Interactive Visualization of the Market Graph

Camilo Rostoker
University of British Columbia
Department of Computer Science
rostokec@cs.ubc.ca

December 20, 2005

Abstract

Financial markets are a fruitful area for data exploration, but the overwhelming size and dimension of the datasets usually prohibit meaningful analysis, especially on a large scale. Thus, there is a need for effective visualization tools to assist in efficiently exploring the data space. In this paper, we present a novel visualization tool that empowers a user with an interactive tool for finding meaningful relationships in historical or real-time financial market data. To reduce the size of data to be visualized, we summarize the areas of interest within the market graph by displaying only the pre-computed clusters, and aggregated inter- and intra-cluster edges. Target graph structures and their associated attributes are encoded using several visually intuitive schemes, and a modified force-directed model is used to layout the graph with minimal visual clutter while retaining important spatial properties. We also provide a brief overview of an underlying parallel data-mining pipeline which enables us to apply this visualization tool to real-time stock market data.

1 Introduction

In this work, we consider the financial stock market, with data describing stock price changes over a given time interval. Mining stock market data is not a new concept, but the algorithm we present is a relatively new approach to this problem. In industry, there are many readily-available financial analysis methods and tools, but the meaning of the particular results and the conclusions one can draw are dependent on the method. Here, we present a tool for visualizing subsets of stocks computed by heuristic methods that can efficiently find interesting patterns in large multi-dimensional stock market datasets. This type of interactive visualization tool is a suitable candidate for various financial analysis tasks such as portfolio management, exploratory analysis and real-time market analysis. While there are various system design options and algorithmic challenges to creating such a system, the visualization techniques used to make sense of the data are even more important. Thus, there is evident need for a powerful tool that can interactively and dynamically visualize the extracted knowledge in a meaningful way. In this paper we present a prototype system that can be used to extract and visualize meaningful information from the market graph. While the focus of this paper is on the visualization aspects, we include some background information on the related material that motivate this work.

The remainder of this paper is organized as follows: In Section 2, we briefly review some graph-theory basics. Section 3 presents a quick look at some recent graph-theoretic clustering techniques, while Sections 4 and 5 discuss related work on graph drawing and cluster visualization. Then, Section 6 presents the various components of our proposed visualization tool. Section 7 briefly introduces the enabling parallel framework, Section 8 discusses many issues with our current system and proposes interesting avenues for future work, and finally Section 9 concludes the paper.

2 Graph Theory Basics

In this section we briefly review some graph theory concepts.

Given an undirected graph $G = (V, E)$, where V is the vertex set of G, and E is the edge set of G.

The *complement graph of* $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(i, j) \mid i, j \in V,\ i \neq j \text{ and } (i, j) \notin E\}$.

For a subset $S \subseteq V$, we define $G(S) = (S, E \cap S \times S)$ *the subgraph induced by* S.

A graph $G = (V, E)$ is *complete* if $\forall i, j \in V,\ (i, j) \in E$. A *clique* C is a subset of V such that $G(C)$ is complete. The max-clique problem asks for a clique of maximum cardinality. In figure 1, vertices {3,4,5,6} form a maximum clique.

A *quasi-clique* QC is a *near* clique, meaning the induced subgraph graph $G(QC)$ is not complete, but missing only a small number of edges. More formally, a γ-quasi-clique is defined as a clique C such that $|E(C)| > \delta$, where $\delta = \gamma(\binom{|V(C)|}{2})$. For example, in figure 1, vertices {6,7,8,9} form a $(\frac{5}{6})$-quasi-clique, because there is only one out of six edges missing from the complete graph.

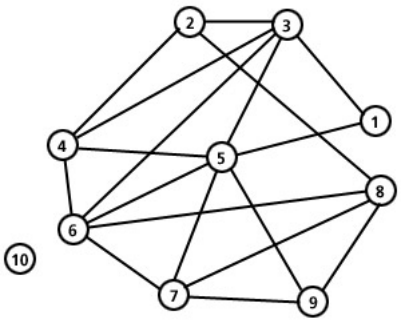

Figure 1. A graph with (quasi-)cliques and (quasi-) independent sets

An *independent set* $I \subset V$ is a set such that $\forall i, j \in V, (i, j) \notin E$, i.e. there is no edges between any of the vertices in I. In figure 1, vertices {1,2,7,10} form an independent set, because there are no edges between them. The maximum independent set problem asks for an independent set of maximum cardinality. It is known that the problem of finding the maximum independent set in G is equivalent to finding the maximum clique in $\overline{G}$.

A *quasi-independent set* is defined similarly as the *quasi-clique*, except in the complement graph $\overline{G}$. An example of a *quasi-independent set* in figure 1 are vertices {1,2,4,7,9,10}. Notice that these are a superset of the independent set listed previously.

3 Graph Clustering

Recently, several researchers have proposed novel graph-theoretic approaches to clustering. In [7], researchers have shown an interesting connection between clustering and finding all maximal cliques in a given undirected graph. Other similar approaches, for example [2], generalize the notion of defining clusters as maximal cliques to a definition that allows clusters to be identified as highly-dense, but not necessarily complete, subgraphs. Such *dense subgraphs* are often referred to as *quasi-cliques* if they have a minimum edge density.

Interestingly, a recent research paper confirmed the existence of small world properties in the stock market graph[4]. In a related paper, the market graph was shown to be a nearly-decomposable complex system evolving from hierarchic subsystems[10]. This approach defined clusters as weighted k-cores. The k-core of a graph is its maximal subgraph with vertices of degree at least k.

In this work, we use a state-of-the-art stochastic local search algorithm called Phased Local Search (PLS)[11] that finds maximum cliques and independent sets in the graph. Our motivation for this project is recent results showing that, in the context of financial market data, cliques represent groups of stocks that display similar trading patterns, while independent sets represent groups of stocks that are completely diversified from all others in the given dataset[4].

4 Graph Drawing

There is a plethora of graph drawing and visualization tools available for both academic research and commercial uses. Most of the publicly available tools are general graph drawing and layout packages, but all offer some advantages and disadvantages over

others. For example, Tulip[3] is a graph drawing tool with excellent scalability and powerful drawing routines, but the complexity/learning curve lends itself to more advanced users. Graphviz[6] is a toolkit designed for static layouts of small data, is easy to use and has been around for several years, but lacks the interaction and scalability of other graph drawing toolkits. Other systems offer specialized graph drawing features for the target application domain. Cytoscape[12], an active project that allows complex genes and proteins interaction networks to be visualized, and includes plug-ins for various related tasks. Large Graph Layout (LGL)[1] was designed for visualizing large-scale protein homology maps.

More recently, `prefuse`[9] was developed, a powerful toolkit for interactive visualization of graphs. While general in purpose, `prefuse` emphasizes the importance of seamless interaction and integration of multiple information visualization concepts including geometric pan/zoom, focus+context, semantic zoom, animations, and many other key features.

The same authors of `prefuse` recently released Vizster[8], an application for visualizing online social networks. Vizster utilizes many of `prefuse`'s built-in tools for interactive visualization, as well as a modified force-directed layout algorithm that effectively visualizes the entire social network showing clusters of friends, friends-of-friends, etc. This work followed from a related paper in which a novel force-directed layout was proposed for interactively exploring small world graphs[13].

5 Visualizing Graph Clusters

Recent approaches to visualizing clusters usually fall into one of two categories. The first approach is to compute the target structures (i.e. clusters) *a priori*, and then use an appropriate layout algorithm to visualize the results. The other more recent approach involves using a layout algorithm which is optimized to layout the nodes in the graph such that the clusters become visually apparent and important spatial properties are preserved. One example of this approach is Vizster, which uses a modified force-directed layout algorithm that automatically identifies and visualizes community structures in a large-scale online social network[8]. An example of how Vizster's layout routine visually identifies clusters can be seen in figure 2.

Figure 2. Vizster: Visualizing online social networks

6 Visualizing the Market Graph

In this section we present our solution to the market graph visualization problem. First we describe how individual clusters are visually encoded to represent (quasi-)cliques and (quasi-)independent sets. Then we describe how we use a force-directed layout algorithm to provide additional visual indicators of clustering results.

This work follows from Jeff Heer's recent Vizster[8] project by implementing several of his visualization techniques and utilizing `prefuse` as the toolkit for building the application. Moreover, the visualization approach we take is a combination of the two standard approaches mentioned in Section 5.

In a pre-processing step (refer to section 7 for details), we create *summaries* of the market graph, which includes the clusters and their intra- and inter-cluster edges. Each cluster is represented by a dummy node, which we call the *cluster root*, and currently serves only to connect cluster members. Figure 3 shows a typical cluster; the cluster root is labeled "Cluster 4".

The rational behind this approach is that for the

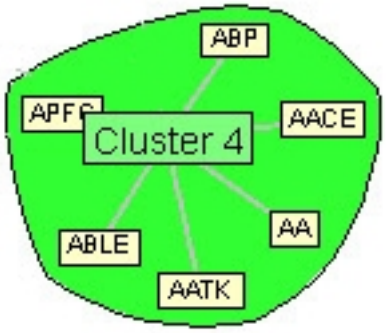

Figure 3. A typical cluster with its' cluster root

most part, our clusters are highly connected (in many cases fully-connected); thus, displaying all the individual edges within a cluster would cause severe visual clutter. The drawback is that individual edges between cluster members contain valuable information; in this particular case, they represent the correlation between the two stocks. In future work we plan on providing the ability to display these edges, but currently we do not, and argue this is acceptable because an approximate conclusion can be drawn knowing the cluster represents edges passing a certain correlation threshold.

Once the "summarized" graph has been created, we employ a force-directed layout algorithm to spatially position the clusters and their associated edges while providing visual cues as to the cluster types, correlation values (i.e. positive, negative or independent) and individual cluster member connectivity. The rest of this section provides full details on these methods.

6.1 Visually Encoding Clusters of Stocks

As discussed previously, the graphs to be visualized are dense clusters of vertices with edges connecting the clusters. Each dense cluster can either be a clique or a quasi-clique. The degree to which a cluster is a quasi-clique determines the intra-cluster strength. The goal of the visualization is to quickly identify these different clusters, the individual relationships within each cluster, and also to see the relation of stocks between clusters.

For each cluster, we draw a interpolated convex hull around the cluster to give it a sense of *encapsulation*. To avoid re-inventing the wheel, we utilize `prefuse`'s built-in convex hull function along with the interpolation code from Vizster. To show positive, negative and independent correlations, we use the familiar color scales of red for negative, green for positive, and a light grey for independent. We set the transparency of the convex hull proportional to the γ of the associated γ-quasi-clique.

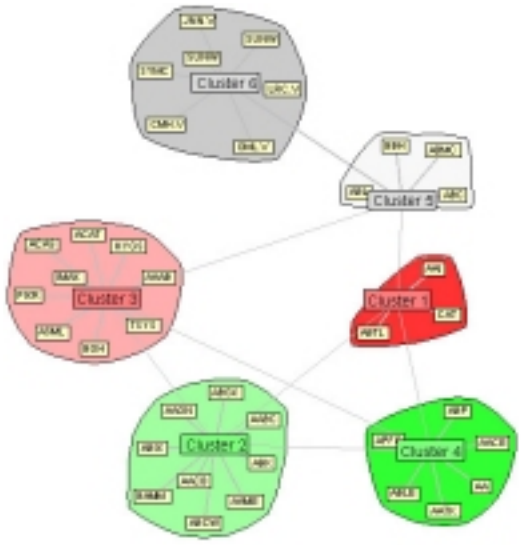

Figure 4. Visualizing different cluster types

For example, in figure 4, a fully transparent convex hull around Clusters 4 and 1 identifies them as positive and negative cliques, respectively, while Cluster 6 is easily identified an independent set. Clusters 2,3 and 5 are γ-quasi clusters, identified by the semi-transparent background.

One problem we have already identified is that the true γ is never known to the user (i.e. the actual degree of intra-cluster connectivity), nor are the edges between individual stocks in the cluster. Please refer to the Future Work section for a proposed solution to this problem.

6.2 Force-Directed Layout

To effectively visualize the subgraphs of the market graph, we use a force-directed layout algorithm. For simplicity, we use the default force-directed algorithm provided by the `prefuse` toolkit, and tweak the edge length and spring co-efficient functions to

optimize the layout, such that highly related clusters are close together, while independent clusters and minimally-related clusters are further apart.

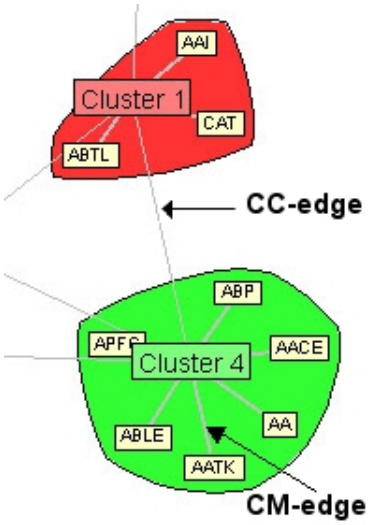

Figure 5. Illustrating CC-edges and CM-edges

In our visualization there are two types of visible edges: cluster-cluster edges (*CC-edges*) and cluster-member edges (*CM-edges*). Refer to figure 5 for an example of each edge type. We calculate both the edge length and tension according to edge and its associated cluster type.

CC-edges are parameterized by the number of inter-cluster edges; that is, edges between members in each cluster. This technique helps to show the *connectedness* of two clusters. CM-edges are parameterized depending on the type of cluster: in a clique-cluster, CM-edges do not need to be differentiated, and thus we simply parameterize the edge lengths using the cluster size, so that as clusters grow in size, their nodes can more easily settle around the cluster root with relatively little energy potential.

CM-edge within quasi-cliques are encoded with a value proportional to their intra-cluster connectively, giving each member a sense of *cluster dedication*, thus representing their individual contribution to the intra-cluster connectivity.

An example of the resulting layout is depicted in figure 6. As you can see, nodes $\{1,7\}$ assume a higher orbit, which indicates they are missing more intra-cluster edges than other members in the cluster. Nodes $\{4,5\}$ have the smallest orbit, and represent nodes with maximal intra-cluster connectivity. Figure 7 shows clusters representing cliques. The figure on the left is a small cluster and is represented by a uniform spread along an orbit. Note that this spread will never be perfect, since the nodes repel each each and thus will deviate off the orbit a small amount. However, by ensuring the distance between orbit levels is sufficiently large, nodes assigned to different orbits will be visually distinct.

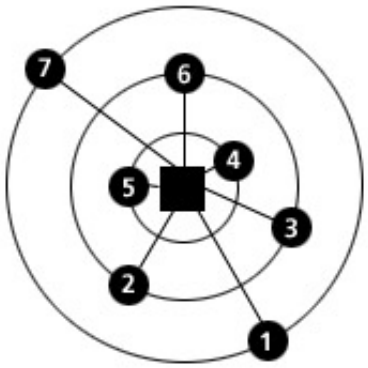

Figure 6. Quasi-Clique Cluster

A problem still exists, however, when the cluster size grows so large that all the nodes can't fit along the orbit. In this case, they will just spread out to a semi-random configuration that minimizes the local energy. This may cause confusion as they look similar to quasi-clique clusters, and although the two can be differentiated by the convex hull color-encoding schemes, we are currently seeking an alternate solution to this problem.

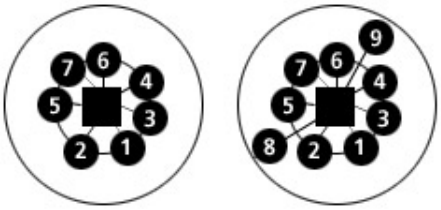

Figure 7. One good, one problematic clique cluster

To assign tension values, we use a constant *tight* tension for CM-edges so that they stay close to the cluster root. CC-edges are assigned tension values proportional to the number of inter-cluster links, which cause highly connected clusters to be closer, while less connected clusters move further away.

6.3 Interaction and Information Integration

Developing our tool was greatly simplified by `prefuse`[9], a powerful Java-based toolkit for interactive visualization of graphs. We utilized many of `prefuse`ś built-in features such as geometric pan and zoom, overview display for retaining global context, and animations for interpolating the color and size of display objects during transitions between states.

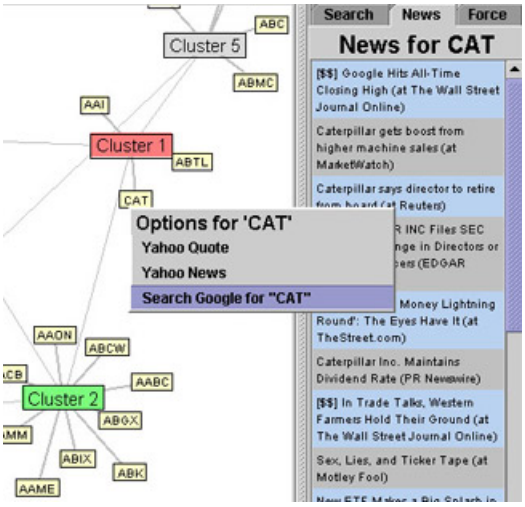

Figure 8. Node Context Menu

One advantage to working with stock market data is that information is readily available from a variety of sources. Exploiting this fact, a context menu provides several options for retrieving additional information (see figure 8. The current implementation supports on-demand retrieval of live news feeds for real-time stock news, aggregated from various sources using RSS technology. Other context menu options provide convenient access to a Google search or a detailed quote page for the stock under consideration.

6.4 Dynamic Graph Capabilities

As discussed earlier, a major goal of this system was to support the visualization of real-time market data. Thus, we have designed our system so that it is capable of performing dynamic graph layout as updates are received in real time. Upon startup, the visualization client connects to a remote update server, where updates are periodically received via a standard TCP socket connection. The updates contain commands to add, remove or replace nodes and edges.

As the underlying graph is updated, the visualization is updated accordingly to reflect the new graph. By using a force-model to layout the graph, new or updated nodes and edges are introduced into the display without having to recalculate the entire layout from scratch. Furthermore, we found that by *intelligently* placing new nodes close to their cluster roots, we are able to minimize the energy increase in the system caused by the updates.

The dynamic graph capabilities, and the associated data transfer protocol (see section 7), has been successfully tested using a mock graph update feed. Unfortunately, last-minute updates to this part of the system resulted in a bug, preventing this feature from being demonstrated during the class presentation.

7 A Real-time Parallel Data Processing Pipeline

In this section, we briefly describe a concurrent project of ours which provides the motivation for the visualization tool presented in this paper. The following scenario outlines the parallel framework which enables interactive visualization of clustering results from *real-time* data.

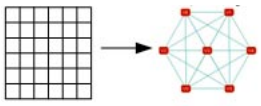

Figure 9. Creating the Graph

First, gather a number of samples of stock prices over a given time interval. We then convert this data set into a similarity matrix using Maronna, a cutting-edge parallel robust correlation method[5]. Then, the similarity matrix is converted to graph representation, where nodes represent stocks and an edge indicates that the similarity between two stocks exceeds a user-specified threshold.

In the next phase, we compute a large set of (quasi-) clique and (quasi-) independent sets, taking various user-defined scoring metrics and returning interest-

ing subsets of stocks. Finally, the results are visualized using a tool such as the one we are presenting in this paper.

While this type of analysis could be done manually by a market analyst, the scope of the analysis is limited to only a small number of stocks at once. Using a graph-based representation coupled with our efficient algorithms allows us to analyze thousands of stocks at once over a given time interval and get a global overview of the relationships between groups of stocks.

One of the key motivations for this approach is the use of a massively parallel framework, which exploits the speedups possible for both the robust correlation computation and the search for maximal cliques. Using this approach, we show how it is possible to perform this clustering analysis in real time, giving end-users (i.e. market analysts) a powerful tool for exploratory tasks such as detecting market trends, price interactions, and extracting a variety of other potentially useful information.

The design of the system will ultimately allow for multiple visualization clients to receive updates from the server. This parallel processing pipeline, and the associated graph update protocol which regulates the information flow between connected clients, has been successfully tested with offline data sources. Unfortunately, last-minute updates to the system resulted in a bug preventing this feature from being demonstrated during the class presentation.

8 Future Work

The system we have presented in this paper serves as good prototype for both an exploratory base-level analysis tool, but also as a front-end to a real-time system where specified relationships (i.e. the similarity metrics being used to construct the graph model) can be monitored and analyzed through the trading day. Perhaps the most important future work of this project is to more accurately determine what are interesting subsets of stocks. Currently, we visualize only the results of a crisp partitioning, with only a minimal amount of additional summary information. For the visualization to be most effective, the entire graph should be available to view, if desired, or filtered using a more appropriate graph clustering algorithm.

Below are several more issues we feel deserve considerable attention in the next round of development.

- **Handle overlapping clusters:** Currently, our clustering algorithm produces a crisp partitioning of the graph. Unfortunately, this is not an optimal approach for clustering in this context because the items being clustered (stocks), are often members of multiple clusters. Dealing with overlapping clusters, i.e., fuzzy clustering, presents challenges in its own in that clusters are no longer distinctly separated. In terms of the layout algorithm, this causes problems trying to visually illustrate the cluster boundaries, especially for clusters which contain many intersecting members. For example, consider figure 10, which is a clique with a single edge removed. This could be represented as a quasi-clique, but it could also be two maximal cliques (vertices $\{1,3,4,5\}$ or $\{2,3,4,5\}$).

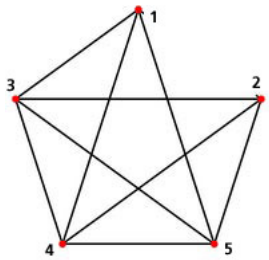

Figure 10. A clique missing a single edge

- **Encoding other variables**: Because there is so much more information than just the price interactions we use to calculate the initial market graph, we would like to encode other meaningful variables into the visualization. For example, node sizes could encode trade volume. Practically speaking, stocks with higher trading volume are considered to be more important as they affect more stocks and usually have much higher liquidity. Thus, encoding node sizes as a function of trading volume would allow an analyst to quickly spot these important stocks.

- **Including the complete underlying market graph**: While we specifically argue that

our approach to summarizing the results is an efficient way to transfer results from the processing components to the visualization system, we have also identified the need to have the entire market graph at the disposal of the user, even if it is not explicitly visualized. By doing this, much more information can be extracted by the user upon request. For example, even simply requesting the correlation values between a stock and all other stocks could easily be listed in a table format if the underlying weighted graph was available. Also, a user might want to explicitly request a certain stock or subset of stocks to be included in the visualization.

- **Other clustering methods besides partitioning via (quasi-)cliques and independent sets**: While this issue is not a visualization one on its own, we briefly mention it here is it provides much of the motivation to the entire visualization task. Currently, we perform a crisp partitioning of the items; in reality, items usually belong to several clusters. By utilizing a *fuzzy clustering* approach, the stocks can be simultaneously clustered into multiple clusters representing complex system interactions

- **Semantic zoom**: Currently, we do not provide any semantic zoom capabilities, but plan to include it in the next implementation. For example, semantic zoom is a great technique to quickly obtain detailed information on a single stock or cluster. For example, by one scenario we considered is that zooming in closer and closer to the stock would display its stock chart at increasing smaller time scales: from several years, to months, weeks, right up to a real-time intra-day trading chart. Other useful information, such as related news items, can be presented to the user upon reaching a specified zoom level.

- **Focus+Context**: The graph we currently work with is relatively small, and thus retaining global context has been easy. However, the full size of the dataset that we have contains over 7000 stocks, and in a real-time setting, the number of stocks could easily be double or triple that amount. In this case, retaining context while providing detailed information will become much more challenging. One solution is to implement some sort of distortion view, like a Fisheye view, such that detailed information can be obtained within a focal point.

9 Conclusion

In this paper we presented a visualization tool for exploring the stock market graph. Recent research findings and background material for this work was presented to motivate many of our algorithmic and design choices. Our visualization system is able to differentiate different cluster types by a simple yet intuitive color-encoding scheme. We employed a force-directed layout algorithm parameterized to visualize intra- and inter-cluster cluster relationships. Basic user interaction features for geometric pan/zoom were easily implemented using `prefuse`. We also provide several on-demand information integration features such as RSS-aggregated news items, detailed quotes and Google search. Furthermore, we have implemented an event-based dynamic graph system which can receive updates from a real-time data processing pipeline.

References

[1] Shannon Wieland Alex T. Adai, Shailesh V. Date and Edward M. Marcotte. Lgl: Creating a map of protein function with an algorithm for visualizing very large biological networks. *Journal of Molecular Biology*, 340(1):179–190, 2004.

[2] Javed A. Aslam, Ekaterina Pelekhov, and Daniela Rus. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications*, 8(1):95–121, 2004.

[3] D. Auber. Tulip. In P. Mutzel, M. Jünger, and S. Leipert, editors, *9th Symp. Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 335–337. Springer-Verlag, 2001.

[4] Vladimir Boginski, Sergiy Butenko, and Panos M. Pardalos. Mining market data: A network approach.

[5] James Chilson, Raymond Ng, Alan Wagner, and Ruben Zamar. Parallel computation of high dimensional robust correlation and covariance matrices. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 533–538, New York, NY, USA, 2004. ACM Press.

[6] Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Softw. Pract. Exper.*, 30(11):1203–1233, 2000.

[7] F. Glover, B. Alidaee, and H. Wang. Clustering of microarray data via clique partitioning. *Journal of Combinatorial Optimization*, 2005.

[8] Jeffrey Heer and Danah Boyd. Vizster: Visualizing online social networks. *InfoVis 2005 IEEE Symposium on Information Visualization*, 2005.

[9] Jeffrey Heer, Stuart K. Card, and James A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM Press.

[10] John Idicula. Highly interconnected subsystems of the stock market. *Working Paper*, 2005.

[11] Wayne Pullan. Phased local search. *Working Paper*, 2005.

[12] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, , and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[13] Frank van Ham and Jarke J. van Wijk. Interactive visualization of small world graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 199–206, Washington, DC, USA, 2004. IEEE Computer Society.