Visualizing Spreadsheet Formula Graphs Compactly

Fanchao Chen
Fudan University
chenfc18@fudan.edu.cn

Dixin Tang
UC Berkeley
totemtang@berkeley.edu

Haotian Li
HKUST
haotian.li@connect.ust.hk

Aditya G. Parameswaran
UC Berkeley
adityagp@berkeley.edu

ABSTRACT

Spreadsheets are a ubiquitous data analysis tool, empowering non-programmers and programmers alike to easily express their computations by writing formulae alongside data. The dependencies created by formulae are tracked as formula graphs, which play a central role in many spreadsheet applications and are critical to the interactivity and usability of spreadsheet systems. Unfortunately, as formula graphs become large and complex, it becomes harder for end-users to make sense of formula graphs and trace the dependents or precedents of cells to check the accuracy of individual formulae and identify sources of errors. In this paper, we demonstrate a spreadsheet formula graph visualization tool, TACO-Lens, developed as a plugin for Microsoft Excel. Our plugin leverages TACO, our framework for compactly and efficiently representing formula graphs. TACO compresses formula graphs using a key spreadsheet property: tabular locality, which means that cells close to each other are likely to have similar formula structures. This compact representation enables end-users to more easily consume complex dependencies and reduces the response time for tracing dependents and precedents. TACO-Lens, our visualization plugin, depicts the compact representation of TACO and supports users in visually tracing dependents and precedents. In this demonstration, attendees can compare the visualizations of different formula graphs using TACO, Excel's built-in dependency tracing tool, and an approach that does not compress formula graphs, and quantitatively compare the different response time of different approaches.

PVLDB Reference Format:
Fanchao Chen, Dixin Tang, Haotian Li, and Aditya G. Parameswaran. Visualizing Spreadsheet Formula Graphs Compactly. PVLDB, 16(12): 4030-4033, 2023.
doi:10.14778/3611540.3611613

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/taco-org/tacolens.

1 INTRODUCTION

Spreadsheet systems are unquestionably the most popular interactive data analysis tools on the planet, with a user base of around 1 Billion [6]. A major reason for their popularity is because they empower non-programmers and programmers alike to express sophisticated computation by embedding formulae along with data.

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611613

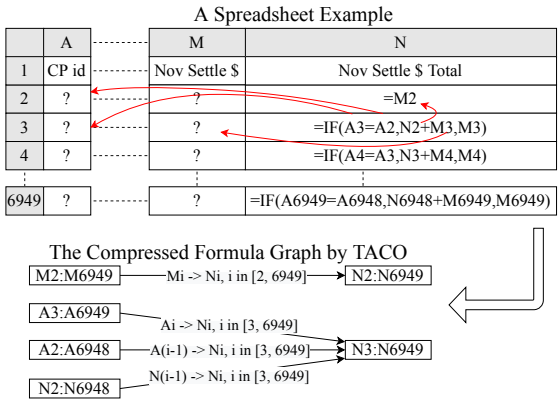
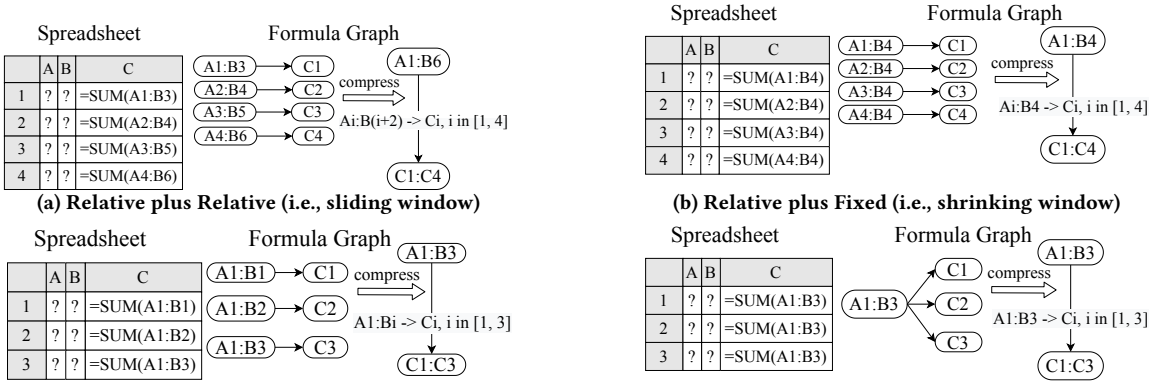


Figure 1: A real spreadsheet with tabular locality [8]

A formula can accept raw data values or the computed results of other formulae as input, generating a network of dependencies across formulae and raw data. These dependencies are typically internally tracked using *a formula graph*, where each dependency is modeled as an edge in this graph. Formula graphs support several crucial spreadsheet functions, and are critical to the interactivity and usability of a spreadsheet system. Therefore, many spreadsheet systems, such as Excel [2] and LibreCalc [5], provide tools for visually tracing dependents and precedents of cells to help users check the accuracy of formulae and identify sources of errors, which are particularly rife in spreadsheets [3, 10]. Recently, a spreadsheet bug led to a severe under-counting of COVID-19 cases [7], while studies indicate that nearly 90% of all spreadsheets contain errors [9].

Spreadsheet errors are commonly caused due to the complexity and scale of formula graphs, which makes it difficult for spreadsheet users to easily and efficiently make sense of formula dependencies. For example, our previous study on real spreadsheets has shown that the number of dependents of a single cell can be as high as 300K while a single path can be as long as 200K edges in a formula graph [11]. But existing spreadsheet systems visualize these edges one by one [2, 5], making it hard for users to understand the overall patterns of the edges or the network, or identify potential errors.

To address this problem, we demonstrate TACO-Lens, our plugin for visualizing spreadsheet formula graphs that leverages an efficient and compact formula graph representation, called *Tabular Locality-based Compression* or TACO [11], developed by us in prior work. TACO leverages tabular locality, a key property in spreadsheets, wherein cells that are close to each other in the tabular spreadsheet layout often employ similar formula structures. Figure 1 shows a column of formulae of a real-world spreadsheet that follows tabular locality. The formulae in the column N follow the following pattern starting from N3: the IF formula in each row references the cell of the same row and the row above from column A (e.g., N3 references A3 and A2), the cell to the left (e.g., M3 for



(a) Relative plus Relative (i.e., sliding window)

(b) Relative plus Fixed (i.e., shrinking window)

(c) Fixed plus Relative (i.e., expanding window)

(d) Fixed plus Fixed (i.e., fixed window)

Figure 2: Examples of four basic patterns for capturing tabular locality

N3), and the cell above (e.g., N2 for N3). Tabular locality is prevalent in real-world spreadsheets because users often do not write each formula by hand but use existing spreadsheet tools, such as copy-paste and autofill, to generate formulae automatically. For example, autofill allows users to repeat the pattern of a cell to fill adjacent cells. Users could also programmatically generate a large spreadsheet, which still likely respects tabular locality.

Therefore, TACO leverages tabular locality to compress the formula graphs, helping end-users easily consume the complex dependencies via a more compact representation. For the example in Figure 1, we compress the dependencies into 4 edges, where each edge is annotated with the information for reconstructing the original dependencies. The first compressed edge, for instance, represents the dependencies M2 $\longrightarrow$ N2, M2 $\longrightarrow$ N3, $\cdots$, M6949 $\longrightarrow$ N6949 as Ni $\longrightarrow$ Mi, where i varies from 2 to 6949. This way, end-users can easily identify patterns or outliers in formula graphs.

Technically, TACO defines four basic patterns to capture tabular locality based on analysis of real-world spreadsheets and adopts a generic framework that compresses formula graphs based on these predefined patterns. This framework additionally includes novel algorithms for finding dependents or precedents directly on the compressed graph and incrementally maintaining the compressed formula graphs with respect to updates to reduce response time.

In this demonstration, we implement an Excel plugin that visualizes the compact representation of TACO, called TACO-Lens, where users can visually explore formula graphs compactly, and find dependents or precedents given a spreadsheet region. In this demonstration, users can compare the visualizations of different formula graphs for TACO-Lens, Excel's built-in dependency tracing tool, and a baseline approach that does not compress formula graphs, and compare the response time and usability of different approaches. In the rest of this paper, we will introduce TACO in Section 2, and present TACO-Lens and our demonstration plan in Section 3. Details about TACO can be found in the main paper [11].

2 THE TACO FRAMEWORK

We briefly introduce some key concepts before introducing TACO.

Preliminaries. A spreadsheet organizes a set of *cells* in a tabular layout, where each cell is referenced using its column index, i.e., A, $\cdots$, Z, AA, $\cdots$, and row index, i.e., 1, 2, $\cdots$. A *range* is a rectangular region of cells, identified by the top-left (called *head*) and bottom-right (called *tail*) cells. A cell that contains is called a formula *formula cell*. A formula graph is a directed acyclic graph (DAG) that stores the dependencies of formulas referencing other ranges as edges. Each formula is parsed to get the set of ranges the formula references, with a directed edge added from each referenced range to the formula cell. Given a directed edge $e = (prec, dep)$, we call *prec* the *direct precedent* of *dep*, while *dep* is called the *direct dependent* of *prec*. Compared to a regular graph, one unique property of a formula graph is that a vertex in this graph can be a range. So, to quickly find ranges that overlap with an input range, we will build an index (e.g., R-Tree) to accelerate this operation.

TACO framework components. The TACO framework includes two parts: the patterns for capturing tabular locality and algorithms for efficiently compressing, querying, and maintaining formula graphs. Each pattern will compress N dependencies into one edge and implement four key functions. Given these patterns along with their key functions, we design novel algorithms that employ these key functions to compress, query, and maintain the formula graphs.

2.1 Patterns for Capturing Tabular Locality

TACO patterns. TACO considers four basic patterns for capturing tabular locality. Our discussion assumes each pattern compresses the dependencies of a column of adjacent formula cells, where each formula has only one referenced range. This can be extended to the case when there are multiple referenced ranges, and when there is a row of adjacent formula cells. We use the examples in Figure 2 to illustrate each of the basic patterns.

The first basic pattern involves each formula cell having the same relative positions with respect to both the head and tail cell of its referenced range, referred to as Relative plus Relative or RR for short. Figure 2a shows an example, where each formula cell Ci references the ranges Ai:B(i+2), where i varies from 1 to 4.

The second pattern has the head cell of each referenced range being relative to the corresponding formula cell, but the tail cell is fixed, referred to as Relative plus Fixed or RF for short. For example, Figure 2b shows that each formula cell Ci references the range Ai:B4, where i varies from 1 to 4.

The third pattern, Fixed plus Relative or FR, is symmetrical to RF. The final pattern, Fixed plus Fixed or FF, has each formula cell referencing the same range, as shown in Figure 2d.

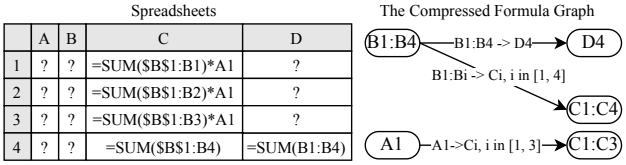


Figure 3: A compressed formula graph

Algorithms for the key functions. The key functions are used to build, query, and maintain an compressed edge. We design algorithms to support these functions while not decompressing the edge. In this paper, we describe the key ideas of the algorithms for building and querying a compressed edge. The details of all algorithms can be found in our full paper [11].

The compression algorithm considers compressing a dependency into a compressed edge or another dependency based on the definition of the basic patterns. For example, we can compress A5:B7 ⟶ C5 into the compressed edge in Figure 2a since it follows the Ai:B(i+2) ⟶ Ci pattern and C5 is adjacent to C1:C4.

Two key functions are adopted to support finding dependents or precedents for an input range in a compressed edge. For the example in Figure 2a, if the input range is A2:A3, its dependents are C1:C3. Our algorithms leverage the relative positions along with the range of the compressed formula cells to find the dependents and precedents at a constant time, independent of the size of the input range, which reduces the time for finding dependents or precedents.

2.2 Compressing, Querying, and Maintaining Formula Graphs

The algorithms in the TACO framework will employ the key functions of each pattern to compress, query, and maintain formula graphs. We now describe the key ideas for the compression and querying algorithms. Additional details and complexity analysis can be found in our full paper [11].

The compression algorithm. This algorithm compresses a list of dependencies between formula cells and their referenced ranges by repeatedly inserting each dependency into the compressed graph and determining how the dependencies are compressed using the aforementioned patterns. Since we compress the dependencies for adjacent formula cells, we use this constraint to quickly find the candidate edges that one inserted dependency can be compressed into. If there are multiple candidate edges, we leverage several heuristics based on our analysis of real-world spreadsheets to decide the edge that can best reduce the graph size (e.g., prioritize column-wise compression over row-wise).

Querying the formula graph. Finding dependents or precedents in a compressed formula graph is done via a modified Breadth-First-Search (BFS). We use the example in Figure 3 to illustrate the modifications and assume the case of finding the dependents of B2.

Similar to conventional BFS, the overall algorithm starts with the input range and recursively finds their direct dependents. There are two major differences when finding the direct dependents for an input range. First, we need to consider all of the ranges that overlap with the input range when finding direct dependents. In our example, we need to consider B1:B4, which overlaps with B2. This step is done via an index (e.g., R-Tree) that is built on the vertices in the formula graph. Second, since an edge in the formula graph can be a compressed one, we need to find the real direct dependents that are related to the input range. For our example, one direct dependent of B1:B4 is C1:C4, but only C2:C4 depend on B2. Therefore, we use the key function for the corresponding pattern to find the real direct dependents for the input range (e.g., C2:C4 for B2 in the edge B1:B4 ⟶ C1:C4). After that, we will recursively find the direct dependents of C2:C4 until the search ends.

3 DEMONSTRATION PLAN

We visualize TACO's compact representation as part of a Microsoft Excel plugin, TACO-Lens. In this demonstration, users can explore formula graphs and trace dependents or precedents within TACO-Lens and compare it with other approaches.

3.1 Implementation and User Interface

TACO-Lens is a web application that leverages the JavaScript API provided by Excel to access the worksheets [4]. This application is supported by a web server that implements the TACO framework and provides the functionalities for building, querying, and maintaining formula graphs. We adopt the Cytoscape library [1] to visualize formula graphs.

The interface of TACO-Lens is shown in the right half of Figure 4. When users click the `Generate the Entire Graph` button, the plugin will send the content of the current worksheet to the web server to build the compressed formula graph and then visualize the graph. We assign different colors to different edges based on their compression patterns. For example, in Figure 4, the blue color represents the RR pattern while the orange color represents the FF pattern. Users can also zoom in/out and pan to explore different parts of the formula graph. In addition, users can inspect the (direct) dependents/precedents of a spreadsheet range. To do this, they select the range they are interested in on the worksheet and click the corresponding button in the plugin. Then, the corresponding dependents/precedents are visualized.

For each operation, including generating the entire graph and finding (direct) dependents/precedents of a selected range, the plugin will track the end-to-end time for the corresponding operation and display the time in the "Response Time" text, as shown in Figure 4. This way, users can quantitatively compare the response time of TACO with other approaches.

3.2 Baselines

We have implemented a baseline approach, called *NoComp*, in TACO-Lens to allow users to explore the uncompressed formula graph. The button at the top of the plugin allows users to switch between the two approaches. Finally, users can try Excel's built-in dependency tracing tool and compare it with TACO and NoComp. The visualization for Excel's tracing tool is shown on the left half of Figure 4, where each dependency is visualized as a blue arrow.

3.3 Usage Scenario

As part of the demo, we will prepare ten spreadsheets whose uncompressed formula graphs have the largest numbers of edges from Enron and Github datasets[8, 11]. Their uncompressed formula

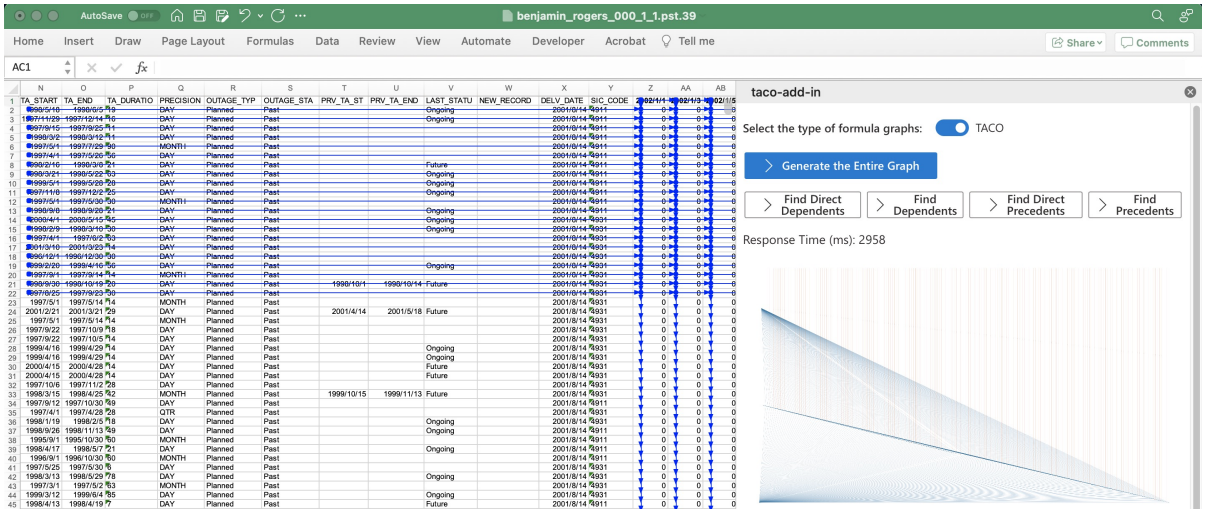


Figure 4: TACO-Lens in Microsoft Excel

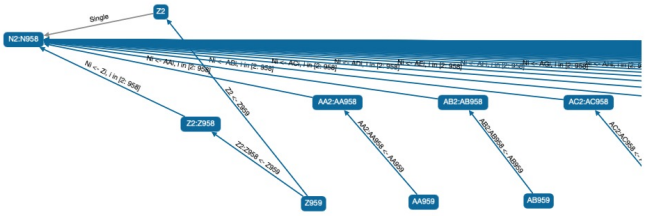


Figure 5: Dependents of N2:N958

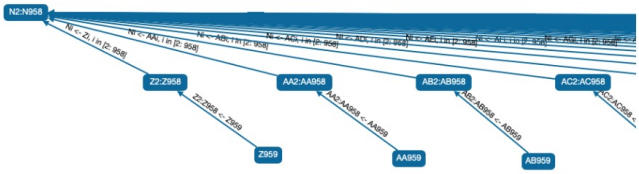


Figure 6: Dependents of N2:N958 after fixing the outlier edge

graphs have 1,343,107 edges on average, and the largest one contains over 3,000,000 edges. TACO can reduce its number of edges to less than 0.01% of the original number of edges.

In addition, we will design use cases where users use TACO-Lens to detect potential errors. We now describe one such user case in one real-world spreadsheet.

Overview of the Formula Graph. First, the user can see an overview of the formula graph for this spreadsheet, as in Figure 4. The user will find most blue edges are connected to two nodes on the left part of the visualized formula graph. After zooming in, the user will find the two nodes represent ranges O2:O958 and N2:N958.

Finding Dependents and Outlier Detection. To understand the dependents of the two ranges, the user will adopt the "Find Dependents" button. Assuming that the user wants to check the dependents of N2:N958, which are visualized in Figure 5, the user will find that the cells Z2:Z958 and many columns to the right (e.g., AA2:AA958) reference N2:N958 using the RR pattern (e.g., Ni → Zi), but there is an edge that is related to Z2 but does not follow this pattern (i.e., the gray edge).

The user can check the formula cell at Z2 or other related cells to decide whether this is an outlier edge. If the user chooses to remove it, then all cells in column Z follow the same pattern, as shown in Figure 6. Without TACO's compressed formula graph and compact visualization, it will be much more time-consuming for the user to understand the patterns and detect potential errors.

4 CONCLUSION

We demonstrate TACO-Lens, which visualizes TACO's compact formula graph representation and allows users to compare our tool with a baseline and with Excel's built-in tracing tool for exploring formula graphs. This demonstration highlights how TACO can help users more easily make sense of the patterns in formula graphs, find dependents/precedents, and detect potential errors.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. We acknowledge support from grants IIS-2129008, IIS-1940759, and IIS-1940757 awarded by the National Science Foundation as well as EPIC lab sponsors: Adobe, Microsoft, Google, G-Research, and Sigma Computing.

REFERENCES

[1] Cytoscape. https://cytoscape.org/ [Accessed: 2023-06-17].
[2] Display the relationships between formulas and cells. https://support.microsoft.com/en-us/office/display-the-relationships-between-formulas-and-cells-a59bef2b-3701-46bf-8ff1-d3518771d507 [Accessed: 2023-06-17].
[3] EuSpRIG Horror Stories. http://www.eusprig.org/horror-stories.htm [Accessed: 2023-06-17].
[4] Excel javascript api. https://learn.microsoft.com/en-us/office/dev/add-ins/reference/overview/excel-add-ins-reference-overview [Accessed: 2023-06-17].
[5] Trace Dependents. https://help.libreoffice.org/latest/lo/text/scalc/01/06030300.html [Accessed: 2023-06-17].
[6] Excel vs. Google Sheets usage — nature and numbers. https://medium.com/grid-spreadsheets-run-the-world/excel-vs-google-sheets-usage-nature-and-numbers-9dfa5d1cadbd [Accessed: 2023-06-17], 2018.
[7] L. Kelion. Excel: Why using microsoft's tool caused covid-19 results to be lost. *BBC News*, 5, 2020.
[8] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.
[9] S. Leung. Sorry, your spreadsheet has errors (almost 90% do). *Forbes*, 2014.
[10] R. R. Panko. What we don't know about spreadsheet errors today: The facts, why we don't believe them, and what we need to do. *CoRR*, abs/1602.02601, 2016.
[11] D. Tang, F. Chen, C. D. Leon, T. Wattanawaroon, J. Yun, S. Seshadri, and A. G. Parameswaran. Efficient and Compact Spreadsheet Formula Graphs. In *ICDE'23*.