An End-User Oriented Graph-Based Visualization for Spreadsheets

Bennett Kankuzi
Department of Computer Science
University of Botswana
P/Bag 0704 Gaborone, Botswana
bfkankuzi@gmail.com

Yirsaw Ayalew
Department of Computer Science
University of Botswana
P/Bag 0704 Gaborone, Botswana
ayalew@mopipi.ub.bw

ABSTRACT

One of the difficulties in understanding and debugging spreadsheets is due to the invisibility of the data flow structure which is associated with cell formulas. In this paper, we present a spreadsheet visualization approach that is mainly based on the Markov Clustering (MCL) algorithm in an attempt to help spreadsheet users understand and debug their spreadsheets.

The MCL algorithm helps in visualizing large graphs by generating clusters of cells. In our visualization approach, we also use compound fisheye views and treemaps to help in the navigation of the generated clusters. Compound fish eye views help to view members of a particular cluster while showing their linkages with other clusters. Treemaps help to visualize the depth we are at while navigating a cluster tree. Our initial experiments show that graph-based spreadsheet visualization using the MCL algorithm generates clusters which match with the corresponding logical areas of a given spreadsheet. Our experiments also show that analysis of the clusters helps us to identify some errors in the spreadsheets.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation—*Spreadsheets*; D.2.5 [**Software Engineering**]: Testing & Debugging—*Debugging aids*

General Terms

Reliability

Keywords

spreadsheets, MCL algorithm, visualization, visual programming, end-user software engineering

1. INTRODUCTION

Spreadsheet systems are widely used and highly popular end-user software engineering environments. They are used for a variety of important tasks such as mathematical modeling, scientific computation, tabular and graphical data presentation, data analysis and decision making. Many business applications are based on the results of spreadsheet computations and as a result important decisions are made based on spreadsheet results. The provision of computational techniques that match users' tasks makes programming easier. The computational models of spreadsheets are adapted in areas such as information visualization, concurrent computation, user interface specifications to name just a few. There is also a trend in using the spreadsheet model as a general model for end-user programming [12]. The wide acceptance of spreadsheet systems is not only due to their simplicity but also due to their features which facilitate programming. The suppression of the low-level details of programming, the immediate visual feedback, and the availability of high-level task specific functions are commonly referred features among many others.

However, different authors have indicated some of the limitations of spreadsheet systems which may influence the quality of spreadsheets [8, 14, 17]. One commonly referred drawback is the invisibility of cell information (e.g., invisibility of formulas, documentation, etc.). While the values of cells are visible, other accompanying properties of a cell are not visible (except at explicit inspection). For example, the formulas which compute the values of cells are hidden. It is possible to see either the formulas or the values but not both at the same time. For a single cell, it is possible to see both but this does not give much information about the overall structure of the spreadsheet. In some cases, this locality to a single cell may help by narrowing the point of focus instead of dealing with the program as a whole, but it is also difficult to get sense of the general structure of the spreadsheet [8]. Nardi [13] described this problem by stating "*It is difficult to get a global sense of the structure of an individual formula that may have dependencies spread out all over the spreadsheet. Users have to track down individual cell dependencies one by one, tacking back and forth all over the spreadsheet.*" This also indicates the difficulty to predict how changes (deleting cells, modifying cell values and formulas, etc.) to one part of the spreadsheet will affect other parts. It is often difficult to identify where the data comes from and where it goes to unless one makes a detailed examination of the relationships.

The invisibility of the data flow (which represents the network structure of data-flow dependencies expressed by the

WEUSE IV, May 12, 2008, Leipzig, Germany.

references in the individual formulas) makes program understanding and debugging very difficult [2, 11, 15]. To address spreadsheet quality from a visualization perspective, the research works [4, 3, 5, 10, 15, 16] tried to examine the validity of spreadsheet programs by tracing the interactions among cells to uncover unintended or missing connections. They indicate irregularities or mismatches between the physical structure and the invisible dataflow structure. However, these approaches do not adequately handle visualization of large spreadsheets. In addition, the generated graphs do not match with logical areas in the corresponding spreadsheet.

This paper aims to provide a visualization approach that can simplify understanding and debugging of spreadsheets. The approach employed is based on Markov Clustering (MCL) algorithm, compound fish eye views, and Treemaps. The MCL algorithm helps in visualizing large graphs by generating clusters of cells. Compound fisheye views help to view members of a particular cluster while maintaining the context in relation to other clusters. Treemaps help to visualize the depth at a particular level while navigating a cluster tree.

2. GRAPH-BASED VISUALIZATION

Although spreadsheets are easy to develop, they are difficult to understand and debug. This is due to the fact that the spreadsheet display does not adequately show the cell dependencies. Information about cell dependencies is one of the most fundamental information that users need to know [5].

In our approach, we want to develop a spreadsheet visualization tool that achieves three main objectives. Firstly, we want to generate the computational data-flow graph of a given spreadsheet with nodes representing cells in the spreadsheet and edges representing dependencies between cells which can make the visualization (the generated dataflow graph) useful for spreadsheet debugging and maintenance. Secondly, we would like to deal with the problem of visualizing large graphs through graph clustering. Moreover, we would like to produce clusters that match with *logical areas* of the given spreadsheet. A logical area in a spreadsheet may be defined as a group of cells that from the spreadsheet user/programmer perspective should be grouped together due to the semantics of the spreadsheet. Thirdly, we would like to separate the graph-based visualization from the spreadsheet so as to avoid the problem of cluttering on the spreadsheet display (which introduces information overload) while maintaining the link (mapping) between the spreadsheet cells and graph nodes. To achieve the stated objectives, we developed a prototype tool whose conceptual architecture is depicted in Fig. 1.

For a given spreadsheet, a graph is generated by employing a parser and producing the resulting graph in the Graph Modeling Language (GML) [9] file format. For our demonstration purpose, we use Microsoft Excel and the parser has been developed using Microsoft Visual Basic for Applications (VBA) which iterates through all used formula cells to extract cell-dependency information. Cell-dependency information is extracted using the precedent cells of a given formula cell. The extracted dependency information is written to a text file in the GML file format while we are iterating through the spreadsheet. Thereafter, a Java-based graph-drawing program (i.e., Graphael) is invoked to display the resulting graph. Graphael is a Java program that visualizes large graphs using compound fisheye views and tree maps using different graph clustering algorithms [6].

We deal with the problem of visualizing large graphs [1, 7] through graph clustering. Graph clustering helps in viewing a manageable subset of nodes (cells) at a time. We employ the MCL algorithm [18, 19] to generate clusters and the generated clusters are displayed using a cluster tree, compound fisheye views and treemaps [1]. The MCL algorithm is a graph clustering algorithm that uses column stochastic (Markov) matrices to create clusters by simulating random walks through a graph. The first step in the algorithm is to associate a given input graph with some column stochastic matrix, M, such that entry M_{ij} indicates the probability of moving from node j to node i (in that order) in the input graph. Then two operations known as expansion and inflation are performed alternately starting with the associated stochastic matrix.

The process of expansion involves taking the power of a stochastic matrix using the normal matrix product; in this case, squaring the stochastic matrix. Inflation involves squaring each matrix entry in the matrix that results from the expansion operation. The process of alternating the expansion and inflation operations is repeated until a doubly idempotent stochastic matrix is produced. A doubly idempotent stochastic matrix does not change with further expansion or inflation operations. The resulting doubly idempotent stochastic matrix is a very sparse stochastic matrix which is associated with the input graph. The sparse stochastic matrix corresponds to the separation of the input graph into different connected components of the graph which are in turn interpreted as clusters.

We separate the visualization of the computational dataflow graph from the spreadsheet display to avoid information overload due to the superimposition of the graph on the spreadsheet display. However, we maintain the logical correspondence between the spreadsheet and the generated graph by displaying (at a time) cells in a graph cluster that match with logical areas in a given spreadsheet. The identified MCL clusters are highlighted/shaded in different colors in the original spreadsheet.

To enhance the logical correspondence between the spreadsheet and the generated graph, we dynamically generate the data-flow graph that is *tightly coupled* with the spreadsheet. To achieve this tight coupling, we have implemented the prototype tool in such a way that upon the click of a command button from a dropdown menu in the spreadsheet, the corresponding computational data-flow graph with clusters is generated in a separate window to the right of the spreadsheet window. A separate window below the cluster window is also presented to display the corresponding treemap. In our implementation, when the user navigates through the generated clusters, the corresponding cell members of a selected cluster are automatically highlighted in the original spreadsheet thus completing a two-way spreadsheet-visualization interaction. A screenshot of the prototype in action with a sample spreadsheet is given in Fig. 2.

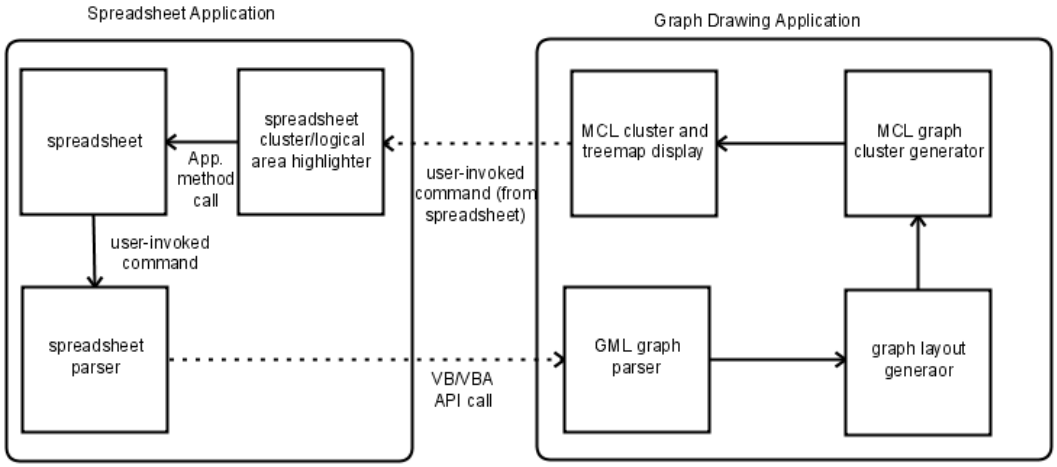

Figure 1: Conceptual architecture of the spreadsheet visualization

3. AN EXAMPLE

We now demonstrate how our prototype tool identifies and displays clusters in a given spreadsheet. Consider the spreadsheet given in Fig. 2. The spreadsheet is used to represent balance sheet statement of a company. Upon the click of a command button on a drop-down menu (see Fig. 2), a computational data flow graph for the spreadsheet is generated using the MCL clustering algorithm and the resulting clusters are displayed using compound fisheye views and treemaps (see Fig. 2).

As already stated above, we generate and display MCL clusters using the Graphael program. Fig. 2 shows the Graphael program displaying an MCL cluster (in top-right window) with cell members E9, E10, E11, E12, E13, E14, E15, E16 and E17. These cell members belong to a logical area which is defined by the formula E17=SUM(E9:E16). The formula view of the same spreadsheet is given in Fig. 3.

In the Graphael cluster window, members of a selected cluster are labeled while unselected clusters are represented by unlabeled nodes. Compound fish eye views help to view members of a particular cluster while showing their linkages with other clusters hence the large number of edge crossings in the cluster window. The bottom right Graphael window in Fig. 2 depicts a treemap for the cluster tree being navigated in the Graphael cluster window (top-right window). A treemap is an important complementary navigation aid becuase it not only helps to indicate the depth we are at when navigating a cluster tree but it also indicates the number of cells (nodes) which are in a selected cluster. For example, in Fig. 2, the treemap shows that the current cluster has nine cells which are E9, E10, E11, E12, E13, E14, E15, E16 and E17. The treemap also shows that the the cells we are are currently viewing are at depth 2 in the cluster tree (if we take the root to be at depth 0). We know this by counting the thickened borders in the treemap window.

Once the graph is displayed, the user may be interested in to see the logical area corresponding to the current cluster. This can be easily accomplished by clicking on an appropriate command button in the drop down menu in the spreadsheet interface. The MCL clusters will then automatically be highlighted in the spreadsheet. For example, in Fig. 2 cells E9, E10, E11, E12, E13, E14, E15, E16 and E17 have been highlighted since they belong to the same MCL cluster (and also same logical area). Upon demand from the user, all logical areas in the spreadsheet(corresponding to the different clusters) can also be automatically highlighted at the same time (not just once logical area at a time). This is achieved by using different cell background colors and cell border styles in the spreadsheet.

4. CONCLUSION

In this paper, we presented a prototype of a visualization tool that can simplify the task of debugging and understanding (and hence maintenance) of spreadsheets. In the approach presented, we tried to address three important aspects:

- A graph-based visualization that matches with the logical areas of a given spreadsheet
- A clustering algorithm to handle the visualization of large spreadsheets. This has been handled with MCL algorithm which is specifically developed to handle the visualization of large graphs.
- A separate display from the original spreadsheet. The main purpose of separating the graph from the spreadsheet is to avoid information overload on the user. An issue that can be raised is the difficulty of mapping between the spreadsheet and the graph. We handle this in two ways. Firstly, the tool shows the link between spreadsheet cells and the corresponding graph nodes using cell addresses. Secondly, upon demand the user is able to *regenerate* a correspond graph to the spreadsheet by simply clicking on an appropriate command button in a dropdown menu in the spreadsheet.

Our experiments with different spreadsheets show that MCL clusters match with logical areas in most cases. In some few cases, we have observed a discrepancy between cells in a cluster and cells in a logical area when a given cell belongs to different logical areas. A cell will belong to a cluster where

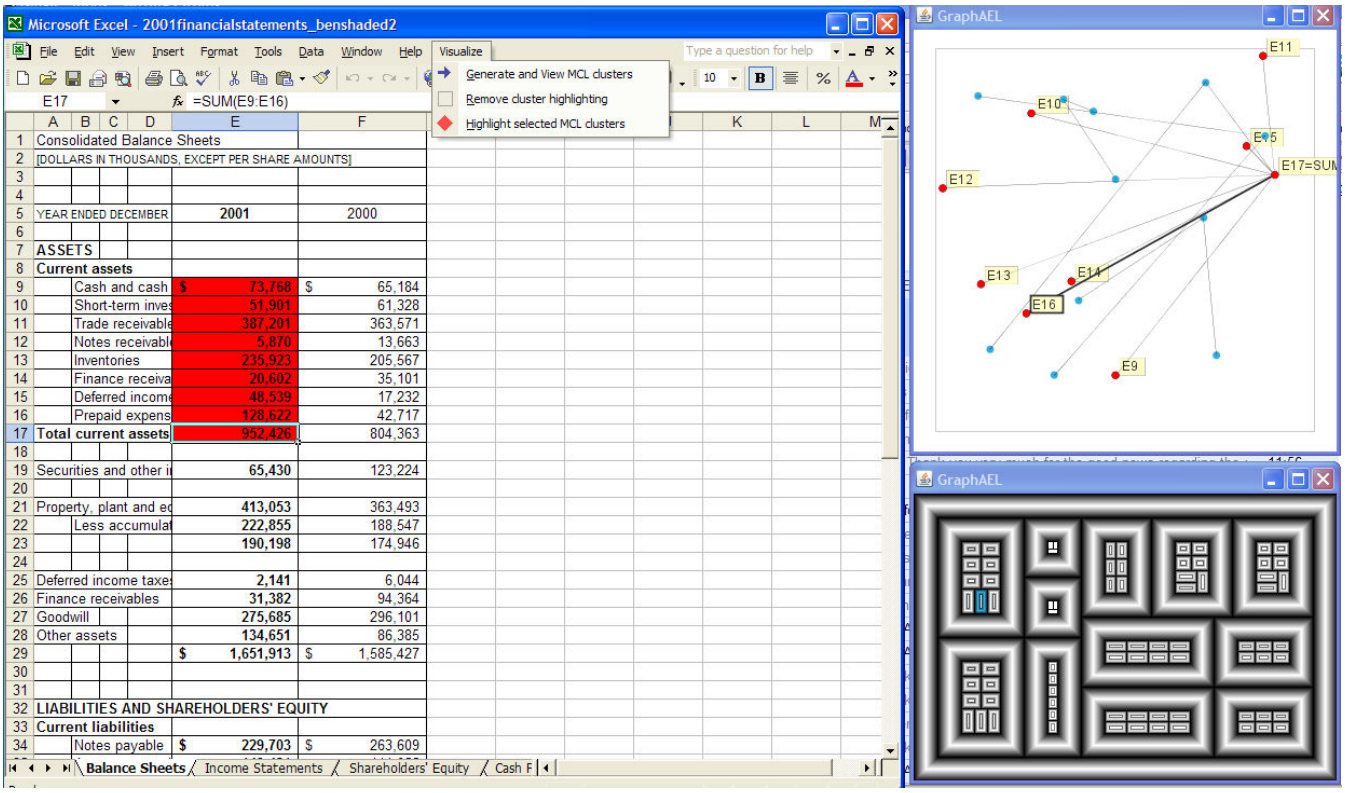

Figure 2: A screenshot of the prototype for the visualization with a "Balance Sheeet" spreadsheet, a cluster window (top-right window) and a treemap window (bottom-right window).

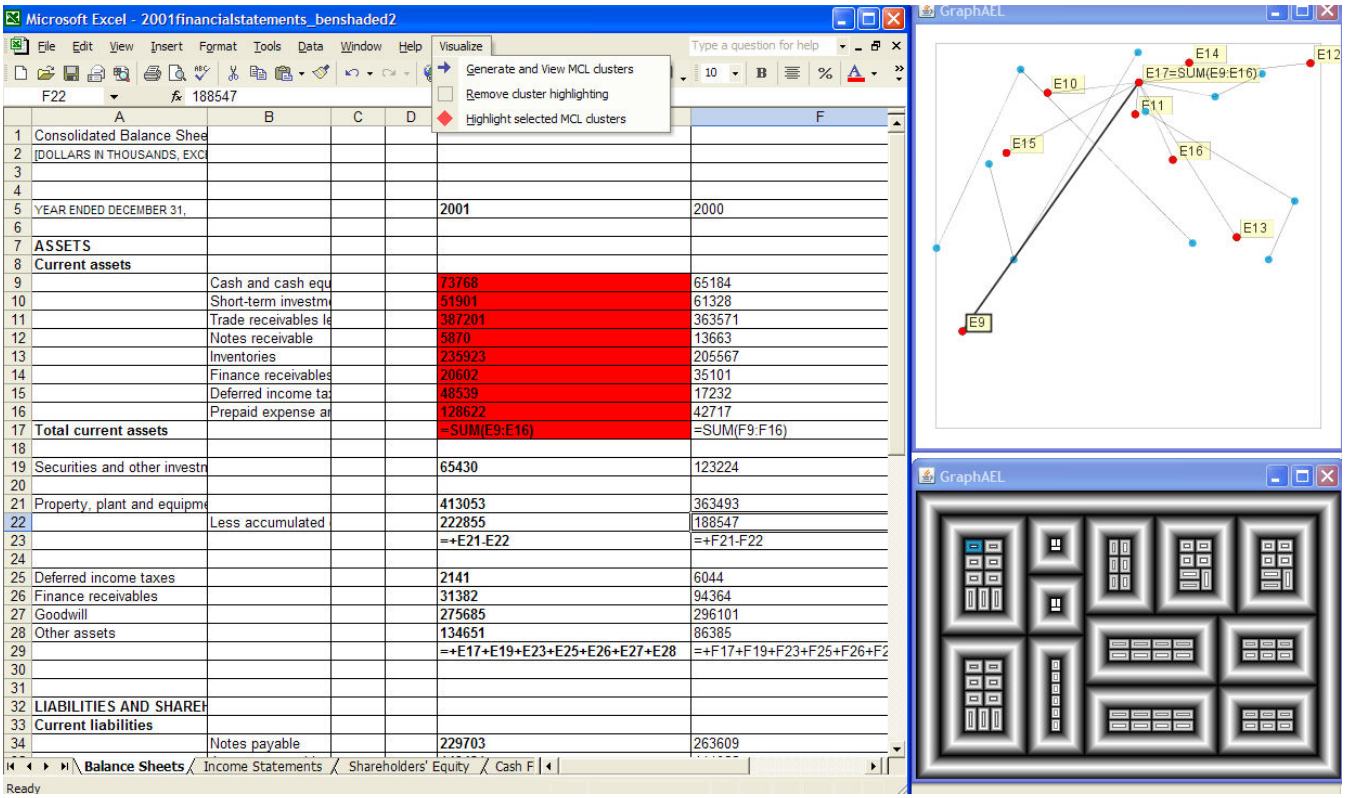

Figure 3: A screenshot of the prototype showing the formula view of the "Balance Sheet" spreadsheet.

there is a higher probability of being visited in a random walk as defined by the MCL algorithm. Our experiments also show that analysis of the clusters not only helps in understanding of the spreadsheets but also helps us to identify some errors in the spreadsheets hence helping in the spreadsheet debugging process. It is to be noted, however, that we haven't yet evaluated the effectiveness of our approach with actual spreadsheet users. One of our future works will focus on conducting trials of the tool with spreadsheet users to gauge the usefulness and usability of the tool.

5. REFERENCES

[1] J. Abello, S. G. Kobourov, and R. Yusufov. Visualizing Large Graphs with Compound-Fisheye Views and Treemaps. In *Proceedings of the 12th International Symposium on Graph Drawing*, pages 431–441, 2004.

[2] D. Ballinger, R. Biddle, and J. Noble. Spreadsheet structure inspection using low level access and visualisation. In *AUIC '03: Proceedings of the Fourth Australasian user interface conference on User interfaces 2003*, pages 91–94, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[3] M. Clermont. *A Scalable Approach to Spreadsheet Visualization*. PhD thesis, Universität Klagenfurt, Universitätsstrasse 65–67, A-9020 Klagenfurt, Austria, 2003.

[4] M. Clermont, C. Hanin, and R. T. Mittermeir. A Spreadsheet Tool Evaluated in an Industrial Context. In *Proceedings of the European Spreadsheet Risks Interest Group 2002 symposium*, 2002.

[5] J. S. Davis. Tools for Spreadsheet Auditing. *International Journal of Human-Computer Studies*, 45(4):429–442, 1996.

[6] Department of Computer Science - University of Arizona. Graphael home page. URL: `http://graphael.cs.arizona.edu/`.

[7] E. R. Gansner, Y. Koren, and S. C. North. Topological Fisheye Views for Visualizing Large Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, 2005.

[8] D. Hendry and T. Green. Creating, Comprehending and Explaining Spreadsheets: A Cognitive Interpretation of What Discretionary Users Think of the Spreadsheet Model. *International Journal of Human-Computer Studies*, 40(6):1033–1065, June 1994.

[9] M. Himsolt. GML: a portable Graph File Format. URL: `http://www.infosun.fim.uni-passau.de/Graphlet/GML/gml-tr.html`.

[10] T. Igarashi, J. D. Mackinlay, B. W. Chang, and P. T. Zellweger. Fluid Visualization of Spreadsheet Structures. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 118–125, 1998.

[11] R. Mittermeir and M. Clermont. Finding High-Level Structures in Spreadsheet Programs. In *WCRE '02: Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02)*, page 221, Washington, DC, USA, 2002. IEEE Computer Society.

[12] B. Nardi and J. Miller. *The Spreadsheet Interface: A Basis for End User Programming*. Hewlett-Packard, 1990.

[13] B. A. Nardi. *A small matter of programming: perspectives on end-user computing*. The MIT Press, 1993.

[14] J. Paine. Model Master: An Object-Oriented Spreadsheet Front-End. *Computer-Aided Learning using Technology in Economies and Business Education*, 1997.

[15] J. Sajaniemi. Modeling Spreadsheet Audit: A Rigorous Approach to Automatic Visualization. *Journal of Visual Languages and Computing*, 11(1):49–82, 2000.

[16] H. Shiozawa, K. Okada, and Y. Matsushita. 3D Interactive Visualization for Inter-Cell Dependencies of Spreadsheets. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, page 79, Washington, DC, USA, 1999. IEEE Computer Society.

[17] M. Tukiainen. ASSET: A Structured Spreadsheet Calculation System. *Machine-Mediated Learning*, 5:63–76, 1996.

[18] S. Van Dongen. MCL - a Cluster algorithm for Graphs. URL: `http://micans.org/mcl/`.

[19] S. Van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, Centre for Mathematics and Computer Science, University of Utrecht, The Netherlands, 2000.