] leverages standard graph theory methods to visually identify cohesive subgraphs within larger networks. However, it does not include any functionality for exploring the hierarchical structure nor for viewing the density/connectivity of the subgraph.

In summary, there are no existing algorithms or systems that can provide efficient density subgraph algorithms together with visualization capabilities. If we can derive an approach that can visualize mining results, users of a mining tool can identify graph structural features concretely. We next describe our algorithm, CSV, to provide such visual graph mining capabilities.

3. PRELIMINARIES AND PROBLEM DEFINITION

A graph is represented as $G = \{V, E\}$ where V is a set of distinct vertices $\{v_1, ..., v_{|V|}\}$ and E is a set of edges $\{e_1, ..., e_{|E|}\}$. A graph $G' = \{V', E'\}$ is a subgraph of G if $V' \subseteq V$, $E' \subseteq E$ and all end-vertices of edges in E' are in V'.

In this paper, we define the distance between two vertices v_i and v_j as the number of edges on the shortest path connecting v_i and v_j. To simplify discussion, we assume G is

[1]CSV stands for Cohesive Subgraph Visualization

a connected graph. If G consists of a set of unconnected components, the visualization technique in this paper can be applied to each component individually.

To determine the cohesiveness of a subgraph, two measurements are often used: density [1, 17] and connectivity [35].

DEFINITION 1. *Density of G', $\gamma(G')$*
The density of a graph $G' = \{V', E'\}$ is defined as:

$$\gamma(G') = \frac{2 * |E'|}{|V'| * (|V'| - 1)}$$

□

DEFINITION 2. *Connectivity of G',$\kappa(G')$*
The connectivity of a graph G' is defined as $\kappa(G') = k$, where k is the largest integer such that there exists no $k-1$ vertices whose removal disconnect the graph. We also call G' a k-connected graph. □

Intuitively, inside a k-connected graph, there are at least k independent paths from any vertex to any other vertex. The exact algorithm to decide vertex connectivity is not trivial([11] proves that this problem is #P-complete).

Although the above two measurements of cohesiveness are inherently different, both are however maximized when the graph is a clique. In this paper, we will mostly focus on finding this special type of highly cohesive subgraphs (i.e., cliques) due to space consideration.

Given a graph $G = \{V, E\}$, we want to compute an ordering of vertices in G and generate a density plot based on the ordering such that:

Problem 1: All cliques with size greater than k can be identified based on visualizing the density plot. □

While our focus in this paper will be on solving Problem 1, we will provide explanation during the course of discussion on how our solution to Problem 1 can in fact provide a solution to Problem 2 and 3 below:

Problem 2: All subgraphs with density greater than γ_{min} and graph size greater than $size_{min}$ can be identified based on visualizing the density plot. □

Problem 3: All subgraphs with connectivity greater than k can be identified based on visualizing the density plot. □

4. ALGORITHM CSV

Our CSV algorithm is an OPTICS [3] style plot which walks through the vertices of the graph based on two local density measures: **maximum participated clique size** and **maximum co-clique size**.

DEFINITION 3. *maximum participated clique size, $\zeta_{max}(v)$*
The maximum participated clique size of a vertex v, denoted as $\zeta_{max}(v)$ is the size of the biggest complete subgraph(i.e., a clique) in G that includes v. □

Intuitively, $\zeta_{max}(v)$ is analogue to the notion of spatial density in density-based clustering [15, 3]. Unlike spatial density, graph density must take into account the number of both vertices and edges. The definition of $\zeta_{max}(v)$ naturally takes this into account.

Algorithm 1: CSV
Input: Graph G={V,E}
Output: Density plot for visualizing cohesive subgraphs
Method:

```
1.  HEAP=$\emptyset$;
2.  FOR all $v \in V$
3.      v.$\eta_{mseen}$=0;
4.  WHILE $\exists v$ unvisited
5.      IF HEAP=$\emptyset$
6.          $v$=randomly selected unvisited vertex from $V$;
7.      ELSE
8.          $v$=next vertex on HEAP;
9.      plot v.$\eta_{mseen}$;
10.     FOR all $v'$ directly connected to $v$;
11.         IF ($v' \in$ HEAP )
12.             $v'.\eta_{mseen} = \max(v'.\eta_{mseen}, \eta_{max}(v, v'))$;
13.             reorder HEAP;
14.         ELSE
15.             compute $\zeta_{max}(v')$;
16.             $v'.\eta_{mseen}$=$\eta_{max}(v, v')$;
17.             add $v'$ into HEAP;
18.         ENDIF;
19. ENDWHILE;
```

Figure 1: CSV Algorithm Skeleton

Given a vertex v_i, we also want to estimate how densely it is connected to another vertex v_j. For this purpose, we introduce another definition: maximum co-clique size.

DEFINITION 4. *maximum co-clique size, $\eta_{max}(v_i, v_j)$*
The maximum co-cliques size for two vertices v_i and v_j denoted as $\eta_{max}(v_i, v_j)$ is the size of the biggest complete subgraph(i.e., a clique) that contains both v_i and v_j. □

The CSV algorithm is shown in Figure 4. The algorithm maintains a heap for storing visited vertices that have not been output. Vertices stored in the heap are maintained in sorted order based on η_{mseen} and then by ζ_{max}. The variable $v.\eta_{mseen}$ maintains the highest $\eta_{max}(v, v')$ value between v and any visited vertex v' known so far. Intuitively, sorting the vertices based on η_{mseen} means that vertices that are strongly connected to previously visited vertices will be output first. Among those that have the same value for η_{mseen}, vertices with higher value of ζ_{max} are ranked in front as they are more likely to lead the walk towards the region with higher graph connectivity.

The algorithm starts by either picking the next vertex v from the heap or a random vertex v if the heap is empty. The value $v.\eta_{mseen}$ (which have a value of 0 if it is the starting vertex) will then be output. All vertices v' that are directly connected to v are retrieved. For vertex v' which is already in the heap, $v'.\eta_{mseen}$ is updated depending on whether the original value of $v'.\eta_{mseen}$ or $\eta_{max}(v, v')$ is higher. The heap is then reordered based on the updated value. For a vertex v' which has not been visited, $v'.\eta_{mseen}$ is set to $\eta_{max}(v, v'))$ and $\zeta_{max}(v')$ is computed. After that, v' is inserted into the heap. The process is repeated until all vertices have been visited and its $v.\eta_{mseen}$ has been output.

To see how the density plot output by CSV (henceforth referred to as the CSV plot) can be used to visualize the distribution of cliques, we will first prove the follow theorem.

THEOREM 4.1. *Let v'_1,...,v'_k be the vertices of a size k clique in G and assume that they are already sorted based on the ordering computed by CSV. We claim that for any vertex v (not necessary those in the clique) that fall between v'_1 and v'_k (excluding v'_1 but including v'_k), $v.\eta_{mseen} \geq k$.*

PROOF. *: Any vertex v that falls within v'_1 and v'_k is either part of the size k clique or it is not.*

If v is part of the clique, $v.\eta_{mseen} \geq k$. This is because: (i) v'_1 has been output, (ii) all vertices v'_i in the clique are directly connected to v'_1 with $\eta_{max}(v'_1, v'_i) \geq k$, and (iii) Line 12 of CSV in Figure 4 will update $v.\eta_{mseen}$ for all vertices v that directly connects to v'_1 to the maximum of the original $v.\eta_{mseen}$ or $\eta_{max}(v'_1, v)$.

If v is not part of the clique, there must exist some i, $1 < i \leq k$ such that $v.\eta_{mseen} \geq v'_i.\eta_{mseen}$ in order for v to be output before v'_i in the heap. Since v'_2,...,v'_k must be in the heap once v'_1 is output, their corresponding value for η_{mseen} must be equal or larger than k. Since $v.\eta_{mseen} \geq v'_i.\eta_{mseen}$, it must be that $v.\eta_{mseen} \geq k$. □

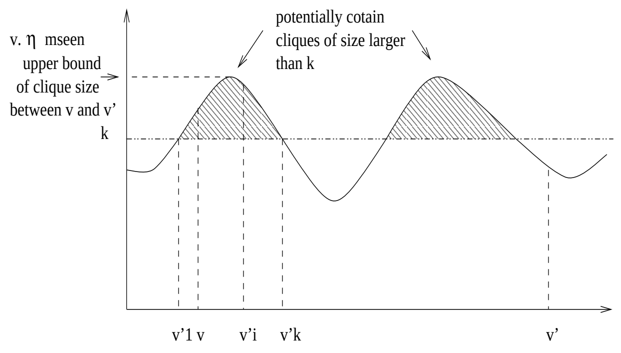


Figure 2: CSV plot correctness proof

Based on Theorem 4.1, it is easy to see that any clique with a size larger than or equal to k must fall into a region of the CSV plot in which there exists a continuous set of more than $k-1$ vertices with $\eta_{mseen} \geq k$ in the plot. As an example, Figure 2 highlights regions in the CSV plot that potentially contain cliques of size larger than k. To be more precise, the CSV algorithm orders the graph vertices into a CSV plot such that the following theorem can be inferred:

THEOREM 4.2. *The highest η_{mseen} value between two vertices v and v' on the CSV plot is an upper bound on the maximum clique size that can be found on the subgraph that is induced by considering all the vertices between v and v' in the CSV plot.* □

For a clique mining algorithm such as CLAN [33], the CSV plot provides a good exploratory tool that can highlights regions of interest for mining and also guide parameter settings such as minimum clique size for the algorithm.[2] Later on in our experimental section, we will show that by using the CSV plot as a filtering method, we can speed up the efficiency of CLAN by up to 80% while finding exactly the same set of cohesive subgraphs that the original CLAN algorithm finds.

[2]The minimum clique size setting in CLAN is used to filter off all cliques that are of a size smaller than a certain threshold

Finding Cohesive Subgraphs Based on Density
We will next discuss how the CSV plot can be used to provide a solution for Problem 2 which is to find all subgraphs G' such that $|G'| \geq size_{min}$ and $\gamma(G') \geq \gamma_{min}$. We will make use of Turan's theorem [32, 8] from extremal graph theory for this purpose.

THEOREM 4.3. *Turan's Theorem*
Let $G = (V, E)$ be a graph that contains no clique of size larger or equal to k; then

$$|E| \leq \frac{(k-2)|V|^2}{2(k-1)}$$

□

Based on Turan's theorem, we can now infer that subgraphs which exceed a certain size and density must contain at least one clique exceeding a certain size. As an example:

EXAMPLE 1. *Let us assume that $size_{min} = 10$ and $\gamma_{min} = 0.8$. Let G' be a subgraph that just satisfies that criteria i.e., it has 10 vertices and 36 edges (thus ensuring $\gamma(G') = 0.8$). Based on reasoning from Turan's theorem, such a subgraph should contain at least one clique of size 4 and above. It is also easy to see that larger graphs that satisfy the density threshold must contain cliques of even larger size.* □

As can be seen, the solution to Problem 1 which helps to identify and visualize cliques of a certain size can be used to provide markers for Problem 2. Any regions in the CSV that do not contain cliques of a sufficiently large size will not contain any subgraphs of interest in Problem 2. Note that this approach of using cliques in a subgraph to measure its cohesiveness is widely accepted in social network analysis [29].

Finding Cohesive Subgraphs Based on Connectivity
We next examine the usefulness of the CSV plot in handling Problem 3. Note that Problem 3 can be converted to a special instance of Problem 2 by observing the following:

1. A k-connected graph must contain at least k vertices.
2. A k-connected graph must contain at least $\lceil(kn/2)\rceil$ edges since each vertex must connect to at least k-1 other vertices.

Based on these observations, we can reduce an instance of Problem 3 into an instance of Problem 2 where $size_{min} = k$ and the minimum density, γ_{min} is set to $\frac{2\lceil(kn/2)\rceil}{k*(k-1)}$. The CSV plot can thus be applied to solve Problem 3 as it is applied to solve Problem 2.

4.1 Multi-Dimensional Mapping

As $\zeta_{max}(v)$ and $\eta_{max}(v_i, v_j)$ is computational expensive due to the clique detection, in this section, we propose a multi-dimensional mapping which computes an upper bound of these two functions.

Given the graph G, we achieve multi-dimensional mapping by first selecting n vertices as **pivots** based on the

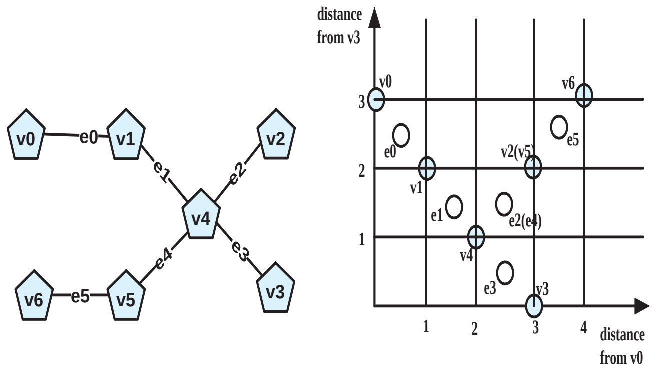


Figure 3: An Example of Graph Mapping

shortest path distance. Various methods for selecting these pivots exist [16, 28, 10]. Here, we adopt a simple strategy of incremental selection. We iterate n rounds and select a vertex that is furthest away (based on average distance) from pivots selected from previous rounds.

Given a pivot p_i, the distance of the shortest path of a vertex v to p_i is denoted as $SD_i(v)$. For an edge e in the graph, we define $SD_i(e)$ as $\frac{1}{2} \times (SD_i(v_1) + SD_i(v_2))$ where v_1 and v_2 are the two vertices connected by e. We can now define mappings of graph elements into n-dimensional space:

DEFINITION 5. **M(v), M(e)**
Given a graph $G = \{V, E\}$ and a set of pivots p_1,...,p_n, a vertex v will be mapped into the point $M(v) = (SD_1(v), ..., SD_n(v))$ while an edge e will be mapped into the point $M(e) = ((SD_1(v)+SD_1(v'))/2, ..., (SD_n(v)+SD_n(v'))/2)$ where $e = (v, v')$. □

Figure 3 shows an example of how a graph is mapped into two-dimensional space by picking two pivots v_0 and v_3. The mapping essentially divides the n-dimensional space into grid cells of unit length and vertices are mapped into the intersection of the grid lines with integer coordinates. Edges on the other hand are mapped exactly in between the mapping of the two vertices that they connect. We will use $D_\infty(M(v_1), M(v_2))$ to represent the distance between $M(v_1)$ and $M(v_2)$ under L_∞ norm, i.e.,

DEFINITION 6. $D_\infty(M(v_1), M(v_2))$
$D_\infty(M(v_1), M(v_2)) = max_{i=n}^{i=1} |SD_i(v_1) - SD_i(v_2)|$ □

Based on triangular inequality, it is possible to prove the following lemma:

LEMMA 4.1. *Let the length of the shortest path between two vertices v_1 and v_2 be $D(v_1, v_2)$, then $D_\infty(M(v_1), M(v_2)) \leq D(v_1, v_2)$*

PROOF. *Assuming otherwise, then there exists a pivot p_i such that $|SD_i(v_1) - SD_i(v_2)| > D(v_1, v_2)$. Without loss of generality, $SD_i(v_1) > D(v_1, v_2) + SD_i(v_2)$, which means that $SD_i(v_1)$ is not the shortest path distance to v_1. This is a contradiction.* □

THEOREM 4.4. *Let $G' = \{V', E'\}$ be a clique of size k in G, then G' will be mapped into a unit grid cell C based on our high dimensional mapping such that:*

- *There exist at least k vertices with a degree greater than $k - 1$ in C.*

```
Algorithm 2: Estimate $\eta_{max}$ and $\zeta_{max}$
Input: Mapping of graph $G$, an empty R-tree $R$
Output: Estimate $\eta_{max}(v, v')$ and $\zeta_{max}(v)$ $\forall v, v' \in G$
Method:
1.  set $v.\zeta_{max} = 0$ for all $v \in G$;
2.  FOR each edge $e \in G$
3.      construct set of cells $C$ containing $e$;
4.      FOR each cell $c \in C$
5.          IF $c$ is found in $R$
6.              Add $e$ into $c$;
7.          ELSE
8.              Add $e$ into $c$;
9.              Insert $c$ into $R$;
10.     ENDFOR
11. ENDFOR
12. FOR each vertex $v \in G$
13.     Locate existing cells set $C$ in R that contains $v$;
14.     Add $v$ into every cell $c \in C$;
15. ENDFOR
16. FOR each $c$ in R
17.     Est_$\eta\zeta(c)$;
18. ENDFOR
```

Figure 4: Estimating ζ_{max} and η_{max} computation algorithms

- *There exist at least $k(k - 1)/2$ edges in C.*

PROOF. *All vertices in a clique are one edge away from each other. Combining this with Lemma 4.1, we know that each pair of vertices v'_1, $v'_2 \in V'$ will be mapped into n-dimensional space such that $D_\infty(M(v_1), M(v_2)) \leq 1$. Since this must be true for all pairs, the only possibility is that all vertices in the clique are mapped into the same grid cell of unit length. Similarly, since edges are mapped between the two points they connect, they must only be found in the same grid cell. Furthermore, since the vertices belong to a clique of size k, they must at least have degrees of $k - 1$ in order to connect to all the other vertices in the clique.* □

We will now explain how an upper bound for $\zeta_{max}(v)$ and $\eta_{max}(v, v')$ can be computed for each vertex v and each of it's connecting vertices v'.

Referring to Figure 4, our algorithm first computes the coordinates of all vertices and edges of the graph after mapping and then insert them into an R-tree[3]. Our R-tree stores unit cells. Each cell has unit length in all dimensions and stores a list of vertices and edges that are mapped into it. To avoid introducing cells that contain only vertices but no edges, we first insert edges into the R-tree and then subsequently the vertices. Complexity arises when an edge is mapped into the boundary of a cell. In that case, more than one cells are added into the R-tree. This happens only when the vertices at the two end of an edge are having the same distance to one or more of the pivots. To minimize such situation, we design the pivot selection method such that the group of pivots are far apart from each other. Experi-

[3]Note that we use an R-tree as it is rather common and readers should feel free to use other spatial indexes for this purpose.

```
Algorithm 3:Est_$\eta\zeta(c)$
Input: A unit cell $c$
Output: Estimate $\eta_{max}$ and $\zeta_{max}$ for vertices in
$c$,
update value of $\eta_{max}$ or $\zeta_{max}$ when necessary
Method:

1.  FOR every $v$ mapped into $c$
2.      IF $c$ is sufficiently "dense" to change $\zeta_{max}(v)$
        or $\eta_{max}(v, v')$
3.          $V' = \{v'|v'$ directly connects to
            $v$ and $v'$ is mapped into $c\}$;
4.          FOR each $v' \in V'$
5.              $V''$ =$\{v''|v''$ connects to both $v$,$v'$ $\bigwedge v'' \in V'\}$,
                    Let $G''$ be subgraph induced from $V''$
6.              Compute degree array $DV''\forall v'' \in V''$
                such that $DV''(v'')$=$degree(v'', G'') + 1$;
7.              $DV'(v')$=$\eta$_bound($DV''(v')$);
8.          ENDFOR
9.          $DV = \eta$_bound($DV'$);
10.         FOR each $v' \in V'$
11.             IF ($\eta_{max}(v, v') < DV'(v')$)
12.                 ($\eta_{max}(v, v') = DV'(v')$);
13.                 IF ($\zeta_{max}(v') < \eta_{max}(v, v')$)
14.                     $\zeta_{max}(v') = \eta_{max}(v, v')$;
15.                 END IF
16.                 IF ($\zeta_{max}(v) < \eta_{max}(v, v')$)
17.                     $\zeta_{max}(v) = \eta_{max}(v, v')$;
18.                 ENDIF
19.             ENDIF
20.         ENDFOR
21.     ENDIF
22. ENDFOR
```

Figure 5: Algorithm to estimate η in c

ments on pivot selection methods in later part of this paper support our choice.

After the insertion of all graph elements, the algorithm will compute $\eta_{max}(v)$ and $\eta_{max}(v, v')$ by processing each cell using the algorithm in Figure 5.

Given a cell c, algorithm Est_$\eta\zeta$ in Figure 5 will compute an upper bounds for $\eta_{max}(v, v')$ for every node v inside c and updates $\zeta_{max}(v)$ or $\eta_{max}(v, v')$ for some v' the upper bound computed are found to be higher than their original value. To achieve this, the algorithm iterates over all vertices mapped inside c. We only start the estimation if c contains enough edges and vertices such that c has potential to update some η_{max} or ζ_{max} values that are relevant to v(i.e. c is sufficiently "dense" enough for v in line 2). For each neighbor vertex v' of v, we find V'' a set of vertices that are connected to both v and v' and which are also found within c (V'' also contains v, v'). We induce a subgraph G'' that contains all vertices in V'' and all edges in the original graph that join two vertices in V'' (line 5). Based on G'', we compute an array DV'' where $DV''(v'')$ stores the degree of v'' within G''. This is the largest possible clique size v'' can participate in G''. It is also a loose upper bound of $\eta_{max}(v', v'')$.

Since a clique of size k must contain k vertices each with degree of $k - 1$, we will pass DV into a function called η_bound(DV) in Figure 6 which will compute an upper bound on the size of the biggest clique that could occur

```
Function 1: $\eta$_bound($DV$)
Input: $DV$: an array of $\eta$ bounds
Output: a new array of tighter $\eta$ bounds
Method:

1.  $\eta_{new} = 0$ ;
2.  S = $DV$ in descending order ;
3.  FOR i=1 TO $|DV|$
4.      IF ($i \geq S(i)$)
5.          $\eta_{new} = S(i)$ ;
6.          BREAK FOR LOOP ;
7.      ENDIF
8.  ENDFOR
9.  FOR i=1 TO $|DV|$
10.     IF ($DV(v_i) > \eta_{new}$)
11.         $DV(v_i) = \eta_{new}$ ;
12.     ENDIF
13. ENDFOR
14. RETURN $DV$ ;
```

Figure 6: Function η_bound

within G''. This is done by sorting DV in descending order, storing it in an array S and going through S starting from the first element until the i^{th} element is greater or equal to $S(i)$. Once the condition is satisfied, i will represent the size of the largest possible clique in G'' and will be use to update η_{new}. Line 9-13 then update DV such that $DV(v_i)$ stores the size of the largest clique that v_i can participate in G''.

Once the function is terminated, Line 8 of Algorithm Est_$\eta\zeta$ will update $DV'(v')$ with the returned value from function η_bound.$DV'(v')$ thus represent the biggest clique that v and v' can participate in. Our next step is based on the observation that a vertex v that participate in a clique of size k must have at least $k - 1$ direct neighbors that participate in a clique of size k. As such, we pass DV' to function η_bound in Line 10 of Algorithm Est_$\eta\zeta$ to find the largest k, such that are k vertices v' which could participate with a clique of size k together with v. Upon exiting from Est_$\eta\zeta$, DV' could contains a better bound that DV'' on $\eta_{max}(v, v')$ for each $v' \in V'$. The values in DV' are then used to update the value of η_{max} and ζ_{max} for v and v' if the estimated upper bound is higher than the original value. Note that in our algorithm description, we have certain redundancy in that when v' is being processed, the same subgraph G'' will again be induced for edge (v', v). This can be easily avoid by imposing an ordering on the vertices and we omit the description to keep our discussion clean.

To illustrate how our algorithm in Figure 5 works, we present an example in Figure 7. To estimate $\eta_{max}(a, f)$, we locate the neighborhood of a and f, $\{a, b, c, d, e, f, g\}$. After sorting the degree array in descending order, we have array $6(a), 6(f), 5(d), 4(b), 4(c), 4(e), 3(g)$ (here we attach the vertex id to each degree value for easy interpretation). Function η_bound(DV) infers from the sorted array that the upper bound of $\eta_{max}(a, f)$ is 5. Similarly, we can estimate an upper bound on η_{max} between a and all its neighbors($\eta_{max}(a, f) =$ 4, $\eta_{max}(a, c) = 4$, $\eta_{max}(a, d) = 4$, $\eta_{max}(a, e) = 5$ and $\eta_{max}(a, g) = 4$) and store them in DV'. After calling function η_bound with DV' , we tighten the value of $\eta_{max}(a, f)$ to be 4, which is in fact the correct value for $\eta_{max}(a, f)$.

Since $\eta_{max}(v, v')$ and $\zeta_{max}(v)$ are independent of CSV

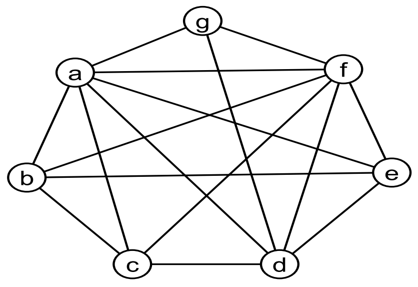


Figure 7: Estimation η between a and its neighbors

traversing order, we can pre-compute them and access them directly when computing the CSV plot. We store the values of all $\eta_{max}(v, v')$ inside a table of $|E|$ entries and all values of $\zeta_{max}(v)$ inside a table of size $|V|$.

Note that since the spatial mapping overestimates the clique size and our approximation computation of η_{max} and ζ_{max} are also upper bound, Theorem 4.1 and 4.2 will apply here. Correspondingly, any CSV plot that is computed based on our approximation method will still be useful for solving Problem 2 and 3.

Complexity Analysis The execution of CSV consists of three parts. The multi-dimensional mapping time, the tree building time and the core algorithm running time.

The multi-dimensional mapping process is the process of computing the shortest path distance between the n pivots and the rest of the vertices. The standard heap implementation has overall time complexity of $O((|V| + |E|)\log|V|)$.

The R-tree that is built on the graph vertices and edges has a complexity of $O((|V| + |E|)log(|V| + |E|))$ if each vertex and edge is inserted once into the tree. The worst case is when each edge is mapped into 2^n grids and so does each vertex. The complexity increase to $O(((|V| + |E|)2^n)log((|V| + |E|)2^n))$. For η estimation algorithm in figure 5, if a vertex has degree d_v, the first run of estimation requires d_v rounds of sorting d_v vertices. the total complexity is $O(d_v^2 \log d_v)$. The next run of tightening the upper bound of η requires $O(d_v \log d_v)$. The overall complexity for one vertex is thus $O(d_v^2 \log d_v)$. Due to the uncertainty of mappings, the distributions of the vertices vary. The worst case arises when the graph is a $|V|$-clique. This $|V|$-clique are mapped to 2^d grids identically. Inside each grid, there are $|V|$ vertices and $|E|$ edges. the complexity is thus $O(|V|^2 \log |V| 2^d)$. By checking whether a cell is sufficiently "dense" (as mentioned in algorithm in figure 4), the number of cells we actually need to perform η estimation is greatly reduces.

The complexity of the core CSV algorithm depends on the number of comparisons to decide which vertex should enter the stack. Before any vertex is output, all its neighboring vertices are checked. The total complexity is thus $O(|E|)$.

Mapping	$O((\|V\| + \|E\|)\log\|V\|)$
Tree Building	$O(\|V\|^2 \log \|V\| 2^d)$
CSV core	$O(\|E\|)$

Table 1: CSV components complexity

The time complexity for each components of CSV are listed in Table 1. Note that the above scenarios are extreme cases that occurred for extreme graphs. Experiments on SMD datasets and DBLP datasets shows the number of grid cell are far below the extreme case. Thus our algorithm achieves better performance for real scenarios.

Example Output

To give an example on how the output from CSV looks like, we generate a synthetic graph with four cliques of size 8, remove 30% of the edges and then embed the cliques into a random graph. Figure 8(a) shows the generated graph with 60 vertices. The numbers shown on the vertices indicate the order of DENSUE's walk on the graph, and the corresponding plot using five pivots for mapping is shown in Figure 8(b). Instead of being cliques of size 8, they now become an overlapping set of cliques of size 6 or 7. However, since their connectivity is still higher than the vertices outside the cliques, their presence can still be discerned.

In conclusion, unlike existing "blackbox" algorithms for finding cohesive subgraphs, our visualization plot goes beyond highlighting them to showing their distributions and how they interact with other components in the graph.

5. EXPERIMENT

Our experiments are evaluated on a Windows-based machine. The machine has P4 3GHz CPU, 1G RAM and 75GB hard disc with Windows XP installed.

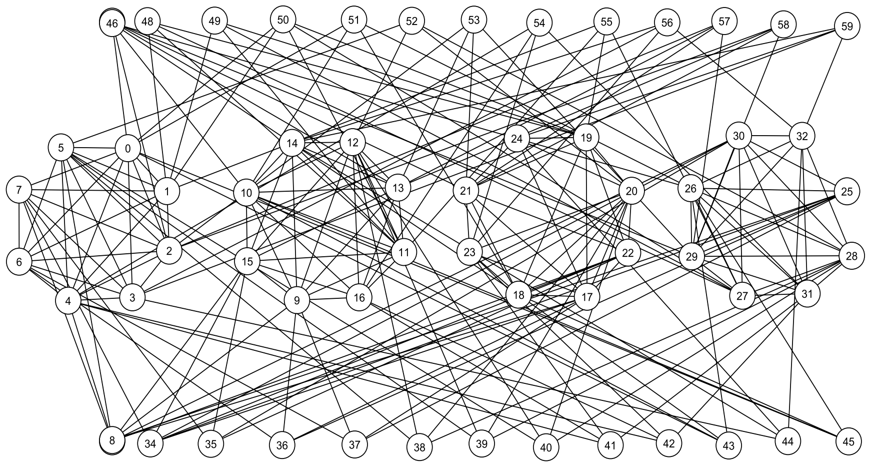
We use DBLP 10 years' co-authorship dataset and Stock Market(SMD) data used in algorithm evaluation in [37] to evaluate the effectiveness and efficiency of CSV. The DBLP data set covers co-authorships across year 1997 to 2006. Each graph vertex represents an author and two authors are connected by an edge if they co-authored in at least one publication within a year. We consider the co-authorship relations is significant only when the two researchers work together for at least two years. The compound DBLP 10-year co-authorship graph contains 2819 vertices and 4808 edges.

The SMD data is a collection of three sets of 11 graphs from [37] which are named SMD-95, SMD-93 and SMD-90 respectively. Here, 0.95, 0.93 and 0.90 are correlation thresholds and an edge exists between two stocks in the graphs if their correlation is found to be above the correlation threshold. Table 2 shows the statistics of these three sets of graphs. For each set of 11 graphs, we create a summary graph in which an edge exists between two stocks if it is found in more than a certain number of graphs. Depending on the threshold applied to each of the datasets, the final summary graphs' size varies. For all our evaluation, the number of pivots are set to 4 unless otherwise stated. We abstract the largest connected component from each data set and performs CSV and CLAN on it.

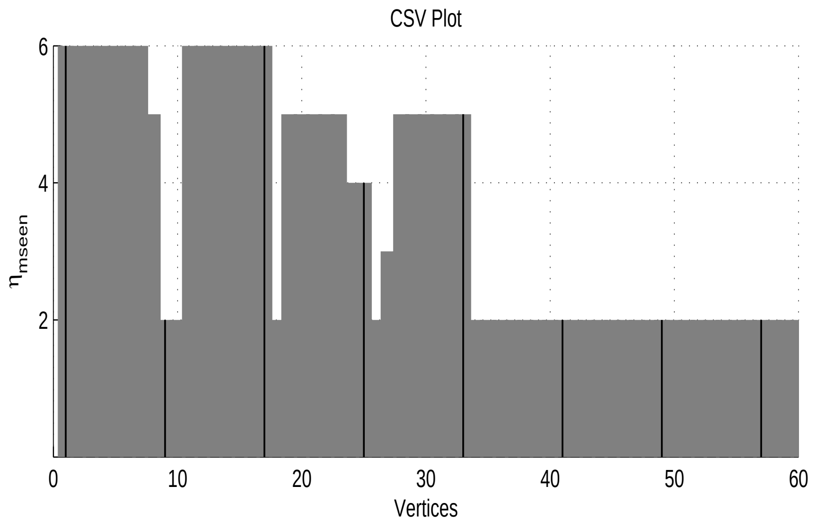
Datasets	# Graphs	Avg. # of vertices	Avg. # edges
Stock Market 0.95	11	1683	20074
Stock Market 0.93	11	2618	68608
Stock Market 0.90	11	3636	206747

Table 2: Stock Market Datasets (SMD) statistics

5.1 Effectiveness of CSV Plot



(a) Graph With Embedded Partial Cliques



(b) Corresponding CSV Plot

Figure 8: An Example of CSV Plot

Sup.	SMD-95		SMD-93		SMD-90	
	V	E	V	E	V	E
1	4264	132359	5323	403190	6008	1064133
2	3409	55945	4617	200882	5498	613823
3	2373	16462	3809	88301	4931	314421
4	1600	6893	2968	35892	4263	150692
5	425	1680	2148	14368	3587	69412
6	237	756	669	1771	2760	31462
7	84	315	361	723	1024	5611
8	65	183	219	364	635	3032
9	45	99	93	504	450	1681
10	13	20	214	409	356	1009
11	10	12	123	230	242	522

Table 3: Statistics of Largest Connected Components for Stock Market Datasets' Summary Graphs(SMD)

In this section, we will look at the effectiveness of the CSV plot.[4]

5.1.1 DBLP Plot

We first test CSV on the DBLP dataset. As an instance of social networks, DBLP data set reflects various social processes such as information processing, distributed search and diffusion of social influence [23]. We experiment on the largest connected component of the compound ten years' DBLP co-authorship data and show it's CSV plot in Figure 9.

As an example to show how the distribution of cliques are depicted by the CSV plot, we show subgraph $sg1$ in

[4]In support of the SIGMOD'2008 experimental repeatability requirement, the ordering of the DBLP (Figure 9) and stock(Figure 14) datasets together with their η_{mseen} values are made available at:
http://www.comp.nus.edu.sg/˜atung/publication/cri.zip

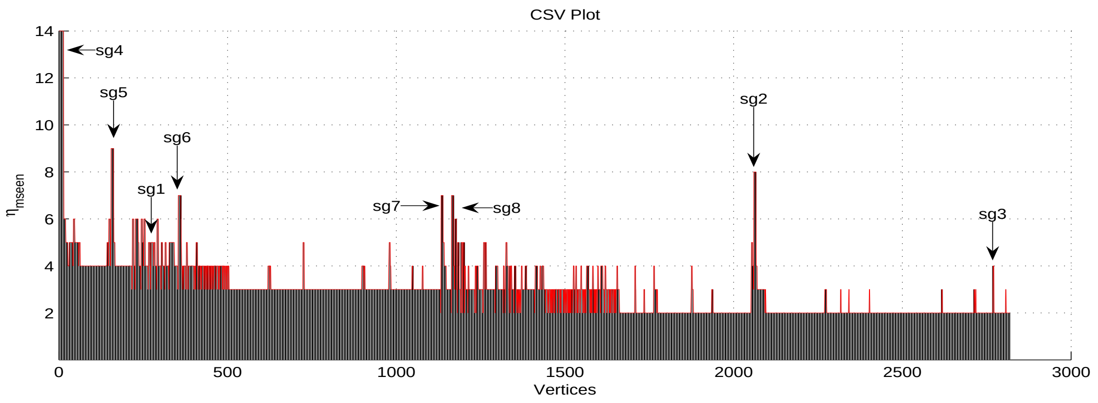


Figure 9: CSV Plot for DBLP 97-06 Co-authorship Graph

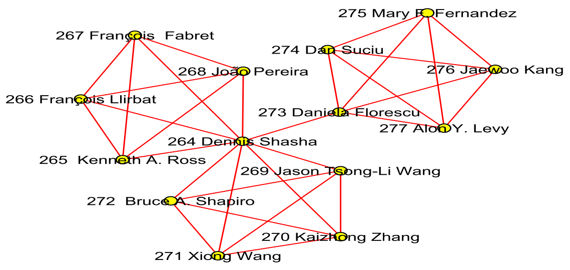


Figure 10: Cliques joined by 2 co-authors in $sg1$

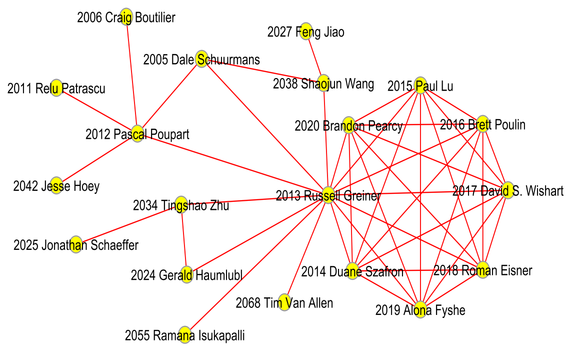


Figure 11: A large clique in $sg2$

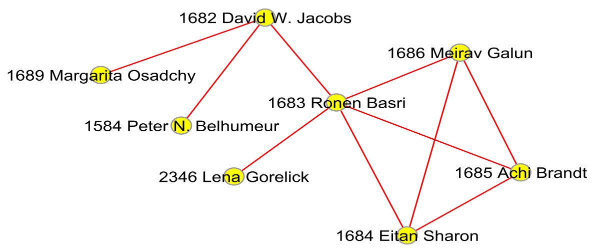


Figure 12: Small clique in $sg3$

No.	Size	Order	Members
$sg4$	14	0-13	J. Miller, S. Sudarshan, M. Nemeth, Rajeev Rastogi, Jerry Baulier, Abraham Silberschatz, Peter McIlroy, P. P. S. Narayan, Henry F. Korth, A. Khivesera, C. Gupta, Philip Bohannon, S. Joshi, S. Gogate
$sg5$	9	153-161	Volker Markl, Rudolf Bayer, Timos K. Sellis, Roland Pieringer, Klaus Elhardt, Frank Ramsak, Robert Fenk, Aris Tsois, Nikos Karayannidis
$sg6$	7	352-358	Mitch Cherniack, Michael Stonebraker, Ugur Ccediletintemel, Anurag Maskey, Stanley B. Zdonik, Nesime Tatbul, Donald Carney
$sg7$	7	1131-1137	Bernhard Schoumllkopf, Thomas Navin Lal, Wolfgang Rosenstiel, Michael Schroumlder, N. Jeremy Hill, Thilo Hinterberger, Niels Birbaumer
$sg8$	7	1162-1168	Peter M. G. Apers, Martin L. Kersten, Henk Ernst Blok, Roelof van Zwol, Willem Jonker, Milan Petkovic, Menzo Windhouwer

Table 4: Large Cliques in DBLP

Figure 10 which consist of vertices ordered from 264 to 277 in the CSV plot. As can be seen from the plot, there are multiple small peaks rising from an otherwise flat region. These are in fact small cliques that are joined together by some co-authors from each clique. In this case, $sg1$ contains three cliques that are joined together by 2 authors, namely, Dennis Shasha and Daniela Florescu. Here, Dennis Shasha is a member of two of the cliques while Daniela Florescu is a member of the remaining clique with all three cliques having a size of 5.

Next, we will show an example of a highly connected subgraph marked as $sg2$ in Figure 9 and displayed in Figure 11. Since the peak of $sg2$ seems to featured prominently in a neighborhood with low cohesiveness, we also abstract some partial neighbors of $sg2$ to confirm our suspicion. As can be seen from Figure 11, $sg2$ in fact represent a group whose member are Russell Greiner, Duane Szafron, Brett Poulin , David S. Wishart, Roman Eisner, Alona Fyshe and Brandon Pearcy. Within the CSV plot, they are ordered consecutively from 2013 to 2020. From the snippet in Figure 12, we can identify a "hub" person: Russell Greiner who is ordered as the first person within $sg2$. Indeed he is a well-known researchers in areas such as Bayesian Networks and leads several research groups which explains his important role in linking up part of the DBLP co-authorship graphs. In general, a sudden raise in the CSV plot usually indicates a key vertex which removal may greatly affects the density distribution of the graph. Note the "sparseness" in other part of the graph in Figure 11 which corresponds to the low density region around $sg2$ in the CSV plot.

As we have stressed, the strength of the CSV plot is not only on its ability to find cliques but to also present the overall density distribution of a graph. As can be seen from Figure 9, cliques of size 4 are in fact available in abundance in the DBLP graph and a "blackbox" pattern mining algorithm will have found many cliques of size 4 without knowing their relationship with their neighborhood. From the CSV plot however, we can see that the subgraph $sg3$ is prominent within its neighborhood despite it being just one of the many size 4 cliques in the graph. The subgraph $sg3$ (Figure 12) consists of vertices ordered from 1683 to 1686 in Figure 9 and researchers in the group include: Ronen Basri , Eitan Sharon, Achi Brandt and Meirav Galun. These four researchers' major research interests are in computer vision and maths. Besides common research interests, they all currently work in Israel. Without prior background information of these researchers, we will not identify this group of researchers from other cliques of size 4 in the DBLP graph.

Finally, we show the remaining cliques that are of size 7 and above in Table 4. Among these cliques, $sg7$ and $sg8$ are the closest to each other on the CSV plot. This can be explain by the fact that both groups essentially work in Germany. However, $gp7$ consists of members who do research in AI while $gp8$ consists of members who do research in databases.

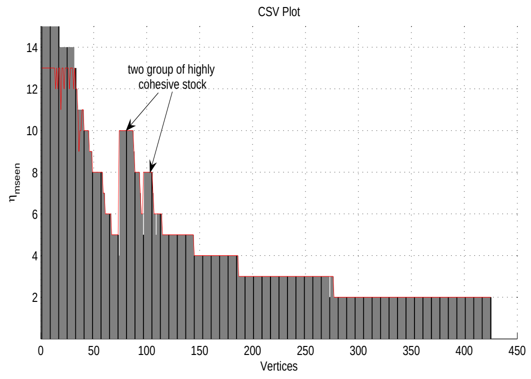


Figure 14: 4D SMD95 CSV Plot with 45% Support Threshold

To assess the accuracy of our estimation for η_{max} and ζ_{max}, we also plot in red the actual η_{mseen} computed by CLAN in Figure 9. As can be seen, our estimation of the η_{max} and ζ_{max} are highly accurate resulting in plot that show us the various interesting discoveries that we have discussed earlier.

5.1.2 Stock Market Data

We also run CSV on SMD-95 with a support threshold of 45% (i.e. 5 graphs) and generate a CSV plot shown in Figure 14. From the CSV plot, we identified two closely related cliques which are at position 73 to 104. The subgraph induced from the stocks within this region are shown in Figure 13. From the figure, we can see two distinct cohesive subcomponents that are yet closely related. Such form of nested structures will not be easily detected by normal "blackbox" pattern mining algorithm but is easily indicated in the CSV plot. The larger components consists of 14 stocks participating in 10-cliques. Out of the 14 stocks, 5 stocks belong to BlackRock Group: BlackRock Broad(BCT), BlackRock Advantage Term Trust, Inc(BAT), BlackRock Global Investment Management (BGT), BlackRock Investment Quality Term Trust(BQT), BlackRock Municipal Target Term Trust (TTR). Anther well-known investment bank Merrill Lynch (NBM) announced in Feb 2006 that it would combine with BlackRock to form the World's Largest Independent Investment Management Firms. The rest of the 8 stocks are in areas of investment management as well. Two stocks in fact are the same trust fund over different years(NGI and NGF are 2003 and 2004's national Government Income Term Trust respectively). We thus infer that this large dense subgraph is formed mainly by BlackRock Group and Merrill Lynch with other stocks in investment management field. The other smaller component (9 stocks forming 8-cliques) has four bank and financial service stocks: Fidelity Bancorp (FFFL), Seacoast Financial (SCFS), Jacksonville Bancorp (JXVL) and PennFed Financial (PFSB). Both the components have interest in insurance and as such they are linked by some health service stocks such as Davita (DVA). CSV plot makes it possible to identify how closely related these stocks are to each other without drilling down to individual stocks.

Like in the DBLP case, we also indicated in red the actual η_{max} values in Figure 14. As can be seen, our estimation of the clique sizes are rather accurate. We also investigate how the CSV plot of SMD-95 is affected when varying the number of pivots by computing the CSV plot with 6, 8, 10 and 12 pivots. However, due to the effectiveness of our bounding method, these plots look mostly the same. As such we show only the plot for 12 pivots here in Figure 15.

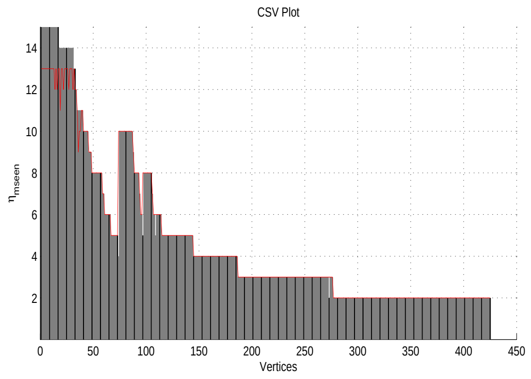


Figure 15: CSV Plot with 12 pivots

5.2 Efficiency

5.2.1 Graph Size and Running Time

By applying CSV on a set of relevant graphs with different sizes, we are able to see how the sizes of the graphs affect the running time of CSV. Experiments are ran on three sets of stock market data (SMD-90, SMD-93 and SMD-95) with

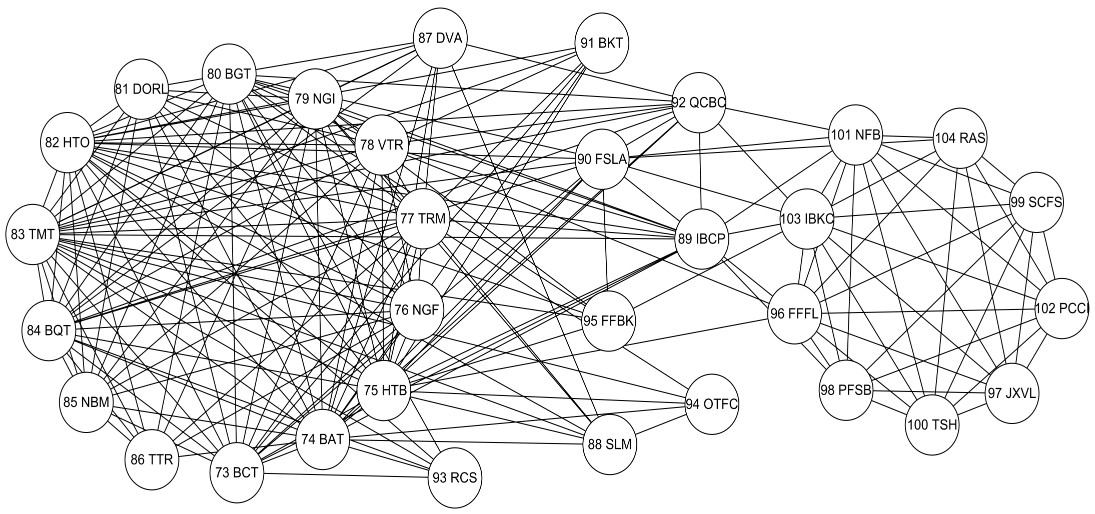


Figure 13: Two groups of highly cohesive stocks

different support thresholds to vary the size of the graphs. Readers are referred to Table 3 for the size of the largest connected component summary graphs. Note that the number of edges doubles for every decrease of 1 in the support threshold.

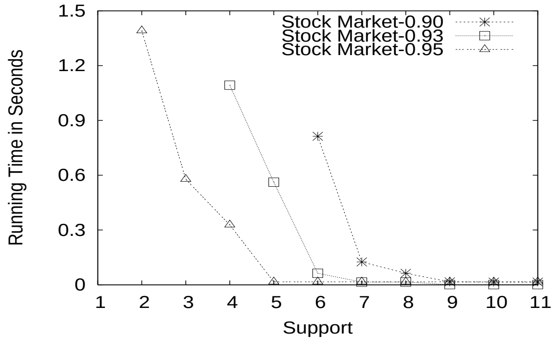
Figure 16(a), 16(b) and 16(c) indicate the running time of the three components of CSV on the three sets of SMD data. In Figure 16(a), we show the spatial mapping time for the three sets of data. As stated previously, the mapping is performed on the largest connected components of the graph. The mapping time ranges from 0.015 sec to 1.391 seconds for graph with size 10 vertices and 13 edges to graph with 3409 vertices and 55945 edges. The mapping process only takes up 0.25-1% out of total CSV running time for a large scale graph. The major time consuming part of CSV algorithm is the tree building process. Figure 16(b) shows the time spent on building the R-trees on the same datasets. The building time does not exceed 2500 seconds for the largest graph that we have process (smd95sp2 with 3409 vertices and 55945 edges). CSV core algorithm's running time on SMD datasets is shown in Figure 16(c). After building up the tree structure, the exploration process is within a second. Note that sometimes the running time of CSV is not monotonically increasing with the graph sizes. The distribution of the edges inside a graph also determines the running time. Graphs with similar number of vertices and edges may be mapping into different number of grids. It is the number of grids and number of elements inside the grids that determine the total running time of CSV algorithm.

We also compared the overall running time of CSV (Mapping + Tree Building + Core Algorithm) with that of CLAN on SMD data sets. From Figure 17, CSV outperforms CLAN both in terms of capability and efficiency. CSV is able to handle graphs 2-4 times larger than CLAN while the running time is only 10%-1% of CLAN when dealing with moderate to large graphs. CSV proves itself to be a good density estimation tool compromising little accuracy with an order of magnitude time savings over exhaustive mining algorithm like CLAN.

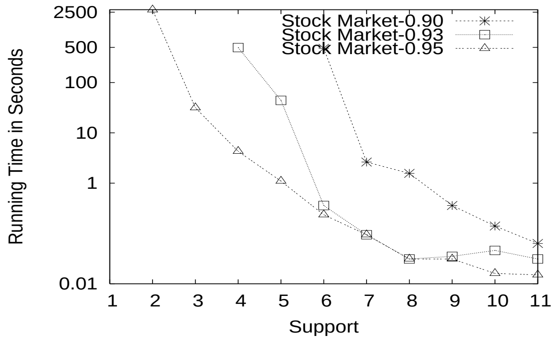
5.2.2 *Pivots Selection Algorithm and their Effect on Running Time*

Although the number of pivots does not affect the accuracy of the CSV plot significantly, we are still interested in the cost when varying the number of pivots. Thus we investigate the effect of the number of pivots selected on the running time over the same SMD-90 datasets. Figure 18 presents the total running time comparison of CSV algorithm on 4D to 12D multi-dimensional points of SMD-90 datasets (with support from 8 to 11). As we increase the number of pivots, the mapping dimension increases accordingly, which results in exponential increase in the number of non-empty grids. During the process of mapping, we are only interested in grids with edges in it. A large portion of grids thus become empty and CSV does not need to spend time exploring those grids. The running time is thus not exponentially increasing with number of pivots. This suggests that future users should be cautious when choose more pivots in the mapping process. Our suggestion here is not to use more than 6 pivots.

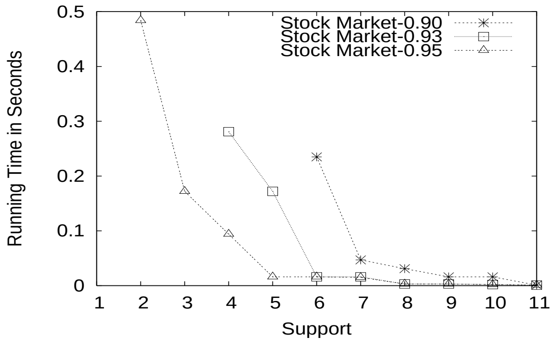
The algorithm for selecting the pivots also affects the efficiency of CSV. Figure 19 shows three lines representing three different approaches for selecting 4 pivots on the SMD-95 dataset. The "Random" approach simply picks the pivots randomly. The "Separated" approach is the one we discussed in previous sections while the "Central" approach is to pick the pivots with the minimum distance to its furthest vertex. Instead of directly plotting the algorithm's running time for the three different approaches, we plot the number of non-empty grid cells that are being handled in the algorithm. This quantity better reflects the pivots' quality. Results are averaged over 5 runs to off-set the randomness of the first approach. From Figure 19, the number of visited grids is



(a) 4D Mapping Time



(b) Tree Building Time



(c) CSV Core Algorithm Running Time

Figure 16: CSV components Running Time on Stock Market Datasets

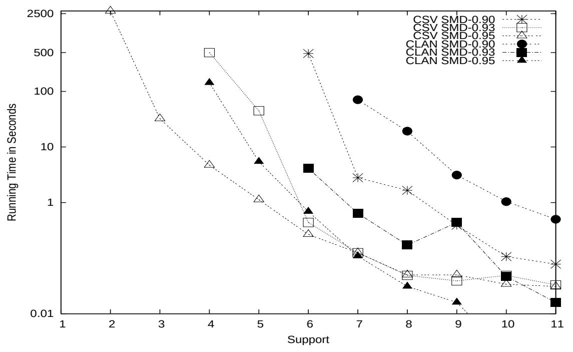


Figure 17: CSV vs CLAN Running Time

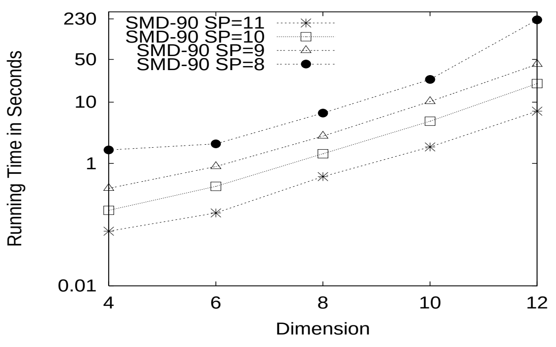


Figure 18: CSV Core Algorithm Running Time Varying Dimensions

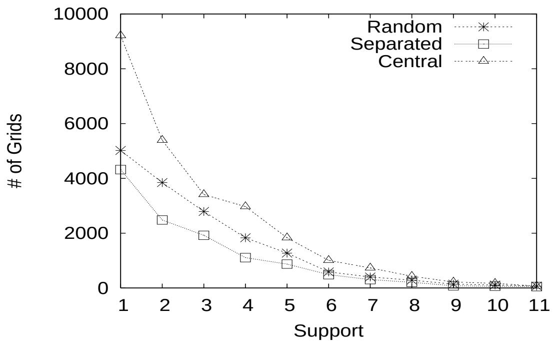


Figure 19: Three Different Pivots Selections Scheme and resulting #Grid

significantly reduced by carefully picking the pivots.

In conclusion, our experiments show that CSV is an efficient and effective tools for visual mining and exploration of dense subgraphs. CSV provides a very useful alternative to the "blackbox" and exhaustive enumeration approach of other graph pattern mining algorithms [37, 33, 17, 35], which often have more parameters to specified as well.

5.3 CSV as a pre-selection method

Since CSV provides a method to quickly estimate a graph's density distribution and CLAN is an exact algorithm computing graph's max closed cliques, we combine the two algorithms in the hope to achieve fast and accurate computation of graph density distribution. In this section, we first apply CSV on the stock market data to obtained an estimation of graph vertices density. After that, we select those vertices that could potentially contain cliques of required size and apply CLAN on the subgraphs that is induced by these vertices. As expected, CSV does not miss any cliques based on what we have proved earlier. Figure20 shows the running time of such an approach compared with that of directly applying CLAN on the data set. This set of experiments are run on 11 sets of stock market data. The x-label of the graph indicates the characteristics of the data set. For example, "90-11-5" denotes smd90 data with absolute support 11 and the minimum clique setting is set to 5 (i.e. we want to find all cliques of size 5 and above). The results show that running CSV as pre-selection method for CLAN saves 23% to 84% of the time compared to running CLAN alone.

Note that the same method here can be used to find the top-k largest cliques in the graphs by iteratively selecting the highest peak on the CSV plot and running CLAN on the region around the peak. We plan to explore more towards this direction in the future.

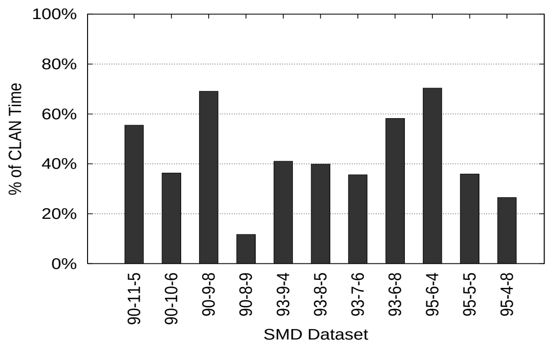


Figure 20: Efficiency of CSV as a pre-selection method

6. DISCUSSION

Next, we discuss further extensions that can be done to CSV .

1. Pivots: The handling and use of pivots can be extended in at least two directions. First, since the selection of the pivots is done initially without a good understanding of the distribution, refinement of pivots selection could be done after the CSV plot is available. Intuitively, if a pivot is selected from a highly connected region in the graph, it's shortest path distances to other vertices in the highly connected region will be short, making it difficult to separate these vertices apart after the mapping. One can also take advantage of spectral plots in this regard. As such, reselecting pivots from less dense regions of the CSV plot could serve to improve the quality of the plot. Second, as mentioned earlier, it may make sense to add in additional pivots when there is a need to hone in on smaller subgraphs.

2. Dynamic Graphs: In the case of dynamic graphs, edges and vertices could be added and deleted dynamically over time. Adapting CSV to monitor emerging or diminishing high connectivity subgraph is both a challenging and interesting problem. A naive extension for deleting/updating vertices/edges is as follows.

- For vertex/edge deletion, the corresponding mapped region in the high dimensional space can easily be updated.
- For added vertex/edges, the shortest path distance to the pivots can be approximated by the vertices or edges in the old graph that it connects to and the mapped region can be updated accordingly.
- For regions in which the order of CSV walk changes, one needs to simply redraw the CSV plot.

Obviously, the above does not take into account the exact changes in shortest path distance when edges and vertices are added/removed. We **conjecture** however that since cliques are closely connected, any error in the estimated shortest path distance due to the updates will be propagated to all vertices in the cliques, causing them to still map to the same grid cell. This is a subject which is worth studying in the future.

3. Handling Directed Graphs: The handling of directed graph could be useful for some applications like keyword search [18, 19, 6] where we want to measure the connectivity between keywords. Applying CSV on a directed graph takes on additional complexity in the following respects. First, vertices might not be reachable from the pivots selected. This can be overcome by adding virtual root node to the graph using techniques described in [31]. Second, after mapping the edges into the high dimensional space, we must record their directions within the grid cell (i.e. the vertex it connects to) and take them into account when computing connectivity. The details of such an approach will be ironed out as part of our future work.

7. CONCLUSIONS

In this paper, we propose CSV , an algorithm for mining and visualizing cohesive subgraphs. Existing approaches to the problem typically perform an exhaustive enumeration and output a set of cohesive subgraphs which are often difficult to correlate and understand. CSV relies on a locally measurable notion of density coupled with novel mapping function to visualize and mine cohesive subgraphs. We demonstrate the efficacy and efficiency of CSV on two real datasets. As an algorithm that executes in polynomial time, CSV can be useful as a tool for general exploration of a graph before a region of interest can be selected for more detailed analysis. Additionally we demonstrate that one can use the CSV plot as a pre- filtering method, to speed up the efficiency of clique mining algorithms such as CLAN by up to 80% while finding exactly the same set of cohesive subgraphs as the original algorithm (CLAN) does.

Acknowledgment: We will like to thank Zhiping Zeng and Jianyong Wang for making the source code and datasets for CLAN available to us. The development of CSV also benefited immensely from a synthetic graph generator provided by Xifeng Yan and Jiawei Han.

8. REFERENCES

[1] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. In *LATIN*, pages 598–612, 2002.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[3] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pages 49–60, Philadelphia, PA, June 1999.

[4] A.P.Gasch and M.B.Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-mean clustering. *Genome Biol.*, 3(RESEARCH0059), 2002.

[5] S. Asur, D. Ucar, and S. Parthasarathy. An ensemble framework for clustering proteinĺcprotein interaction networks. In *ISMB '07: Proceedings of the 15th*

Annual International Conference on Intelligent Systems, 2007.

[6] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proc. 2002 Int. Conf. Knowledge Discovery and Data Mining*, 2002.

[7] V. Boginski, S. Butenko, and P. M. Pardalos. Mining market data: a network approach. *Comput. Oper. Res.*, 33(11):3171–3184, 2006.

[8] B. Bollobas. *Extremal Graph Theory*. Dover Publications, Incorporated, 1978.

[9] B. J. Breitkreutz, C. Stark, and M. Tyers. Osprey: a network visualization system. *Genome Biol*, 4, 2003.

[10] B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recogn. Lett.*, 24(14):2357–2366, 2003.

[11] J. Cheriyan and R. Thurimella. *Fast Algorithms for k -Shredders and k -Node Connectivity Augmentation (Extended Abstract)*. 1996.

[12] G. Cong, K. Tan, A. Tung, and F. Pan. Mining Frequent Closed Patterns in Microarray Data. *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)-Volume 00*, pages 363–366, 2004.

[13] G. Cong, K. Tan, A. Tung, and X. Xu. Mining top-K covering rule groups for gene expression data. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 670–681, 2005.

[14] G. Cong, A. Tung, X. Xu, F. Pan, and J. Yang. FARMER: finding interesting rule groups in microarray datasets. *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 143–154, 2004.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, Portland, Oregon, Aug. 1996.

[16] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'95)*, pages 163–174, San Jose, CA, May 1995.

[17] H.Hu, X.Yan, Y.Huang, J.Han, and X.J.Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21(1):213–221, 2005.

[18] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *Proceedings of 29th International Conference on Very Large Data Bases*, 2003.

[19] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *Proceedings of 28th International Conference on Very Large Data Bases*, 2002.

[20] H. Hu, X. Yan, Y. Huang, J. Han, and X. Zhou. *Mining coherent dense subgraphs across massive biological networks for functional discovery*. 2005.

[21] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Mach. Learn.*, 50(3):321–354, 2003.

[22] G. Karypis and V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. In *Supercomputing '96: Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM)*, page 35, Washington, DC, USA, 1996. IEEE Computer Society.

[23] G. Kossinets and D. Watts. Empirical analysis of an evolving social network. *Science Magazine*, 311(5757), 2006.

[24] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[25] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[26] F. Pan, G. Cong, A. Tung, J. Yang, and M. Zaki. Carpenter: finding closed patterns in long biological datasets. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–642, 2003.

[27] F. Pan, A. Tung, G. Cong, and X. Xu. COBBLER: combining column and row enumeration for closed pattern discovery. *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 21–30, 2004.

[28] C. T. J. R. S. Filho, A. Traina and C. Faloutsos. Similarity search without tears: The omni family of all-purpose access methods. In *Proc. of the 17th IEEE Intl. Conf. on Data Engineering*, 2001.

[29] S. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

[30] A. Srinivasan. The value of strong inapproximability results for clique. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 144–152, New York, NY, USA, 2000. ACM.

[31] S. Tri and U. Leser. Gripp - indexing and querying graphs based on pre- and postorder numbering. Technical report, 2006.

[32] P. Turan. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

[33] J. Wang, Z. Zeng, and L. Zhou. Clan: An algorithm for mining closed cliques from large dense graph databases. In *Proceedings of the International Conference on Data Engineering*, page 73, 2006.

[34] X. Yan and J. Han. gspan: Graph-based substructure pattern mining, 2002.

[35] X. Yan, X. J. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *KDD*, pages 324–333, 2005.

[36] J. W. J. Z., Mellor and C. DeLisi. Visant: an online visualization and analysis tool for biological interaction data. *In BMC Bioinformatics*, 5:17–24, 2004.

[37] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *Proceedings of the International Conf. Knowledge Discovery and Data Mining*, pages 797–802, 2006.