An MCL algorithm based technique for comprehending spreadsheets

Bennett Kankuzi[1] and Yirsaw Ayalew[2]

[1] Department of Computer Science, University of Botswana
bfkankuzi@gmail.com
[2] Department of Computer Science, University of Botswana
ayalew@mopipi.ub.bw

Abstract. Spreadsheets are computer programs. However, as with programs written in traditional programming languages, it is very difficult to understand a spreadsheet created by others. This is because spreadsheet users normally view the superficial numerical (value) view of spreadsheets although computations are specified through "hidden" cell formulas. The cell formulas also define the data-flow structure of the spreadsheet. In this paper, we present a technique that highlights logical areas in spreadsheets which may act as a guide in understanding a spreadsheet. Instead of focussing their attention on the whole spreadsheet, spreadsheet users may narrow their focus to one logical area at a time. We identify logical areas in spreadsheets by using the MCL (Markov Clustering) algorithm on the underlying spreadsheet data-flow graph.

Keywords: POP-I.C.End-User (Office) Applications, POP-II.A.End-Users, POP-II.B.Program Comprehension, POP-II.B.Debugging, POP-III.B.Spreadsheets, POP-III.C.Visual Languages, POP-III.D.Visualization

1 Introduction

Spreadsheet systems are very popular among computer end-users. Computer end-users may be defined as people for whom conventional computer programming is not their main job although they use computers as part of their daily lives (Blackwell, 2002). End-users are not professional programmers although they might find themselves performing "programming" tasks such as creating spreadsheets. Spreadsheets are computer programs even though they might not be perceived to be so by their creators (Mittermeir & Clermont, 2002). The simplicity of the spreadsheet metaphor shields spreadsheet creators from the complexities of programming unlike in traditional programming. Despite this simplicity in creating spreadsheets, most spreadsheets contain errors (Panko, 2000; Panko & Sprague, 1998). Moreover, studies indicated that it is difficult to understand spreadsheets created by others due to the invisibility of the data dependency (data-flow) graph (Ballinger, Biddle, & Noble, 2003; Sajaniemi, 2000).

Spreadsheets have got three views namely the numerical (value) view, the formula view and the data-flow graph view. An illustration of the three views of a spreadsheet is given in Fig. 1. Normally, spreadsheet users look at the superficial numerical view. However, the key to understanding spreadsheets is to understand cell dependencies as defined through cell formulas (Davis, 1996). Sometimes spreadsheet users might use the formula view to understand a spreadsheet by analyzing individual cell formulas. However, this is tedious and cumbersome (Nardi, 1993). On the other hand, spreadsheet users may not be familiar with analyzing the underlying data-flow graph of a spreadsheet, since it may appear too complex to comprehend. In addition, spreadsheet users are used to working in an already familiar environment of the numeric view of a spreadsheet. Hence, in our effort to provide tools that assist in understanding spreadsheets, we opted to analyze data dependency graphs of spreadsheets and provide that information in the form of logical areas on the familiar spreadsheet view.

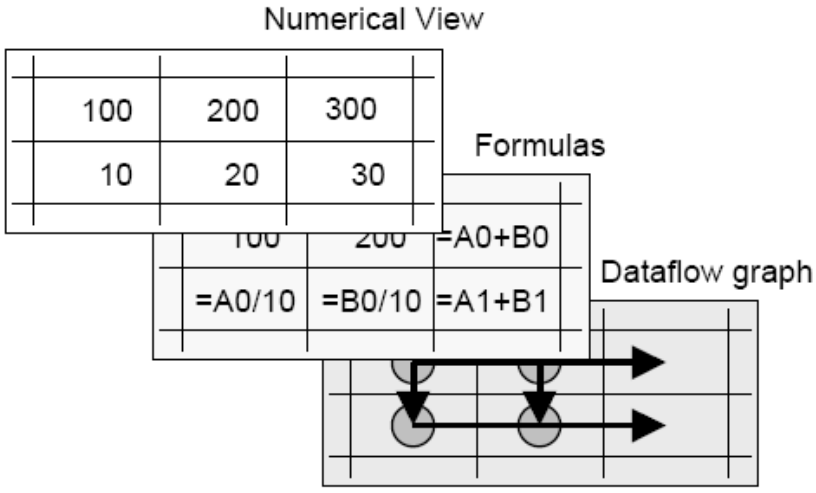


Fig. 1. An illustration of different views of a spreadsheet from Igarashi et al. (1998)

This paper presents a technique that highlights logical areas in spreadsheets which may act as a guide when understanding a spreadsheet by enhancing the detection of patterns in spreadsheets. A logical area in a spreadsheet may be defined as a group of cells which from the spreadsheet creator/user perspective form a logical unit due to the semantics of the spreadsheet (Mittermeir & Clermont, 2002). In other words, logical areas are an attempt to construct an abstract presentation of a given spreadsheet by grouping cells based on the similarity of their contents (Mittermeir & Clermont, 2002) . Logical areas in a spreadsheet are highlighted using different cell background colours. Instead of focussing their attention on the whole spreadsheet, spreadsheet users may narrow their focus to one logical area at a time. We identify the logical areas in spreadsheets by employing the MCL (Markov Clustering) algorithm (van Dongen, 2000b, 2000a) on the underlying data-flow graph of a given spreadsheet. We believe that for one to be able to debug a spreadsheet, one must understand the spreadsheet first, therefore we anticipate that our technique shall also be useful in the spreadsheet debugging process.

The rest of the paper is organized as follows: Section 2 presents related works. We present details of the MCL algorithm based technique in Section 3. An example on the applicability of the technique on a spreadsheet is presented in Section 4 while Sections 5 and 6 provide experiment results on the evaluation of the MCL algorithm as well as discussion on the same. We finally provide concluding remarks and future works in Section 7.

2 Related Work

Software visualization involves the provision of visual cues to enhance detection of patterns in software. Software visualization techniques provide visual representations of some aspects of a program, which could otherwise be difficult to notice by merely looking at “plain” source code of the program. Source code colouring is one software visualization technique that has also been exploited in the comprehension of source code in traditional programming languages like Java (Hendrix, James H. Cross, Maghsoodloo, & McKinney, 2000). For example, the jGRASP (James H. Cross & Hendrix, 2008) Integrated Development Environment (IDE) provides, among many of its features, syntax colouring of source code which is supposed to aid programmers in the comprehension of the code. In our case, we highlight generated MCL clusters in a spreadsheet using different cell background colours. There are also several research works that have exploited the use of visualization techniques to aid in the comprehension of spreadsheets.

Sajaniemi (2000) developed the S2 and S3 spreadsheet visualization tools in which logical areas or semantic units in a spreadsheet are highlighted and data-flow between logical areas is indicated through arrows. A screenshot of the S2 visualization tool is given in Fig. 2. The S3 visualization is a slight improvement to the S2 visualization. Highlighted areas in the visualization describe the plan structure of the spreadsheets and deviations from this structure show clearly in the visualization hence helping in the spreadsheet debugging process. We believe that superimposing data-flow arrows on the spreadsheet display introduces cluttering of the spreadsheet view, hence our technique does not indicate data-flow arrows on the spreadsheet display.

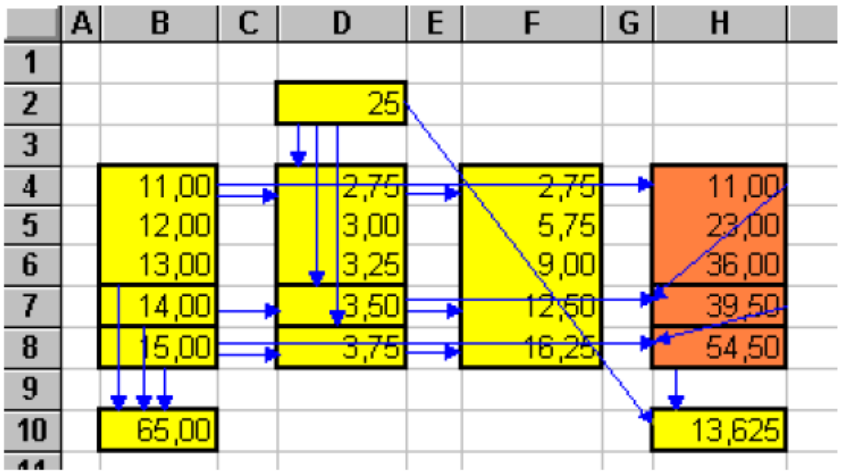


Fig. 2. A screenshot of the S2 visualization by Sajaniemi (2000)

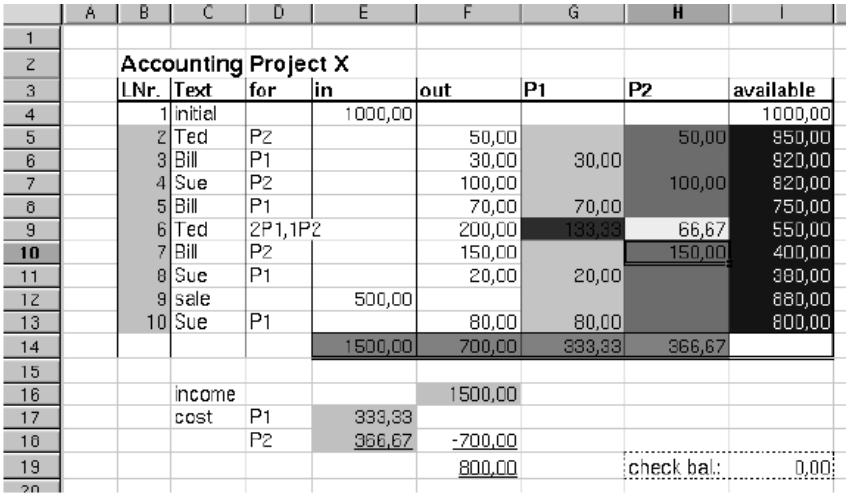


Fig. 3. A spreadsheet with highlighted logical areas (equivalence classes) by Mittermeir & Clermont (2002)

Igarashi, Mackinlay, Chang, and Zellweger (1998) also developed a spreadsheet visualization tool that depicts a fluid-like flow of data in a spreadsheet. The main emphasis in this visualization tool is the visualization of the hidden data-flow structure behind the tabular layout of a spreadsheet through fluid-like flow of data in the spreadsheet. Transient local views are used

to visualize data-flow structures associated with individual cells while it is possible to view the data-flow structure of the entire spreadsheet at once.

Mittermeir and Clermont (2002) developed a spreadsheet visualization toolkit that partitions a spreadsheet into logical areas known as *equivalence classes*. Equivalence classes are mainly based on structural similarity of formulas (Clermont, Hanin, & Mittermeir, 2002). Identified equivalence classes are then highlighted in the original spreadsheet as in Fig. 3. In our case, we highlight logical areas based on the cluster of cells that are generated by employing the MCL algorithm on the underlying spreadsheet data-flow graph.

3 An MCL (Markov Clustering) Algorithm Based Approach

Given a spreadsheet data-flow graph, with nodes representing cells and edges representing cell dependencies as defined by cell formulas, the need for graph clustering arises so that one should be able to view a manageable subset of the graph nodes at a time. Graph clustering is indeed important because normally spreadsheet data-flow graphs have a large number of nodes and as such the graph display becomes cluttered. However, we require that the cluster of cells which are displayed at a time should match with logical areas in the corresponding spreadsheet; otherwise, the clusters produced would be "meaningless".

The MCL algorithm (van Dongen, 2000b, 2000a) is one clustering algorithm that produces "natural" clusters from graphs. The algorithm finds clusters in graphs by simulating random walks on a given graph. This means that MCL clusters are characterized by graph nodes that are strongly interlinked together since the probability of a random walker in visiting more nodes in a cluster is high. The first step in the algorithm is to associate a given input graph with some column stochastic matrix, M, such that entry M_{ij} will indicate 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.

Due to its ability to find natural clusters in graphs, the MCL algorithm has also been used in many advanced applications. For example, the algorithm has been reliably used in the assignment of proteins into families based on precomputed sequence similarity information (Enright, Van Dongen, & Ouzounis, 2002). Our experiments show that the clusters which the MCL algorithm produces match with logical areas in spreadsheets.

We have already implemented a prototype of the spreadsheet visualization tool using the Microsoft Excel spreadsheet system in conjunction with the Graphael (Forrester, Kobourov, Navabi, Wampler, & Yee, 2004) graph drawing software. We used Microsoft Excel because it is a popular spreadsheet system. On the other hand, we also used the Graphael graph drawing software because it has an implementation of the MCL algorithm. A screenshot of the spreadsheet visualization tool with a spreadsheet side by side with the Graphael program is depicted in Fig.

4. A detailed description of the conceptual architecture of the visualization tool can be found in Kankuzi and Ayalew (2008) (Kankuzi & Ayalew, 2008b).

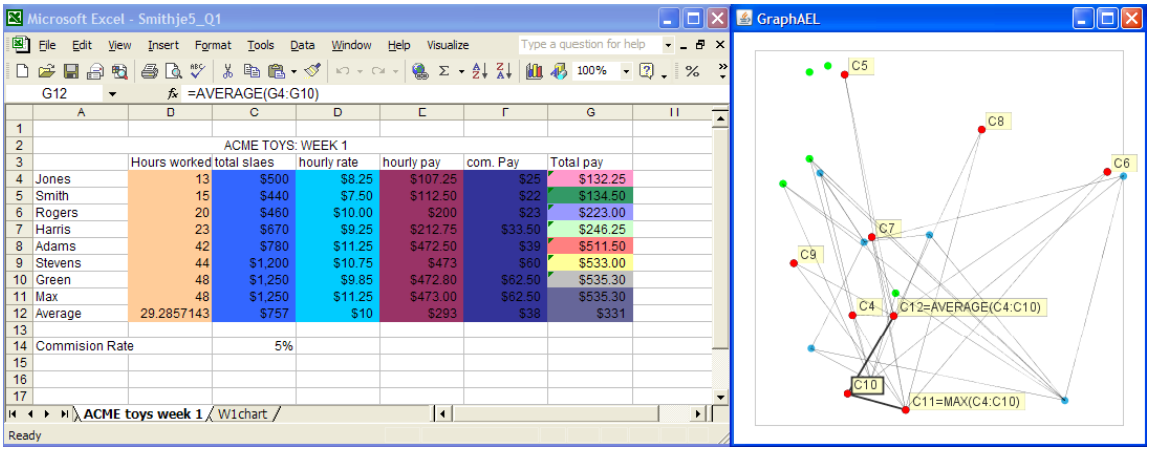
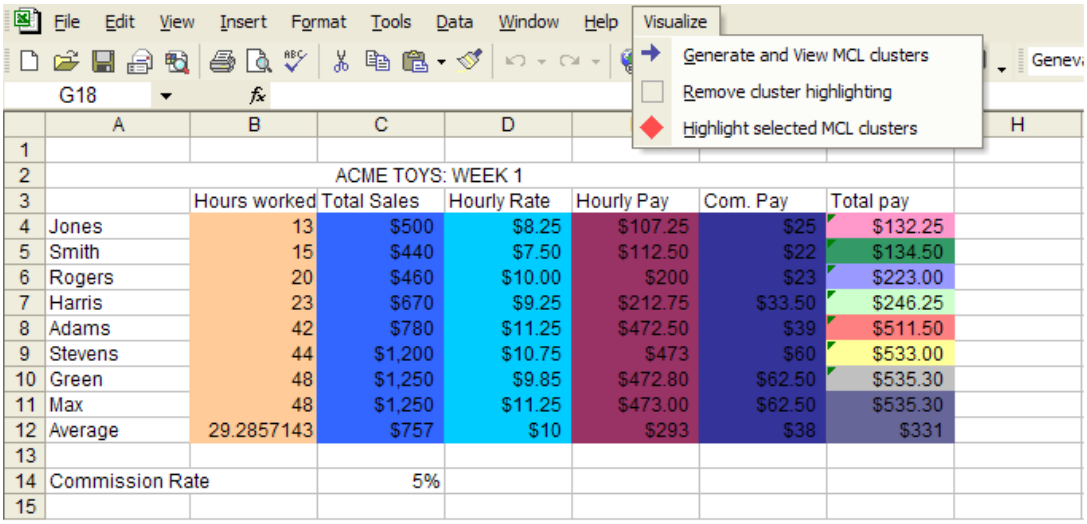


Fig. 4. A screenshot of the spreadsheet visualization with a spreadsheet side by side with the Graphael program

Upon the click of a command button on the "Visualize" drop-down menu in the spreadsheet window menu bar, clusters from a corresponding data flow graph for the spreadsheet are generated and displayed in a Graphael program window side by side with the spreadsheet. The clusters are represented by nodes. A user then navigates through the cluster nodes by clicking on the nodes using either the left or right mouse buttons. To see a logical area corresponding to a selected cluster in the Graphael window, the user clicks on an appropriate command button in the "Visualize" drop down menu in the spreadsheet window menu bar. The logical area is then automatically highlighted in the spreadsheet. This process is then repeated until all clusters in the Graphael window have been selected. Different logical areas in the spreadsheet are highlighted using different cell background colours since the colour codes are randomly generated. A detailed description on how clusters are generated and how one navigates through the clusters is given in (Kankuzi & Ayalew, 2008a, 2008b). In this paper, we focus on the highlighted logical areas as a spreadsheet comprehension aid.

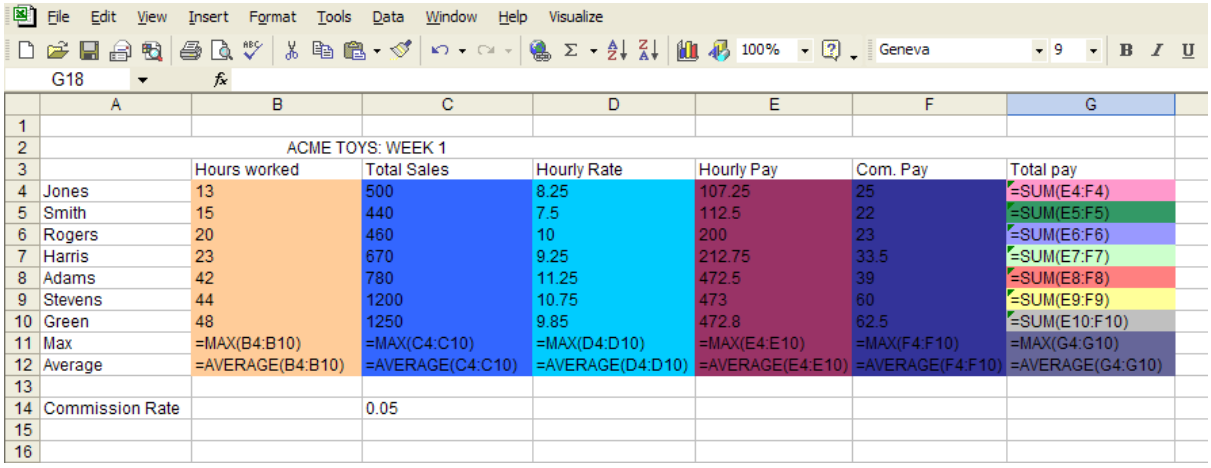
4 Example

Consider the spreadsheet given in Fig. 5. The spreadsheet was sourced from the EUSES spreadsheet corpus (Fisher & Rothermel, 2005). The spreadsheet is used to calculate payments to sales workers of a certain toy manufacturing company. The payments are based on hours worked, total sales and commission on sales. Using appropriate commands from the "Visualize" menu in the spreadsheet menu bar as depicted in Fig. 5, we generate MCL clusters from the underlying spreadsheet data-flow graph. The generated MCL clusters are then highlighted with different cell background colours in the spreadsheet as in Fig. 5. We have deliberately emphasized the highlighted MCL clusters by formatting cluster borders using different border styles. For example, cell range B4:B12 is highlighted as one MCL cluster. We also use the formula view of the spreadsheet in this example just to verify whether the clusters generated by the MCL algorithm conform to what is also defined by formulas. A look at the formula view of the spreadsheet in Fig. 6 confirms that indeed cells in cell range B4:B12 are in the same logical area. Similarly for clusters in columns C, D, E, and F. We also observe by using the formula view of the spreadsheet that calculations for "Hourly Pay" were not performed using formulas.



	A	B	C	D	E	F	G	H
1								
2	ACME TOYS: WEEK 1							
3		Hours worked	Total Sales	Hourly Rate	Hourly Pay	Com. Pay	Total pay	
4	Jones	13	$500	$8.25	$107.25	$25	$132.25	
5	Smith	15	$440	$7.50	$112.50	$22	$134.50	
6	Rogers	20	$460	$10.00	$200	$23	$223.00	
7	Harris	23	$670	$9.25	$212.75	$33.50	$246.25	
8	Adams	42	$780	$11.25	$472.50	$39	$511.50	
9	Stevens	44	$1,200	$10.75	$473	$60	$533.00	
10	Green	48	$1,250	$9.85	$472.80	$62.50	$535.30	
11	Max	48	$1,250	$11.25	$473.00	$62.50	$535.30	
12	Average	29.2857143	$757	$10	$293	$38	$331	
13								
14	Commission Rate		5%					
15								

Fig. 5. Numeric view of a sample spreadsheet with highlighted MCL clusters



	A	B	C	D	E	F	G
1							
2	ACME TOYS: WEEK 1						
3		Hours worked	Total Sales	Hourly Rate	Hourly Pay	Com. Pay	Total pay
4	Jones	13	500	8.25	107.25	25	=SUM(E4:F4)
5	Smith	15	440	7.5	112.5	22	=SUM(E5:F5)
6	Rogers	20	460	10	200	23	=SUM(E6:F6)
7	Harris	23	670	9.25	212.75	33.5	=SUM(E7:F7)
8	Adams	42	780	11.25	472.5	39	=SUM(E8:F8)
9	Stevens	44	1200	10.75	473	60	=SUM(E9:F9)
10	Green	48	1250	9.85	472.8	62.5	=SUM(E10:F10)
11	Max	=MAX(B4:B10)	=MAX(C4:C10)	=MAX(D4:D10)	=MAX(E4:E10)	=MAX(F4:F10)	=MAX(G4:G10)
12	Average	=AVERAGE(B4:B10)	=AVERAGE(C4:C10)	=AVERAGE(D4:D10)	=AVERAGE(E4:E10)	=AVERAGE(F4:F10)	=AVERAGE(G4:G10)
13							
14	Commission Rate		0.05				
15							
16							

Fig. 6. Formula view of the spreadsheet given in Fig. 5

"Hourly Pay" was supposed to be calculated from "Hours worked" and "Hourly Rate". We suspect that the calculations were entered manually. We note that this is prone to errors. We also observe that commission pay ("Com. Pay") in column F was also calculated manually. Hence, we see that commission rate as specified in cell C14 does not belong to any highlighted cluster. This is not an error on the part of the MCL algorithm because cell C14 is correctly not part of the spreadsheet data-flow graph due to the fact that it is not part of any cell formula in the spreadsheet. A cell which is part of any cell formula becomes part of the data-flow graph hence the cell becomes part of the input to the MCL algorithm. This is not the case with cell C14.

We also take note that there are seven single-membered clusters namely; G4, G5, G6, G7, G8, G9 and G10. These clusters match with their respective logical areas. For example, cluster with cell G4 is total pay for worker "Jones" in row 4 which is calculated from "hourly pay" and "com. Pay". A similar explanation goes for the other single-membered clusters. We also take note that a cell may belong to more than one logical area but it can belong to *only one* MCL cluster at a time. For example, cell F4 could have also belonged to a logical area which is defined by the formula G4=SUM(E4:F4) (i.e. logical area with cells E4, F4 and G4). But instead of being put in this row-wise logical area, the MCL algorithm has put the cell as part of the columwise logical area in column F. This is something that we cannot have control of as

it is up to the MCL algorithm to place a cell in a cluster where it computes that the cell has a higher probability of being visited in a random walk. We acknowledge that this is one of the limitations of the MCL algorithm since a cell may sometimes correctly belong to more than one cluster. In our case, we are mainly interested in whether the MCL algorithm has placed a cell in at least one correct logical area or not.

We also observe that there is one MCL cluster that does not necessarily match with our anticipated logical area. The double-membered cluster with cells G11 and G12 would have been better if it was split into single-membered clusters as it is the case with the other clusters in column G. However, we still take note that it is not an error on the part of the MCL algorithm to put the two cells in one cluster since they are related by the fact that they have the same precedent range (i.e. G4:G10) through formulas G11=MAX(G4:G10) and G12=AVERAGE(G4:G10).

5 Evaluation of the MCL Algorithm

As with the spreadsheet in Fig. 5, we proceeded to generate MCL clusters for seven more spreadsheets (i.e. we used eight spreadsheets including the one used in the previous example). All the spreadsheets used were randomly selected from the EUSES spreadsheet corpus (Fisher & Rothermel, 2005). The main aim of the experiment was to find how well the MCL algorithm produces clusters that match with logical areas of a given spreadsheet. This is important because we expect that a clustering algorithm has to produce "meaningful" clusters i.e. each cluster of cells produced should match with what the user anticipates to be a logical area on a spreadsheet. We use the *success rate* as a measure of performance of the MCL algorithm. We define the success rate of the MCL algorithm in terms of the number of all generated MCL clusters in a given spreadsheet and the number of those generated MCL clusters that entirely match with respective logical areas in the spreadsheet. For example, for the spreadsheet in Fig. 5 (the one used in the previous example), the total number of highlighted MCL clusters is 13. However, 12 of them exactly match with our anticipation of logical area. Therefore the success rate of the MCL algorithm on this spreadsheet is 12/13 which translates to 92.3%. We present results of the experiment in Table 1.

6 Discussion

Referring to Table 1, we report a number of observations on the experiment. The success rate of the MCL algorithm for spreadsheets 2, 4, 5 and 6 was 100% in each individual spreadsheet. This means that in all these spreadsheets, the MCL clusters produced matched with what we perceived to be logical areas.

We also observed that for spreadsheet 3, there was one cluster that seemed not to obviously match with a logical area because the cells were not in a contiguous group (neither row-wise nor column-wise). However, a closer inspection of the cell formulas revealed that indeed there were cell dependencies among the cells. This is a case where a logical area might not seem to be obvious to the one comprehending the spreadsheet. We also observed that there were about fifteen numeric cells in spreadsheet 3 that were not part of any highlighted MCL cluster. Left-out cells are not highlighted since they are not part of the spreadsheet data-flow graph. Our inspection also showed that they were indeed not part of any cell formula. We suspect this to be an error on the part of the spreadsheet creator or may be the cells were just used for documentation purposes.

For spreadsheet 4, we observed that although the MCL algorithm perfectly found clusters that

Table 1. Experiment results on the evaluation of the MCL algorithm

Spreadsheet no.	**Description**	**Total no. of MCL clusters**	**No. of matching clusters**	**Success Rate**
1.	Calculation of payments to sales workers based on hours worked, total sales and commission	13	12	92.3%
2.	Employee payroll based on hours worked	17	17	100%
3.	Consolidated statements of cash inflows	14	13	92.9%
4.	Statistics on land restitution claims	12	12	100%
5.	Analysis of a financial plan for a housing scheme	5	5	100%
6.	financial highlights of for some company	82	82	100%
7.	balance sheet for some funds	44	40	90.9%
8.	Statement of financial performance	52	48	92.3%
			Average Success Rate	**96.05%**

matched with logical areas, there were also six numeric cells which were not part of any highlighted cluster. Our inspection showed that the numeric values in these cells were entered manually and not through cell formula calculations. We suspect that the spreadsheet creator might have done this unknowingly as performing calculations manually in spreadsheets is error-prone.

We also noted that for spreadsheet 5, some of the highlighted MCL clusters were including blank cells. Our inspection showed that the spreadsheet creator used cell formulas that were referencing to cell ranges that included blank cells. Instead of refering to specific cells that had to be included in a cell formula, the spreadsheet creator was just including all cells in a particular range. This is also error-prone.

We also observed that for spreadsheet 7, there were four MCL clusters that did not match with what we anticipated to be logical areas. Closer inspection of the cell formulas revealed that the spreadsheet creator had performed erroneous calculations in a number of cells by referencing to blank cells in some cell formulas.

Spreadsheet 8 included four clusters that did not match with anticipated logical areas on the spreadsheet. We observed that for two clusters, the spreadsheet creator included text cells in cell formulas. This led to the cells to be included in the spreadsheet data-flow graphs. There were also numeric cells which were unreferenced. As such, they appeared to be isolated among a group of highlighted cells. This led to two more clusters which did not match with logical areas.

The average success rate of the MCL algorithm on the eight spreadsheets used in the experiment is 96.05%. Based on the results of this experiment with success rate as a performance measure, we conclude that the MCL algorithm performs satisfactorily. However, we take note

that determining whether an MCL cluster matches with a logical area or not, depends on the judgement of the one comprehendng the spreadsheet. This might affect evaluation results of the MCL algorithm. To improve on the reliability of the approach, we propose that a spreadsheet should be independently evaluated by more than one spreadsheet user and thereafter compare the evaluation results. We also take note that our experiment was conducted on relatively smaller-sized spreadsheets. However, we take note that the spreadsheets were actually used in business enterprises. And as such, we still found them to be appropriate for this experiment.

7 Conclusion and Future Work

In this paper, we have presented a novel technique which can aid in the spreadsheet comprehension process. The technique involves the use of the MCL algorithm to find clusters in a spreadsheet data-flow graph. Identified clusters are then highlighted in the corresponding spreadsheet. We have also presented experiment results on the applicability of the technique on a number of spreadsheets. We plan to conduct trials on the usability of the technique on professionals who use spreadsheets in their daily work. This will be one of our future works.

References

Ballinger, D., Biddle, R., & Noble, J. (2003). Spreadsheet Structure Inspection Using Low Level Access and Visualisation. In *Proceedings of the Fourth Australasian Conference on User Interfaces* (pp. 91–94). Darlinghurst, Australia: Australian Computer Society, Inc.

Blackwell, A. F. (2002). What is Programming? In J. Kuljis, L. Baldwin, & R. Scoble (Eds.), *Proceedings of the 14th Workshop of the Psychology of Programming Interest Group* (pp. 204–218). London, UK: PPIG.

Clermont, M., Hanin, C., & Mittermeir, R. T. (2002). A Spreadsheet Tool Evaluated in an Industrial Context. In *Proceedings of the 3rd European Spreadsheet Risks Interest Group Symposium.* Cardiff, Wales.

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

Enright, A. J., Van Dongen, S., & Ouzounis, C. A. (2002). An Efficient Algorithm for Large-Scale Detection of Protein Families. *Nucleic Acids Research*, *30*(7), 1575–1584.

Fisher, M., & Rothermel, G. (2005). The EUSES Spreadsheet Corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. *SIGSOFT Software Engineering Notes*, *30*(4), 1–5.

Forrester, D., Kobourov, S. G., Navabi, A., Wampler, K., & Yee, G. V. (2004). Graphael: A System for Generalized Force-Directed Layouts. In J. Pach (Ed.), *Graph drawing* (Vol. 3383, p. 454-464). Springer.

Hendrix, T. D., James H. Cross, I., Maghsoodloo, S., & McKinney, M. L. (2000). Do Visualizations Improve Program Comprehensibility? Experiments with Control Structure Diagrams for Java. *SIGCSE Bull.*, *32*(1), 382–386.

Igarashi, T., Mackinlay, J. D., Chang, B. W., & Zellweger, P. T. (1998). Fluid Visualization of Spreadsheet Structures. In *Proceedings of the IEEE Symposium on Visual Languages* (pp. 118–125).

James H. Cross, I., & Hendrix, T. D. (2008). jGRASP: An Integrated Development Environment with Visualizations for Teaching Java in CS1, CS2, and beyond. *Journal of Computing Sciences in Colleges*, *23*(3), 169–171.

Kankuzi, B., & Ayalew, Y. (2008a). A Dynamic Graph-Based Visualization for Spreadsheets. In *Proceedings of the 3rd IASTED Conference on Human-Computer Interaction* (pp. 198–203). Innsbruck, Austria.

Kankuzi, B., & Ayalew, Y. (2008b). A User-Centered Graph-Based Visualization for Spreadsheets. In *Proceedings of the 4th International Workshop on End-User Software Engineering (WEUSE '08)* (pp. 86–90). Leipzig, Germany: ACM Press.

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

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

Panko, R. R. (2000). Spreadsheet Errors: What We Know. What We Think We Can Do. In *Proceedings of the european spreadsheet risk interest group symposium.*

Panko, R. R., & Sprague, R. H. (1998). Hitting the Wall: Errors in Developing and Code Testing a Simple Spreadsheet Model. *Decision Support Systems*, *22*(4).

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

van Dongen, S. (2000a). *Graph Clustering by Flow Simulation.* Unpublished doctoral dissertation, Centre for Mathematics and Computer Science, University of Utrecht, The Netherlands.

van Dongen, S. (2000b). *MCL - an algorithm for clustering graphs.* (URL: `http://micans.org/mcl/`, Access date: 1st August, 2007)