Matrix depictions for large layered graphs

Benjamin Watson, *Senior Member, IEEE*, David Brink, Matthias Stallmann,
Ravinder Devajaran, Matt Rakow, Theresa-Marie Rhyne, *Senior Member, IEEE*, and Himesh Patel

Abstract—Traditional node-link depictions of layered graphs such as flow charts and process or genealogy diagrams are in widespread use. Layers emerge from applied context (e.g. process stages or familial generations), or are inserted to improve visual clarity. However, these diagrams quickly lose their utility as graph complexity grows. Layout algorithms such as crossing minimizers can preserve utility for larger graphs, but also quickly reach their limits. We offer three new matrix depictions for layered graphs: sorted matrices, centered & sorted matrices, and quilts. Unlike node-link depictions, these matrix depictions scale well to layered graphs containing several hundred nodes. To date, we have only implemented the most complex and scalable of these depictions: quilts. We demonstrate quilting using activity-based management (ABM), an application that must depict layered graphs with thousands or even hundreds of thousands of nodes. For displaying layered graphs that reach such sizes, we show how to summarize these depictions using online analytical processing (OLAP) hierarchies.

Index Terms—E.1.d Graphs and networks, G.2.2.a Graph algorithms, I.6.9.c Information visualization.

1 The Scalability of Layered Graph Depictions

Layered graphs such as process diagrams and structure or flow charts have wide-ranging application. These graphs group nodes into layers defined by applied context, or introduced to increase visual clarity.

Traditional, node-link depictions of layered graphs order layers top-to-bottom, and arrange nodes inside each layer into a horizontal line [3], [14]. *Proper links* connect nodes on succeeding layers, while *skip links* bypass layers, moving forward or backward in layer order. Crossing minimization algorithms reduce the intersection of proper links, and can improve legibility. Nevertheless, as the number of nodes and links grows, these depictions can become quite muddled, with viewers having trouble understanding graph connectivity. Figures 5 and 6 illustrate this *scalability* problem.

Scalability is also a problem for unlayered graphs. The matrix depiction first suggested by Bertin [4] offers a good solution. Ghoniem et al. [12] compared matrix to node-link depictions, finding that matrices were much clearer than nodes and links when graphs had more than 20 nodes. For path finding however, matrices were not superior until graphs had 100 nodes.

To address the scalability problem for layered graphs, we introduce three new matrix representations: *sorted matrices*, *centered & sorted matrices*, and *quilts*. Below, we review existing methods for depicting layered graphs, describe the meaning and manufacture of each new representation, and compare their strengths and weaknesses. To date we have only implemented quilts [27], the most scalable of these representations. We offer our experience with quilts in the activity-based management (ABM) application created by SAS, one of the world's largest software companies. Because no depiction is infinitely scalable, we show how to summarize these depictions with online analytical processing (OLAP).

- *Benjamin Watson is with NC State University, Email: bwatson@ncsu.edu*
- *David Brink is with SAS Institute, Email: david.brink@sas.com*
- *Matthias Stallmann is with NC State University, Email: matt_stallman@ncsu.edu*
- *Ravinder Devarajan is with SAS Institute, Email: ravi.devarjan@sas.com*
- *Matt Rakow is with NC State University, Email: marakow@ncsu.edu*
- *Theresa-Marie Rhyne is with NC State University, Email: tmrhyne@ncsu.edu*
- *Himesh Patel is with SAS Institute, Email: himesh.patel@sas.com*

2 Node-link Depictions

There are many layout algorithms for layered graph depiction, the best known of which is Sugiyama's [23], based on work by Warfield [25] and Carpano [5]. Most methods for depicting layered graphs have three phases: layer assignment, crossing minimization, and horizontal placement. In this section, we sketch the approaches taken in these phases by the STT and other methods [3], [10].

If layers are not derived directly from the application, layout [8], [9], [17] can introduce them by minimizing one or more of the following objectives: (a) height, that is the number of layers; (b) height given a fixed width; and (c) dummy nodes, that is the total number of layers skipped by skip links. Note that minimizing the width by itself is trivial — assign one node to each layer — but does not lead to aesthetically pleasing drawings.

To minimize link crossings, layout algorithms permute the nodes on each layer. While this is an NP-hard problem with only two layers [7], [11], several good heuristics are available, most based on sweeping from the top layer to the bottom, then from bottom to top, a predetermined number of times [3], [10]. During each sweep, an algorithm fixes the order in the current layer while permuting the next, then fixes the order of the newly permuted layer order as a starting point for the layer after that.

The final phase positions each layer at particular y-coordinates, attempting to minimize the number of bends in the skip links and/or the total/max deviation from vertical of the links issuing from the layer. This is a quadratic programming problem.

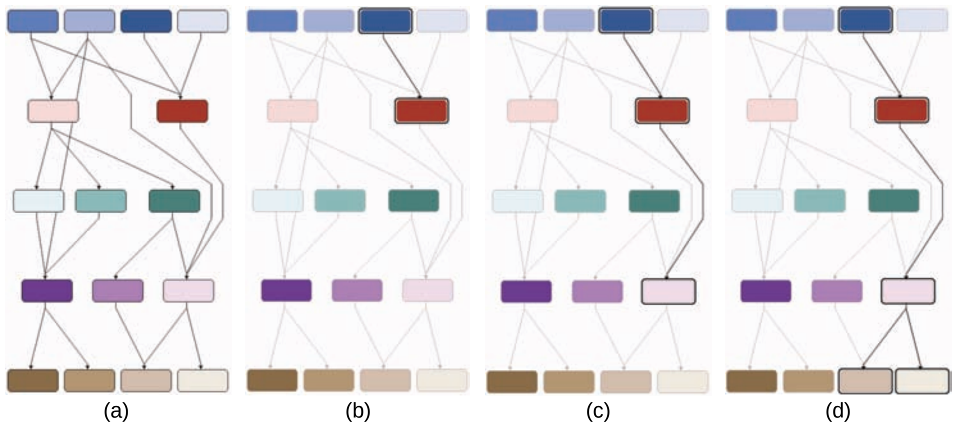


Fig. 1: ordered left to right — (a) a simple layered graph, colored here only for comparison with following figures; (b) following a link from the dark blue to the dark red node; (c) to the light purple node, skipping the green layer; (d) to the two light nodes on the brown layer.

As a final note, a layered layout does not require acyclic nor even directed graphs. If there are directed cycles, a layout algorithm can temporarily reverse some link directions to create an acyclic graph. Minimizing the number of such links is NP-hard but a reasonably good greedy algorithm exists [3]. If the graph is undirected, there are many ways to assign directions to the links to create a directed acyclic graph (DAG) – any numbering of nodes defines a DAG if we direct all links from lower to higher numbered nodes. The main problem with this numbering is that it may not be clear what sort of numbering will result in an aesthetically pleasing depiction.

3 Three new matrix depictions

We derive sorted and centered & sorted matrices quite directly from matrix depictions for unlayered graphs, preserving familiarity while gaining scalability. Quilts trade some extra complexity for additional scalability. In this section, we introduce these three depictions, making regular comparisons to the node-link depictions of the simple graph and paths lengthening in Figures 1a to 1d.

3.1 Sorted matrices

We may view a layered graph as an unlayered graph with its nodes formed into disjoint sets, and those sets ordered with respect to one another. With such a view, one can easily imagine specializing Bertin's matrix depiction [4] to layered graphs by sorting the rows and columns of the matrix by layer. Figure 2a shows the sorted matrix that corresponds to the node-link graph in Figure 1a. We add color primarily for illustrative reasons, emphasizing the correspondence between the nodes and rows/columns in these two depictions, but color also more clearly distinguishes nodes from links in the matrix.

Reordering the graph nodes should not reduce the scalability that Ghoniem et al. [12] admired. Moreover, the ordering creates a useful grouping of links by layer pairs, with intra-layer links centered around the diagonal, backward skip links below it, proper links just above it, and forward skip links above the proper links. The further a link from the diagonal, the more layers the link skips. The sorted matrix in Figure 2a emphasizes these groups with negative spacing (spatial separation w the background color), while Figure 2b labels these groups for illustration only. The resulting visual hierarchy is particularly useful when seeking links in homogeneously populated matrices. Figure 2a's sorted matrix highlights the fact that the graph has only three skip links, while this is difficult to see in Figure 1a.

Sorted matrices retain the matrix depiction's weakness in depicting paths, introducing a discontinuity (from the top of a column to the beginning of the corresponding row) at every path node except the first and last. Figures 2c through 2e illustrate this by following the path shown in Figures 1b through 1d. Also, by placing layers at the perimeter, sorted matrices deemphasize the central feature of layered graphs.

3.2 Centered & sorted matrices

Shen and Ma [22] recently noted that most (including Ghoniem et al. in [12]) overlook an alternative way of reading matrix representations of unlayered graphs we call *centering*. When following paths, one reaches graph nodes by moving toward the matrix diagonal rather than the end of a row or column. Thus to follow a link from node A to node B, one simply moves from node A on the diagonal to link AB in the matrix, and then back to the diagonal at node B. Centering not only eliminates path discontinuities, it reduces path length. It does this without altering the matrix itself.

If we combine Shen and Ma's centering with sorting by layer, we have a novel and very effective centered & sorted matrix representation for layered graphs. Sorting preserves scalability and creates identifiable layers in the matrix, while centering shortens paths, eliminates their discontinuities, and makes layers central to the depiction. Figures 3a through 3c illustrate how centering changes following of the same graph path shown in Figures 1b – 1d and 2c – 2e. The centered path in Figure 3c is much simpler than the traditional path in Figure 2e. Note that this path, which does not follow any backward skip links, lies completely on the "forward" side of the diagonal, while the same path on the sorted matrix (Figure 1d) spans the entire matrix. For centered & sorted matrices, color is particularly useful in differentiating nodes from links, and drawing attention to layers.

3.3 Quilts

As Figures 2 and 3 show, when the majority of links in a layered graph are proper, both sorted and centered & sorted matrix depictions are quite sparse. Unfortunately, proper links often dominate layered graphs, and our two matrix depictions are not as spatially efficient and

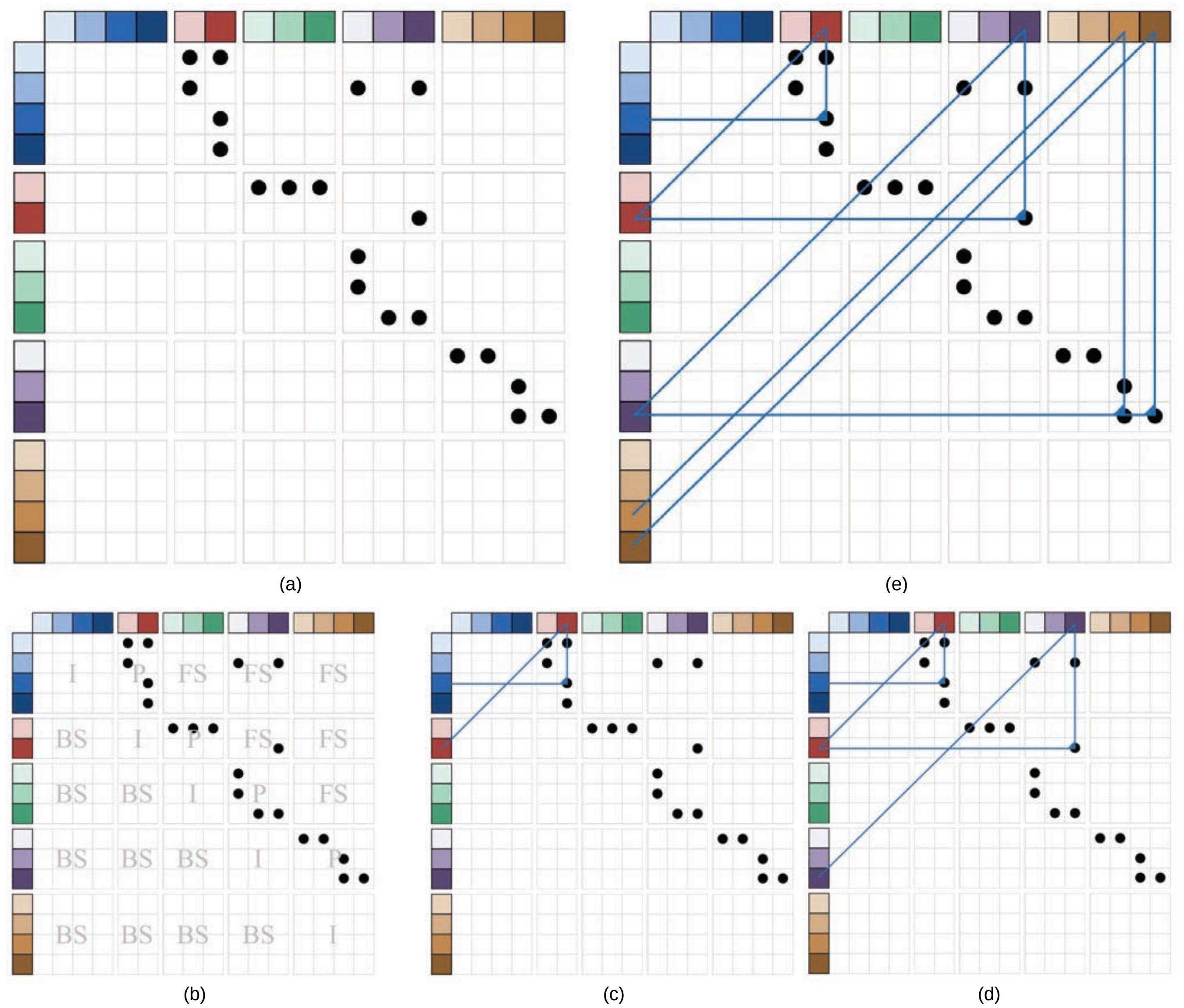


Fig. 2: in counterclockwise order — (a) the layered graph of Fig. 1 depicted as a sorted matrix, with the same color scheme; (b) highlighting the link grouping induced by sorting, with I=intra-layer, P=primary, FS=forward skip and BS=backward skip; (c) following the same link followed in Fig. 1b; (d) the same links followed in Fig. 1c; (e) in Fig. 1d. Note the sparseness of the matrix, and the complexity of the link path.

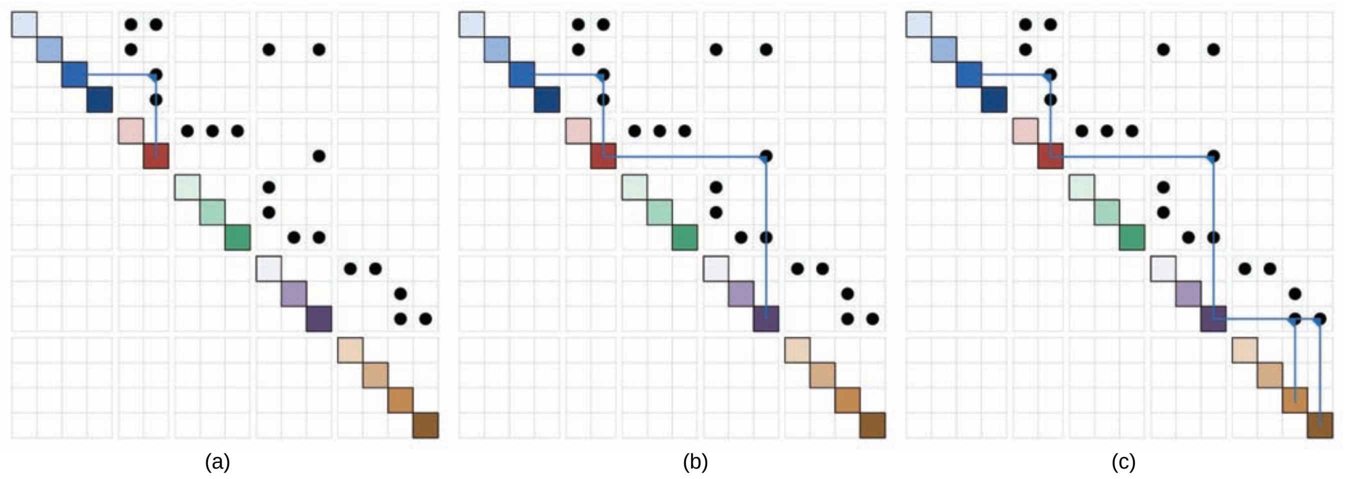


Fig. 3: ordered left to right — (a) the layered graph of Fig. 1 depicted as a sorted & centered matrix, with the same color scheme, and following the same link followed in Fig. 1b; (b) the same links followed in Fig. 1c; (c) in Fig. 1d. Note the visual simplicity of the link path.

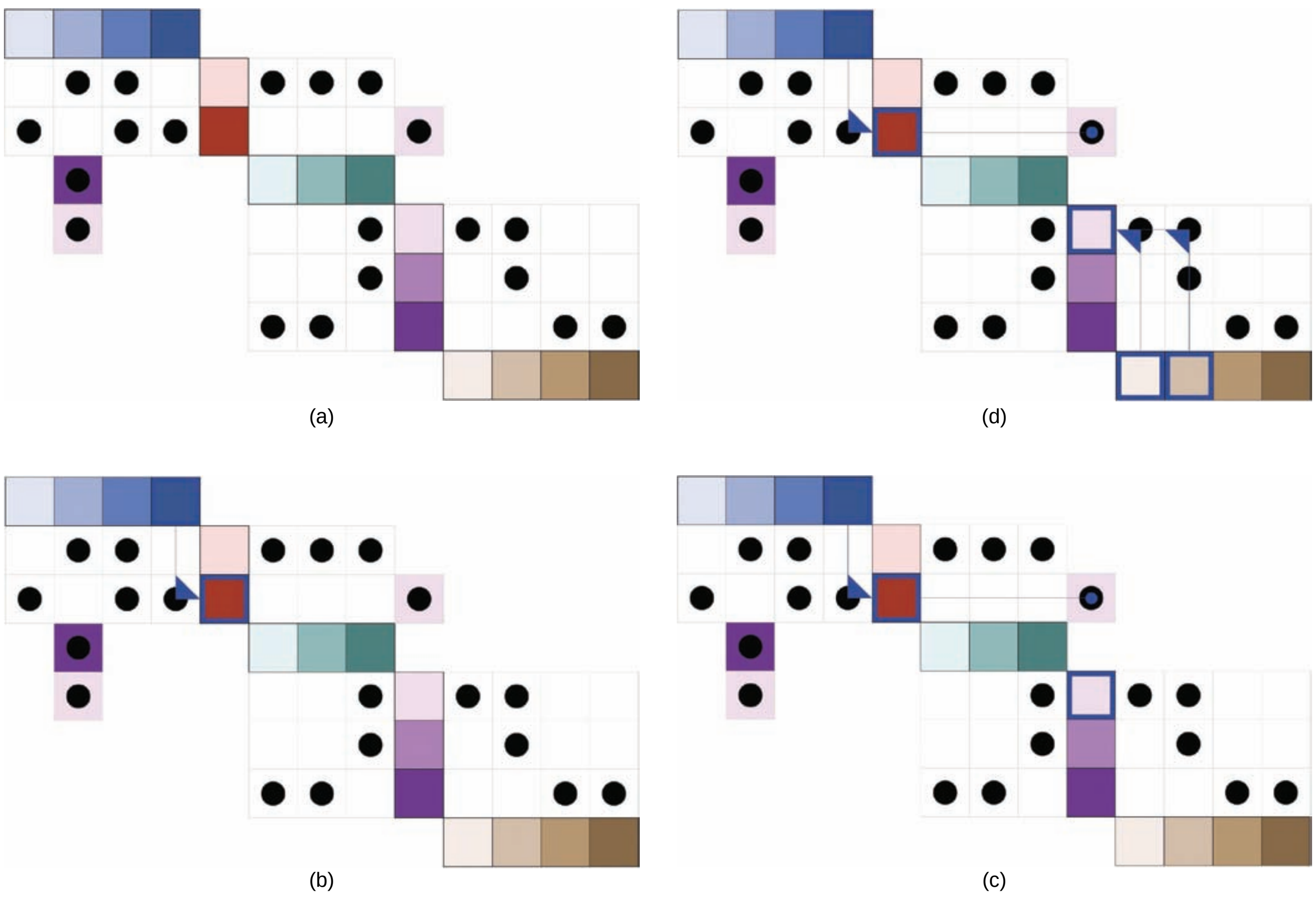


Fig. 4: in counterclockwise order — (a) the layered graph of Fig. 1 depicted as a quilt, with the same color scheme; (b) following the same link followed in Fig. 1b; (c) the same link followed in Fig. 1c, note the transition across the purple skip link to the identically colored node; (d) the same links followed in Fig. 1d. Note also the spatial efficiency of this depiction.

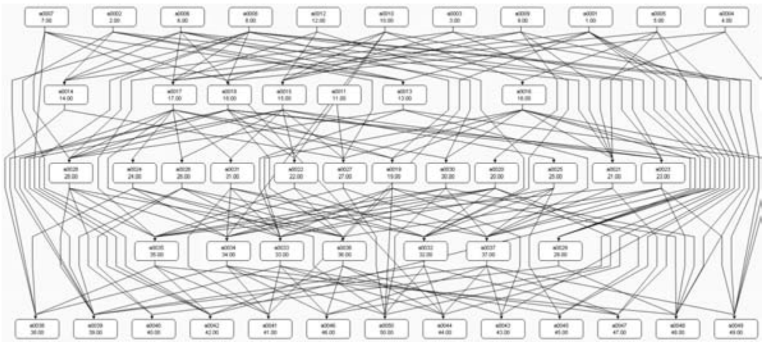
scalable as we might like. Moreover because viewers often have difficulty locating isolated links precisely, our depictions are also less legible than we might wish.

We designed quilts [27] to address these problems. Figure 4a shows a quilt depiction of the graph in Figure 1a. We represent each graph layer using a colored *level*, with each cell in the level corresponding to a node in the layer. Thus the blue, red, green, purple and brown layers in Figure 1a correspond to the similarly colored levels in Figure 4a. We depict the set of proper links connecting each pair of adjacent layers using an achromatic matrix, with each dot in the matrix corresponding to a link in the set. Figure 4a has four such matrices, corresponding to the sets of proper links between the blue and red, the red and green, the green and purple and the purple and brown layers in Figure 1a.

To lay out the quilt, we interleave the levels and matrices in the same order as the corresponding layers and sets of proper links. We orient the first level horizontally to form a row, and place the first matrix beneath it. We orient the second level vertically to form a column, place it to the right of the first matrix, and then place the second matrix to the second level's right. We orient the third level horizontally, place it below the second matrix, and continue similarly, concluding with a horizontal level if the number of layers in the graph is odd, a vertical level if the number is even.

For each skip and intra-layer link in the graph, we create a corresponding *skip cell* in the quilt. These cells indicate their source node with position, and their destination with color. Figure 1a shows three skip links, two with the light purple node as their destination, and one (partially occluded by the red and green layers) with the dark purple node as its destination. The three corresponding skip cells in Figure 4a are located outside the proper link matrices, and colored to match their destination nodes. We locate the two skip cells with the light blue source node below the corresponding level cell's column of proper links, and the skip cell with the red source node to the right of the corresponding level cell's row of proper links. When a node has more than one skip or intra-layer link, the corresponding skip cells form a *skip list*, which we sort by destination level, and then by the destination level's cell order. Figure 4a contains one two-cell skip list below a light blue level cell.

Figures 4b through 4d follow the same paths followed in Figures 1b – 1d, 2c – 2e, and 3a – 3c. Figure 4b follows the proper link from the dark blue to the dark red node, marking the link with a small triangle oriented to the direction of the link's "turn," and highlighting the node reached by outlining the corresponding level cell. Figure


(a)

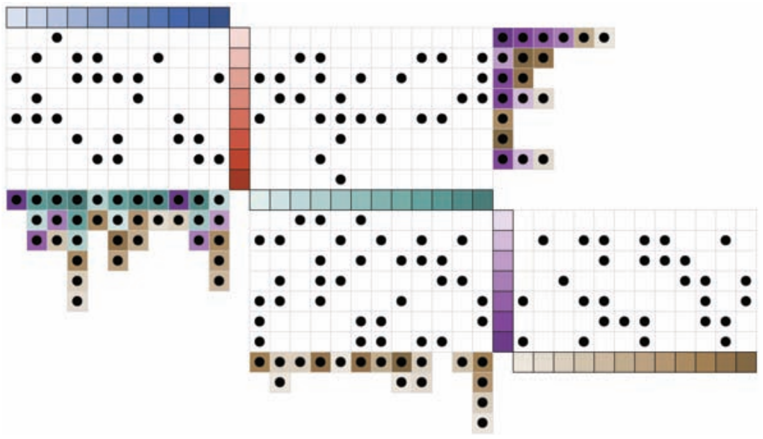

(b)

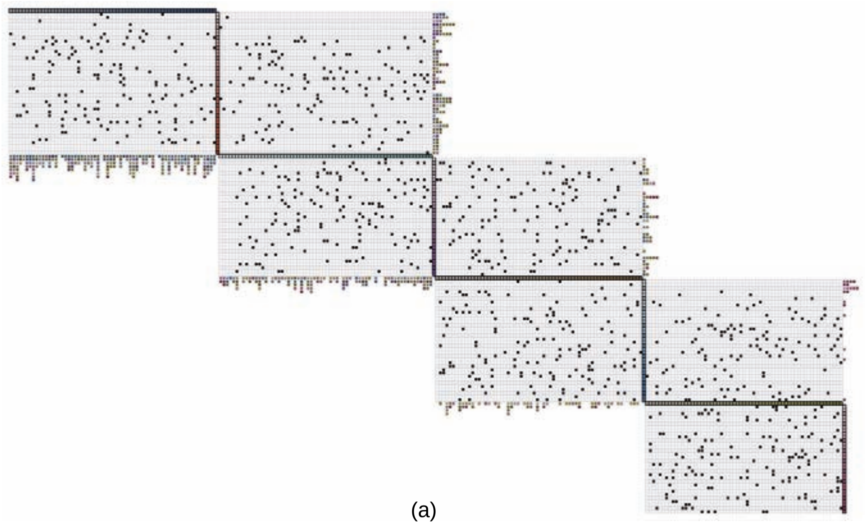
Fig. 5: (a) node-link depiction of a randomly generated 50-node graph, with 32% of its links skipping layers; (b) quilt depiction of the same graph. Links are much more legible in the quilt, though there is some uncertainty about skip link destination. Skips increase quilt size only modestly here, but exceptionally well-connected layered graphs can increase quilt size significantly.

4c follows the dark red node's skip link, marking it with a circle and bypassing the green layer to jump directly to the light purple node. Finally, Figure 4d follows the light purple node's two proper links to the brown layer.

Color plays an important role in quilts, distinguishing levels from matrices and from one another, and indicating the destination nodes of skip cells. We assign a unique chromaticity (hue and saturation) to each quilt level, rotating through moderately saturated hues first, and if the number of levels is particularly large, adding additional rotations with increased saturation. We assign each cell in a level a unique brightness, making the color of every graph node unique. (Brightness does not approach minimum or maximum, which collapse chromaticity to black and white). As a result, skip cells indicate their destination level with chromaticity, and their destination node in that level with brightness.

Quilts scale much more successfully to large graphs than our other two matrix depictions. Sorted and centered & sorted matrices place a copy of each node on both of their spatial dimensions, resulting in a size proportional to n^2, where n is the number of graph nodes. Quilts place each node along only one of their two spatial dimensions, resulting in a size proportional to $n^2/4$. The key to this spatial efficiency is the use of color rather than position to indicate the destination node of skip links. In fact, quilts typically scale even more successfully than this simple analysis indicates, since visual content outside the levels and matrices is limited. Figures 5 and 6 show how graph size affects quilt legibility.

Color perception limits quilt scalability and legibility. As the number of nodes and skip links begins to grow, it can be difficult to match a skip cell's brightness precisely to the brightness of its destination cell, though the tion level remains obvious (e.g., Figure 5). As the number of nodes and skip links grows further, skip and level cell size shrinks, locating the destination cell becomes more difficult, and the chromaticity matching required to identify the destination level be-


(a)

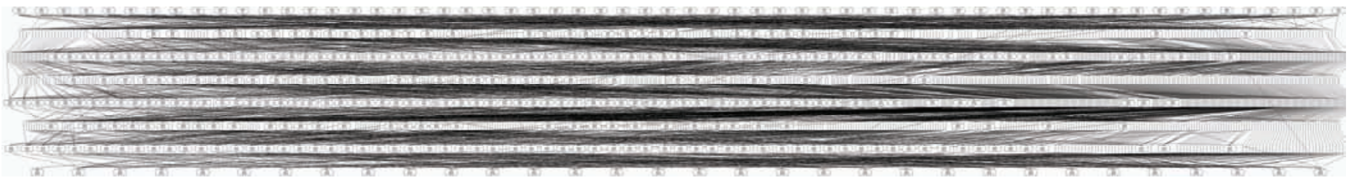

(b)

Fig. 6: (a) quilt depiction of a randomly generated, 400-node layered graph; (b) node-link depiction of the same graph. The quilt is much more legible, revealing that the first several nodes on the 2nd, 3rd and 6th layers have few proper links, and that the last few layers have very few skip links. Yet here quilts have reached the limits of their scalability: skip link destinations (both nodes and layers) are very difficult to perceive.

comes challenging (e.g., Figure 6). Research and practical experience indicate that the number of discriminable lightnesses [15], [16], [19] and chromaticities [18], [19], [23] each number in the low hundreds at best, with the small color field sizes and significant spatial separations [23] in quilts likely reducing discriminability still further.

Quilt scalability and legibility also rests on the assumption that the majority of graph links are proper. This *proper link dominance* ensures scalability by limiting the length of skip lists, and enhances legibility by limiting the total number skip cells, each of which represents a path discontinuity. Assuming proper link dominance is not unreasonable: when instead skip links are at least as numerous as proper links, layering does little to improve graph legibility, and one might argue that a different layering [8], [9], [17] or indeed an unlayered graph would be more appropriate.

Nevertheless, Figure 5 shows what can occur in a quilt when only two thirds of the links are proper. Quilt size grows modestly, while the number of path discontinuities increases with each new skip cell. When some nodes link to almost every other node, the length of skip lists can approach the number of nodes in the graph, and the quilt's bounding box can actually become somewhat larger than the corresponding sorted matrix. In this case, the number of path discontinuities in the quilt and sorted (uncentered) matrix would be similar.

3.4 Comparison

Table I compares the effectiveness we expect for the node-link and matrix depictions of medium to very large layered graphs (roughly 50 nodes and up). Node-link depictions scale most poorly, both cognitively (see [12]) and computationally (with crossing minimization being NP-hard). Sorted matrices scale more poorly than centered & sorted matrices because without centering, the number of path discontinuities increases with graph size. Quilts scale most effectively due to their efficient use of space.

With legibility, we refer to clarity of meaning for a given graph size. We base our expectations primarily on ease of path following, but also weigh the influence of the remaining comparative dimensions in the table. Ghoniem et al. [12] reported that node-link depictions were illegible by most measures when the number of nodes rose above 20; we expect the same holds true with layered graphs. We believe that sorted matrices are more legible than node-link depictions, but with frequent path discontinuities, less legible than quilts and centered & sorted matrices. We think it likely that quilts are runners-up in the legibility contest, with easily followed proper links, but also path discontinuities at skip cells that can be difficult to follow due to perceptual color constraints. Centered & sorted matrices are the obvious winners, with no path discontinuities.

Since all of these depictions portray layered graphs, we believe layers should stand out from the rest of the graph, and be distinct from one another. With alignment, node-link depictions group layer nodes and separate them from the rest of the graph. Each layer has a different vertical position. In quilts, layers trace a distinctive staircase pattern through the graph, with succeeding layers having differing orientation and color. Thus in both the node-link and quilt depictions, layer clarity is excellent. In centered & sorted matrices, layers form a diagonal stripe through the center of the depiction. Negative space and color identify different layers, but we believe that this combination of cues is not quite as strong as the combinations in the node-link and quilt depictions. Finally, the sorted matrix uses the same negative space and coloring as the centered & sorted matrix, but relegates layers to the periphery and confusingly, splits them into two parts.

All of these depictions are sensitive to the distribution of links between layers. With their use of lines and curves to indicate links, node-link depictions are most sensitive, with more skip links complicating curves and creating more crossings. Even the number and distribution of proper links has a major impact. Quilt scalability and legibility also decline as the proportion of skip links grows, but not so rapidly. Paths in both sorted and centered & sorted matrices can be more difficult to follow when the spatial distribution of links is homogeneous, but this is a relatively minor concern.

The visual property a graph depiction uses to portray links impacts not only its scalability and legibility, but also its adaptability to new applied settings such as summarization. Node-link depictions of even medium-sized graphs create a visual complexity that often prevents useful adaptation. Quilts' extensive use of color for links almost precludes color's application to any other purpose. For example, it is nearly impossible to choose a color for the quilt's path highlight that always has high contrast (see the blue lines in Figure 4). Close attention is required if quilts will be used by color deficient viewers. With their purely position-based link mapping, sorted and centered & sorted matrices have the greatest adaptability, though the saturation of the uniquely effective position property does limit adaptation somewhat.

TABLE 1
COMPARISON OF GRAPH DEPICTIONS ALONG FIVE DIMENSIONS

	scalability	*legibility*	*layer clarity*	*link distribution*	*link mapping*
node-link	-	-	++	-	line or curve
sorted	√	√	√	+	position
centered & sorted	+	++	+	+	position
quilts	++	+	++	√	position and color

The four depictions are ordered on each dimension using the symbols -, √, + and ++, where - is poor and ++ is excellent.

4 Implementing Quilts

As the most scalable of our matrix depictions, quilts promise the best fit to SAS's needs. This section describes our first implementation of quilts, in particular its interactivity and its summarization functionality. Although developed for quilts, many features of this implementation should work quite well in our other matrix representations.

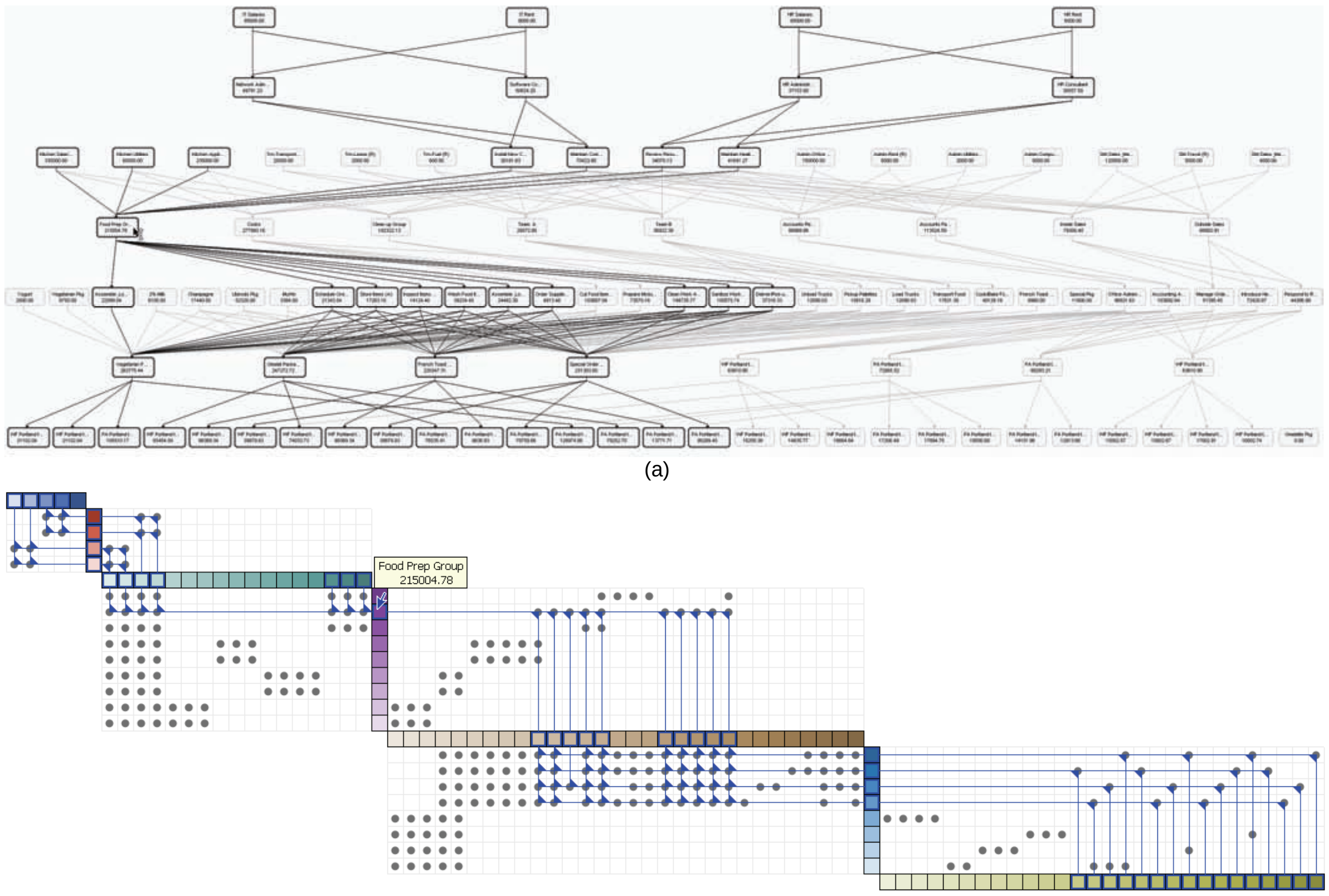


Fig. 7: (a) node-link depiction of a small graph generated by SAS's ABM application; (b) linked quilt depiction of the same graph. With click-through, the user has highlighted all links coming into and out of the "food prep group."

4.1 Interactivity

To ensure good scalability, quilt cells are typically too small to support continuous display of application-assigned node properties, especially textual labels. Figure 7 shows how viewers can hover over a cell to reveal a node or link's properties.

Figure 4 illustrates how we help viewers follow paths with highlighting. This can be particularly important in larger quilts, when following skip links is difficult. Clicking on a node highlights it and any immediately adjacent nodes. Each additional click highlights nodes that are one more link distant. We call this *click-through*. Backward click-through removes highlights from the nodes last reached, and is available by clicking on the source node while the control key is pressed. If instead the user presses the shift key while clicking, we highlight all the nodes reachable from the clicked cell. Clicking on any portion of the depiction not containing a node or link removes the highlight.

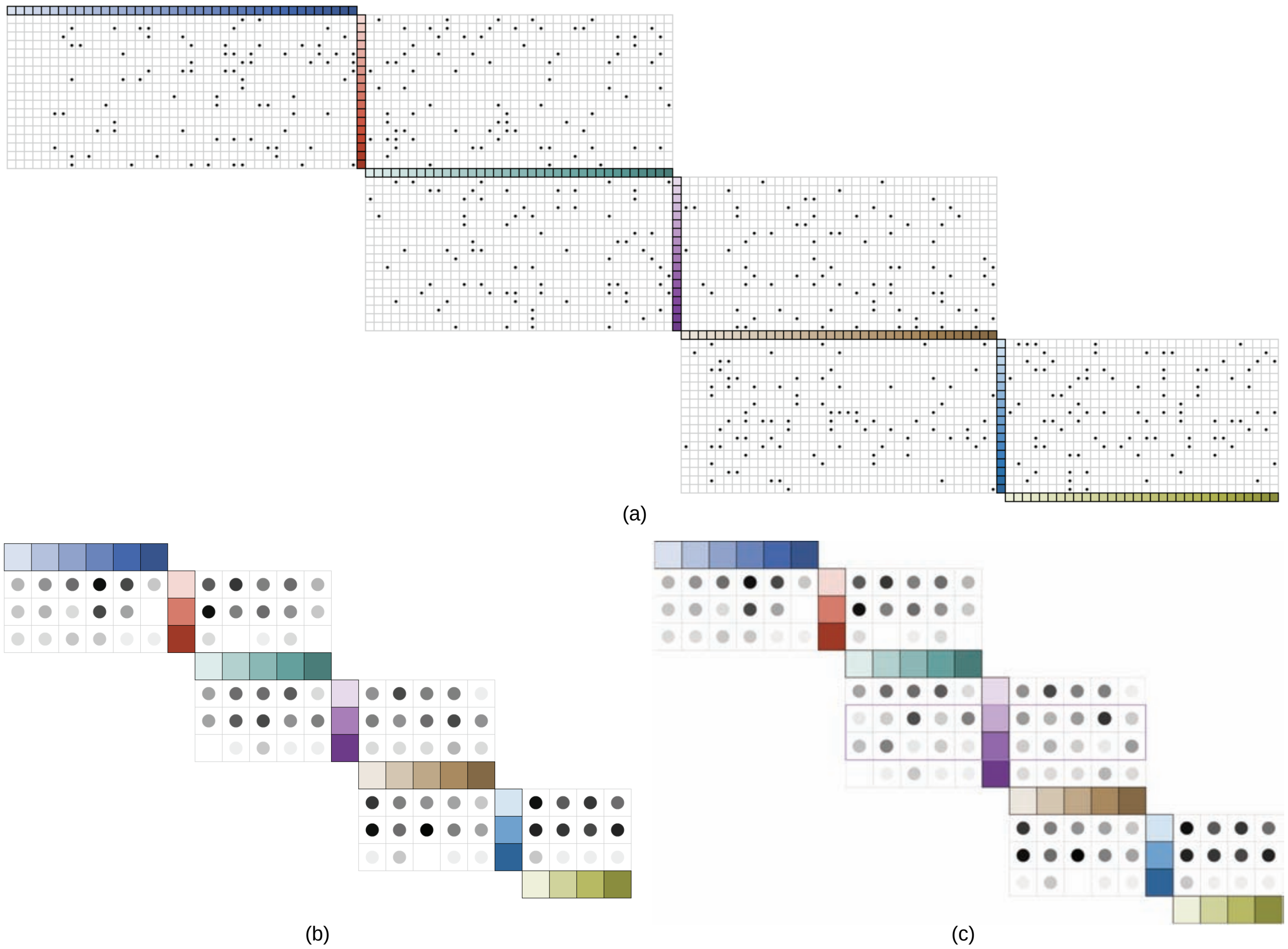
To provide viewers with the strengths of multiple layered graph depictions, we link them interactively. Figure 7 also shows how clicking in either a node-link or quilt depiction will highlight the appropriate graph path in the other.

4.2 Regular summarization

When graphs contain several hundred nodes or more, they can require summarization, since the corresponding depiction will not fit in a typical display. To summarize our new matrix depictions, we adopt the methods of much prior work [1], [2], [7], [21]: we cluster nodes.

Figures 8a and 8b show the quilt after application of a simple regular clustering scheme, in which each summary level cell except the last represents s unsummarized nodes. Each summary matrix or skip cell represents up to s^2 links. (In SAS's applied setting, we find that summary cells represent far fewer links). We depict the number of links in a summary cell to the luminance of the cell's dot, with a darker dot indicating more links.

Figure 8c shows a local zoom in the regularly clustered quilt, with one summary cell on the purple level expanded into less summarized detail cells. We assign brightness to detail cells in levels by interpolating the brightness of the surrounding summary level cells. The level zoom also expands links on the preceding and following matrix rows, creating matrix cells containing at most $s^2/2$ links. We assign brightness to the dots in these cells with the same brightness mapping used by the surrounding summary cells. We also highlight the boundary



(a)

(b)

(c)

Fig. 8: (a) quilt of a 200-node randomly generated graph; (b) summarized quilt of the same graph, with eight original nodes per summarized node; (c) zooming in on the purple level's second node.

between summary and unsummarized matrix cells with the color of the expanded level cell.

Hovering over a summary cell displays the number of summarized cells, as well as statistical summaries of any application properties in those cells. For example, hovering displays categorical properties using the number of summarized cells belonging to each category.

4.3 Quilts for ABM

We developed the quilt in the applied context of SAS's activity-based management application (ABM). ABM is an analytical application that models an organization's processes to determine accurately the cost and profitability of products and customers. ABM uses the same "numbers" found in an accountant's general ledger, but instead of viewing cost and revenue centers in traditional hierarchies (e.g. group and division or products and services), ABM models the interactions between the groups and assigns revenue to those responsible for the products or services. This allows the true nature of cost and revenue to be determined.

SAS's ABM system uses a directed graph to model the interactions within an organization. Large organizations and their processes yield complex models often containing hundreds of thousands of vertices and millions of links. While it is easy to answer specific cost or revenue questions in complex models, only visualization reveals hidden trends and uncovers the true wealth of this information. Do the processes for one product correlate to another? Are profitable products more efficiently processed? Are the most profitable customers linked to the most profitable products? Unfortunately, most of our ABM graphs overwhelm traditional node-link depictions, making visualization with them inefficient at best and often futile.

Figure 7 depicts a small ABM graph using linked quilt and node-link views. The graph models the cost processes involved in an airlines' flight catering service. By following the links coming into the food prep group highlighted in Figure 7, we find the objects that contribute to it, such as kitchen salaries and maintain healthcare programs. By following its outgoing links, we learn that the group contributes to objects such as assemble/load main trays and deliver/pickup containers. These incoming and outgoing objects have their own cost associations that affect the costs of the food prep group up and down its chain of dependencies. Next to the food prep group is the cooks group. In Figure 9 we select it, finding that even though it has the same set

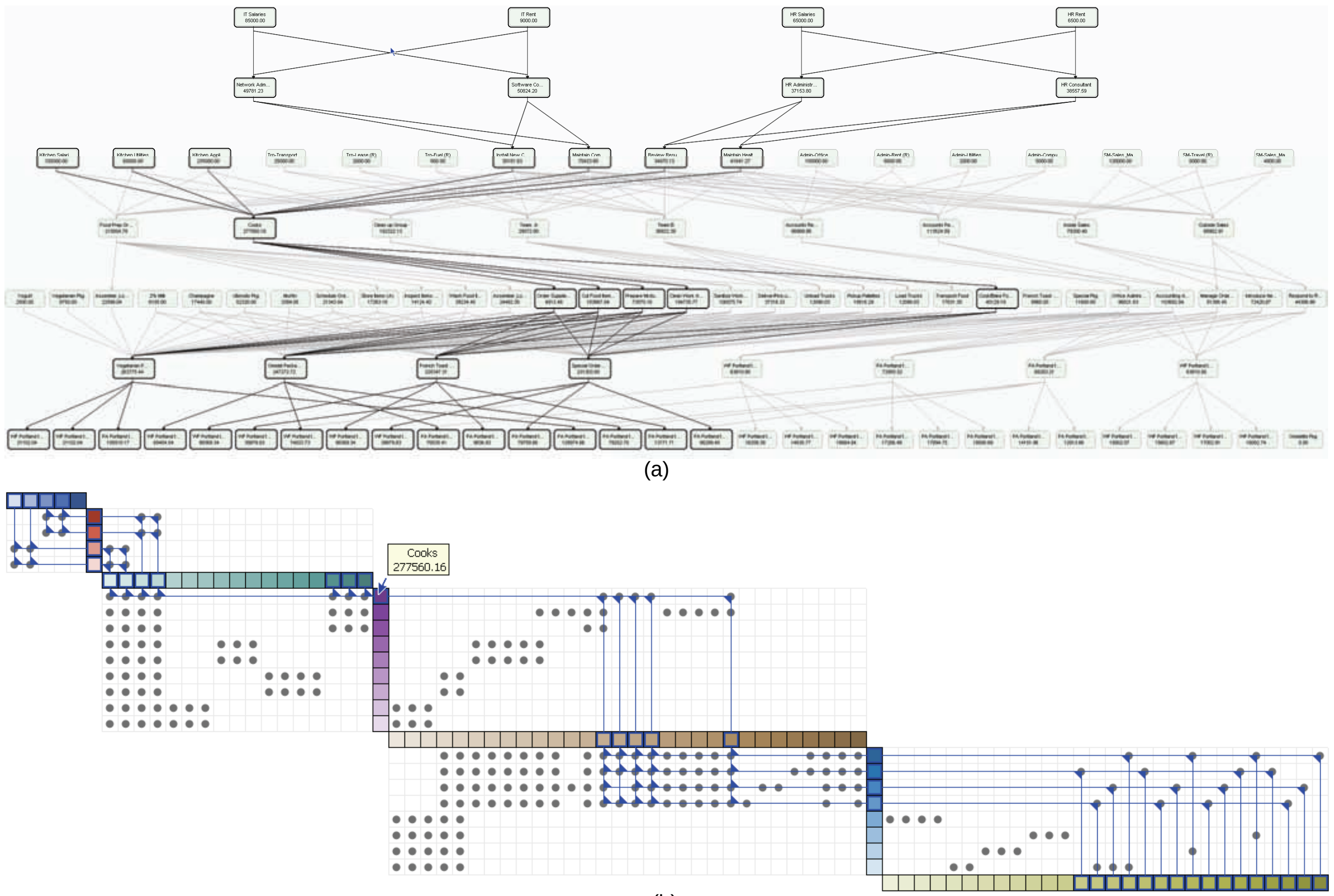


Fig. 9: (a) node-link depiction of the graph from Fig. 7a with the "cooks" node selected; (b) linked quilt depiction of the same graph. This node has the same incoming nodes as the "food prep group" node, while all but two of its outgoing nodes are different.

of incoming links as the food prep group, its outgoing links differ, except for ordering supplies and clean work area. Among many other things, this visualization tells us that any effort to reduce the cost of food preparation will also reduce the cost of any work done by cooks.

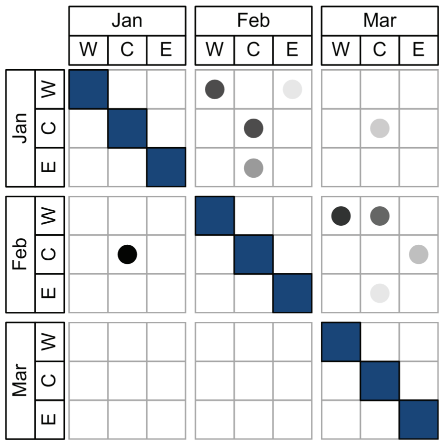
5 Summarization with OLAP Hierarchies

In the previous section, we discussed our implementation's summarization functionality. Unfortunately, regular clustering is inadequate for many applications. In fact, SAS sometimes pairs its ABM tool with an OLAP hierarchy [6], permitting businesses to examine summarizations ("rollups") and subsets (using "selection") of their activity models across several dimensions. For example, we might unroll the flight catering model of Figures 7 and 9 into time to reveal a different model for each quarter. Alternatively, we might select a region to extract the model specific to it.

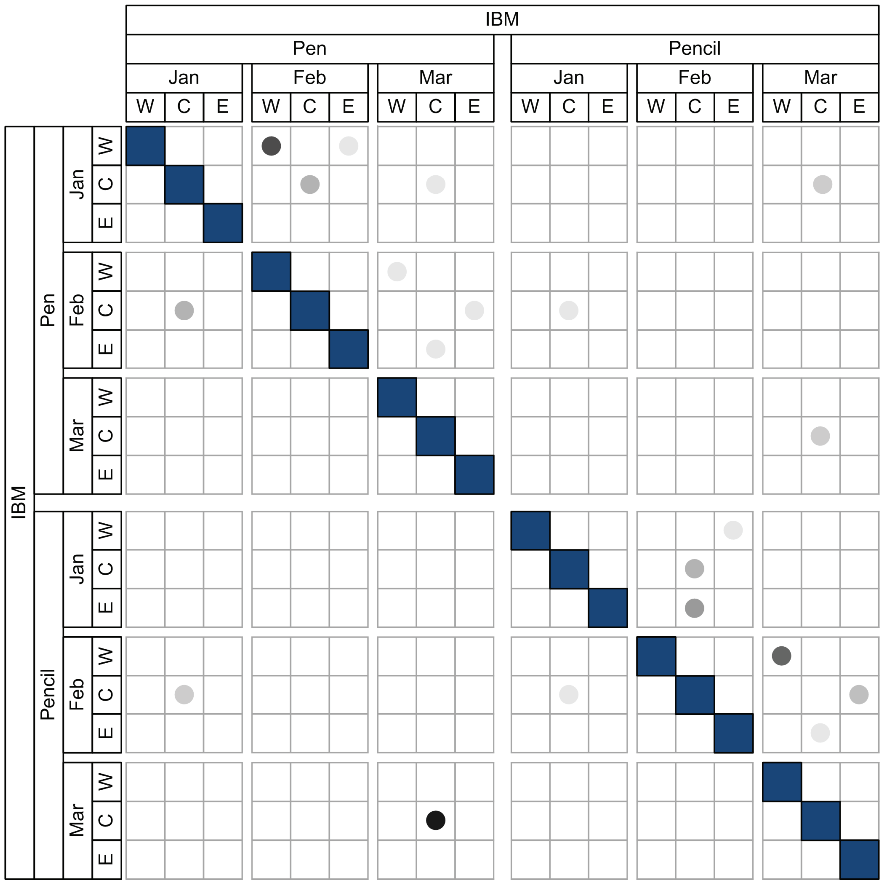
Wattenberg [26] has described PivotGraph, a tool that supports interactive exploration of unlayered graph summarizations using OLAP hierarchies. The user maps one OLAP dimension to the vertical display dimension, another to the horizontal dimension. PivotGraph then clusters the nodes in each cell of the resulting table, indicating the number of clustered nodes by changing the size of the summary node. Similarly, PivotGraph clusters the links between each pair of table cells, indicating the number of clustered links by changing the width of the summary link. Wattenberg notes that one could create a similar OLAP mapping with matrix depictions, though because they map every node to *both* the horizontal and vertical dimensions, such an OLAP-matrix mapping would have to embed one dimension recursively inside the other (e.g. time1: region1, region2; time 2: region1, region2). This reduces the legibility of the inner dimension, because it scatters each value of the inner dimension across the matrix.

We seek a similar, interactive OLAP summary capability for very large, layered graphs. Since the node-link depictions Wattenberg prefers do not scale to the sizes that interest SAS, we seek to add such a capability to the matrix depictions we have described in this paper, despite the scattering that results. (In fact, PivotGraphs will also suffer from scattering as soon as they are unrolled across three or more dimensions).

We start with summarizations across two dimensions using centered & sorted matrices. We map each value of the outer dimension to a different layer, and each value of the inner dimension to the nodes inside a layer. Figure 10a illustrates this, with the *time* dimension mapped to layers, and the *region* dimension to nodes. We separate layers from one another using negative spacing. We use



(a)



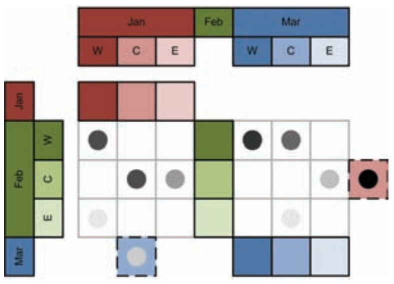
(b)

Fig. 10: (a) centered & sorted matrix depiction of a graph that spans an OLAP hierarchy, unrolled into the *time* and *region* dimensions; (b) the same graph additionally unrolled into the *product* dimension to group layers, with *customer* "IBM" selected.

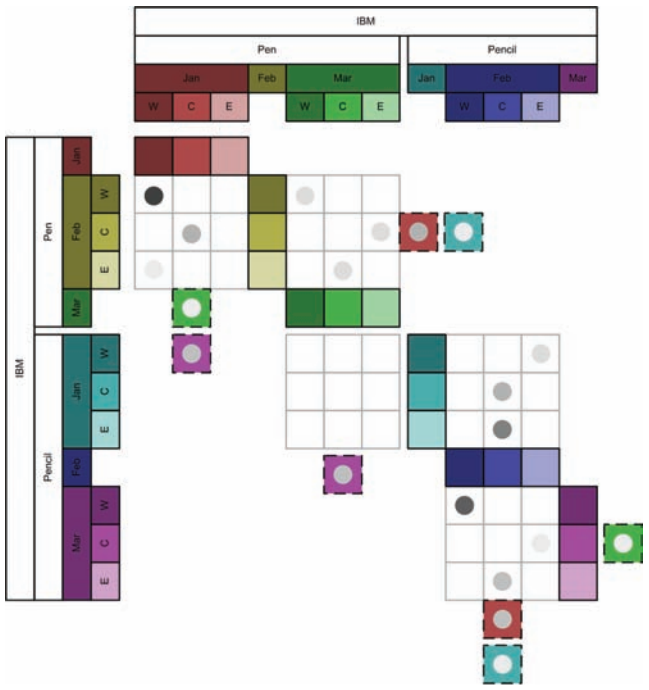
only a single color, to distinguish nodes from links. Column and row headings indicate the OLAP dimensions and values mapped to each layer and node.

To depict additional dimensions with centered & sorted matrices, we must assign layers and nodes to particular levels in the OLAP hierarchy. To enable link display, we must map nodes to the lowest (innermost) hierarchical level. We may however map layers to any hierarchical level, with higher levels grouping layers, and lower levels grouping nodes. We use additional negative spacing to indicate groups of layers, and additional positive spacing (spatial separation in the color outlining cells) to depict groups within layers. Figure 10b illustrates a depiction that groups layers. The highest hierarchical level is *customer*, in which we have selected "IBM." We have unrolled the matrix across the remaining *product*, *time*, and *region* dimensions. Because we still map *time* to individual layers, the higher-level *product* dimension groups layers. We indicate this grouping with wider negative spacing between *product* values, and by extending this spacing into the *product* headings. Were we instead to map layers to different *products*, we would use uniform negative spacing between layers, but indicate the grouping effect of *time* within layer by varying positive spacing.

OLAP summarization of quilts is similar, but must adjust to the color mapping indicating skip link destination, and the alternating use of the vertical and horizontal display dimensions. Figure 11a depicts the same rolled graph shown in Figure 10a, with *time* mapped to layers and



(a)



(b)

Fig. 11: (a) quilt depiction of the OLAP-summarized graph in Fig. 10a; (b) quilt depiction of the OLAP-summarized graph in Fig. 10b.

region to nodes. For quilts, this implies that *time* is also mapped to chromaticity, and *region* to brightness. Alternating use of horizontal and vertical dimensions varies the width/height of the *time* headings, and allows a *region* heading for each node on only one of the two spatial dimensions. We highlight skip cells amidst the increased visual complexity with dashed outlines.

To depict additional OLAP dimensions in quilts, we apply the same spatial grouping techniques we use for sorted & centered matrices. Figure 11b shows the unrolled graph depicted in Figure 10b. Negative spacing separates layers grouped by different *product* values. (Because chromaticity and orientation already group nodes into layers quite effectively, quilts do not also demark individual layers with negative spacing). We emphasize the mapping of layer to *time* by leaving headings corresponding to the higher *product* and *customer* OLAP levels uncolored. Were layers mapped to individual *products*, there would be no negative spacing at all, and positive spacing within quilt levels would indicate the grouping effect of *time*. We would emphasize the grouping effect of *product* by coloring each set of *time* values with the chromaticity of its grouping *product* value. Dashed skip cell outlines are especially valuable in this example, which highlights the fact that these cells do not respect the OLAP-induced spatial organization and spacing of level and matrix cells (nodes and proper links). As we sort skip cells, we add the same positive and negative spacing between OLAP groups that we use with level and matrix cells.

6 Conclusion and Future work

We have presented three new depictions of layered graphs that remain legible even when the depicted graphs have several hundred nodes. Although sorted matrices should scale much more effectively than node-link depictions, we see little reason to prefer them to centered & sorted matrices or quilts. These latter two depictions offer different strengths: quilts are much more scalable and give layers visual emphasis, while centered & sorted matrices are more legible and less sensitive to the distribution of links between layers.

Our work on these depictions has only just begun. We plan to continue our work by:

Proceeding with implementation. We will integrate these depictions into SAS's current ABM and OLAP product line, and evaluate their utility. This will require implementing the centered & sorted matrix depiction, as well as OLAP summarization of these matrices and quilts.

Investigating layer and crossing optimization. The application of these optimizations minimizes the number of skip links, shortens the length of those that remain, and reorders nodes within layers to minimize proper link crossings. In quilts, layer optimization should increase proper link dominance and reduce the length of skip lists, while crossing optimization should reorder level cells to group matrix dots around the diagonal. In centered & sorted matrices, layer optimization will reduce the number of skip cells and move those that remain closer to the diagonal, while crossing optimization should organize the proper links in each proper link grouping. We plan to study the effect of these optimizations on scalability and legibility.

Experimenting with interactive editing. Interactive user adjustment of node-link depictions of both layered and unlayered graphs [12], [20] is a difficult problem, given the complex optimizations they require. Interactive editing of matrix depictions of unlayered graphs [13] does not introduce the same layout challenges. Editing matrix depictions of layered graphs might offer similar advantages, with the primary challenge being the support of edits in the context of layers.

Performing user studies. It would be extremely useful to compare the effectiveness of matrix and node-link depictions of layered graphs, in much the same way that Ghoniem et al. [12] compared depictions of unlayered graphs. We would focus on legibility as the number of nodes, links, and link distribution varied. We hypothesize that the additional visual structure provided by our matrix depictions makes them more legible than a matrix depiction of an unlayered graph of identical size. In fact, we might at the same time study the effect of centering on matrix depictions of unlayered graphs: centering should make matrices more effective for smaller graphs.

7 Acknowledgements

At SAS, the Visualization group hosted our collaborative meetings, and provided the applied context that drove this research. Scott Sutton of the Activity Based Management solution group at SAS helped develop quilts for SAS's ABM application, and suggested improvements to this manuscript. Frank Wimmer of the UI Analysts group and Paul Layne in the Graph Development group provided timely help in the implementation of the test quilt application.

At NC State, the Center for Visualization and Analytics initiated and nurtured the SAS collaboration. David Crist made many useful edits and created many of the figures in this paper, while Chris Healey provided many valuable suggestions upon reading the paper.

NSF grants 0639426 and 0552802 supported this research.

8 References

[1] J. Abello & J. Korn. 2002. Mgv: a system for visualizing massive multidigraphs. *IEEE TVCG*, *8*, 1, 21–38.

[2] J. Abello & F. van Ham. 2004. Matrix Zoom: a visual interface to semi-external graphs, Proc. IEEE Info. Vis., 183-190.

[3] G.D. Battista, P. Eades, R. Tamassia & I.G. Tollis. 1999. *Graph Drawing: Algorithms for the Visualisation of Graphs*, Prentice Hall.

[4] J. Bertin. 1967. *Sémiologie graphique: Les diagrammes - Les réseaux – Les cartes.* Editions de l'Ecole des Hautes Etudes en Sciences.

[5] M.J. Carpano. 1980. Automatic display of hierarchical graphs for computer aided decision analysis. *IEEE Trans. Systems, Man, & Cybernetics*, *10*, 705-715.

[6] E.F. Codd, S. B. Codd & C. T. Salley. 1993. Beyond decision support. *Computerworld*, *27*, 30, 87-89.

[7] P. Eades & Q.W. Feng. 1996. Multilevel visualization of clustered graphs, Proc. Graph Drawing, LCNS 1190, Springer-Verlag, 101-112.

[8] M. Eiglsperger, M. Siebenhaller, M. & M. Kaufmann. 2005. An efficient implementation of Sugiyama's algorithm for layered graph drawing. *J. Graph Algorithms & Apps.*, *9*, 305-325.

[9] E.R. Gansner, E. Koutsofios, S.C. North & K.-P. Vo. 1993. A technique for drawing directed graphs. *IEEE Trans. Soft. Eng.*, *19*, 214-230.

[10] E.R. Gansner & S.C. North. 2000. An open graph visualization system and its applications to software engineering. *Software - Practice & Experience*, *30*, 11. 1203-1233.

[11] M.R. Garey & D.S. Johnson. 1983. Crossing number is NP-complete. *SIAM J. Algebraic & Discrete Methods*, *4*, 312-316.

[12] M. Ghoniem, J.-D. Fekete & P. Castagliola. 2004. A comparison of the readability of graphs using node-link and matrix-based representations. Proc. IEEE Info. Vis., 17-24.

[13] N. Henry & J.-D. Fekete. 2006. MatrixExplorer: a dual-representation system to explore social networks. *IEEE TVCG*, *12*, 5.

[14] I. Herman, G. Melancon & M.S. Marshall. 2003. Graph visualization and navigation in information visualization: a survey. *IEEE TVCG*, *6*, 24-43.

[15] H. Levkowitz & G. Herman. 1992. Color scales for image data. *IEEE CG & A*, *12*, 1, 72-80.

[16] R.P. Loce, P.G. Roetling & Y. Lin. 1999. Digital halftoning for printing and display of electronic images. Ch. 7 in E. Dougherty (ed.), *Electronic Imaging Technology*, SPIE Press.

[17] N.S. Nikolov, A. Tarassov & J. Branke. 2005. In search for efficient heuristics for minimum-width graph layering with consideration of dummy nodes. *ACM J. Exp. Algorithmics*, *10*, 1-27.

[18] J. Pujola, F. Martínez-Verdúb, M.J. Luquec, P. Capillac & M. Vilasecaa. 2004. Comparison between the number of discernible colors in a digital camera and the human eye. Proc. European Conf. Colour Graphics, Imaging and Vision (CGIV), 36-40.

[19] P. Rheingans. 1999. Task-based color scale design. Proc. AIPR Workshop: 3D Visualization for Data Exploration and Decision Making, SPIE Vol. 3905, 35-43.

[20] K. Ryall, J. Marks & S. Shieber. 1997. An interactive constraint-based system for drawing graphs. Proc. ACM UIST, 97-104.

[21] D. Schaffer, Z. Zuo, S. Greenberg, L. Bartram, J. Dill, S. Dubs & M. Roseman. 1996. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM TOCHI*, *3*, 2, 162-188.

[22] Z. Shen & K.-L. Ma. 2007. Path visualization for adjacency matrices. Proc. Eurographics/IEEE EuroVis.

[23] V. Smith & J. Pokorny. 2003. Color matching and color discrimination. In S.K. Shevell. *The Science of Color.* Opt. Soc. Am.

[24] K. Sugiyama, S. Tagawa & M. Toda. 1981. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man, & Cybernetics*, *11*, 109-125.

[25] J.N. Warfield. 1977. Crossing theory and hierarchy mapping. *IEEE Trans. Systems, Man, & Cybernetics*, *7*, 502-523.

[26] M. Wattenberg. 2006. Visual exploration of multivariate graphs. Proc. ACM CHI, 811-819.

[27] B. Watson, D. Brink, M. Stallmann, R. Devarajan, M. Rakow, T.-M. Rhyne & H. Patel. 2007. Visualizing very large layered graphs with quilts. IEEE Info. Vis. Poster.

Benjamin Watson is Associate Professor of Computer Science at North Carolina State University. His Design Graphics Lab focuses on the creation of meaning in imagery, and spans the intersections between graphics and perception, design, and interaction. His work has been applied in digital entertainment, computer security, financial analysis, education, and medical assessment. Watson co-chaired the Graphics Interface 2001, IEEE VR 2004 and ACM I3D 2006 conferences, and was co-program chair of I3D 2007. Watson is an ACM and senior IEEE member. He earned his doctorate at Georgia Tech's GVU Center.

David Brink has been a software developer at SAS Institute in Cary, NC for 8 years in the Data Visualization department. He works on various types of graphical components such as charting and node link diagrams using ActiveX and Flash technologies. These components are used in various client applications throughout SAS. Previously he worked at IBM, AT&T Bell Laboratories, and other smaller software companies.

Matthias Stallmann is Professor of Computer Science at North Carolina State University. He has been actively involved in a wide variety of research areas including graph algorithms, combinatorial optimization, experimental algorithmics, VLSI design, operations research, graph drawing, and databases. He is an associate editor of the ACM Journal of Experimental Algorithmics and has served on program committees, panels, and working groups related to experimental algorithms. He has supervised 7 PhD students, two of whom are in academic positions; the remainder are in research/development for SAS, IBM, Amazon, Hewlett-Packard, and I2 Technologies. Dr. Stallmann received his PhD at University of Colorado, Boulder, in 1982.

Ravinder Devajaran is Senior Software Manager in the Data Visualization Department at SAS Institute, and has been working on visualization at SAS for the past 18 years. Ravi earned his Bachelors degree in Civil Engineering at Delhi College of Engineering in India. He has a Masters degree in Civil Engineering from Clemson University, and a second Masters in Computer Science from North Carolina State University.

Matt Rakow is an undergraduate in the Computer Science and Applied Mathematics departments at North Carolina State University. He participates in the Computer Science and University honors programs, is a member of the Phi Kappa Phi honor society and the National Residence Hall Honorary, and is a National Merit Scholar.

Theresa-Marie Rhyne is Director of the Center for Visualization and Analytics in the Department of Computer Science and the Director of the Renaissance Computing Institute's Engagement Center at North Carolina State University. She is also the Visualization Viewpoints Editor for IEEE Computer Graphics & Applications Magazine and serves on the magazine's editorial board. Theresa-Marie served as Eurographics 2001 Papers Co-Chair, lead co-chair of IEEE Visualization 1998 and Panels Chair for SIGGRAPH 1996. She is a Senior Member of ACM and a Senior Member of IEEE and the IEEE Computer Society. She has the BS, two MS and the Degree of Engineer in Civil Engineering from Stanford University.

Himesh Patel's SAS career started with testing of their graphics products. For past 10 years, he has managed and directed SAS Institute's Data Visualization Department. Himesh has a Bachelors degree from North Carolina State University's Computer Science department.