Compressed Adjacency Matrices: Untangling Gene Regulatory Networks

Kasper Dinkla, Michel A. Westenberg, and Jarke J. van Wijk

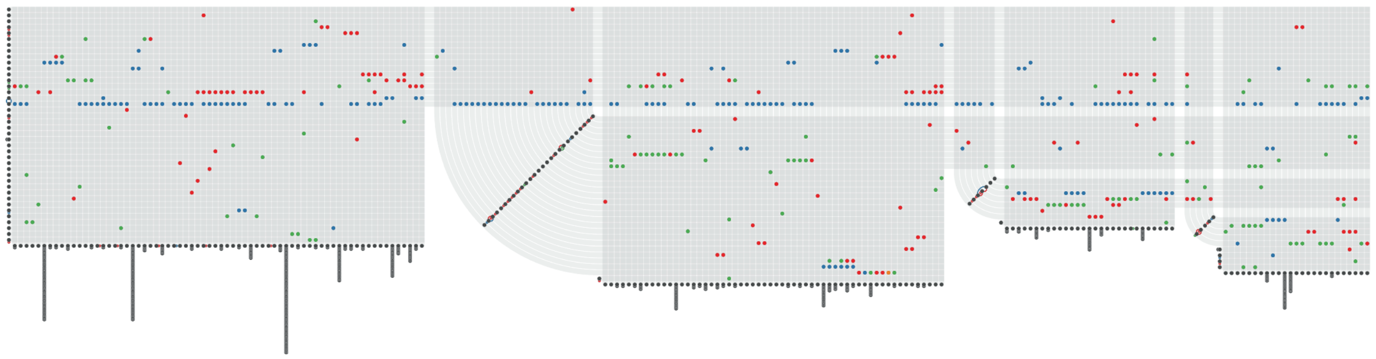


Fig. 1. Compressed adjacency matrix of the gene regulatory network of *Bacillus subtilis*, which consists of approximately 700 genes and 1000 regulations [30].

Abstract—We present a novel technique—Compressed Adjacency Matrices—for visualizing gene regulatory networks. These directed networks have strong structural characteristics: out-degrees with a scale-free distribution, in-degrees bound by a low maximum, and few and small cycles. Standard visualization techniques, such as node-link diagrams and adjacency matrices, are impeded by these network characteristics. The scale-free distribution of out-degrees causes a high number of intersecting edges in node-link diagrams. Adjacency matrices become space-inefficient due to the low in-degrees and the resulting sparse network. Compressed adjacency matrices, however, exploit these structural characteristics. By cutting open and rearranging an adjacency matrix, we achieve a compact and neatly-arranged visualization. Compressed adjacency matrices allow for easy detection of subnetworks with a specific structure, so-called motifs, which provide important knowledge about gene regulatory networks to domain experts. We summarize motifs commonly referred to in the literature, and relate them to network analysis tasks common to the visualization domain. We show that a user can easily find the important motifs in compressed adjacency matrices, and that this is hard in standard adjacency matrix and node-link diagrams. We also demonstrate that interaction techniques for standard adjacency matrices can be used for our compressed variant. These techniques include rearrangement clustering, highlighting, and filtering.

Index Terms—Network, gene regulation, scale-free, adjacency matrix.

1 Introduction

A network is a powerful abstraction that consists of elements and connections between these elements. Many common-day phenomena can be encoded as a network. People and their social contacts, for example, are summarized as social networks and subsequently studied in the scientific domain of sociology [18]. The bioinformatics domain makes heavy use of networks to make sense of interactions between organic compounds. However, large networks are hard to visualize. Standard network visualization methods, the node-link diagram and adjacency matrix, are popular due to their intuitive simplicity and their generic applicability (see Fig. 2). However, they are unable to scale to complex networks, leading to the infamous hairballs in case of node-link diagrams. Recent developments in the field of network visualization move towards the exploitation of predetermined network characteristics in order to create visualizations that are more effective for analysis by domain experts. We present such an approach, which has been specifically designed for gene regulatory networks of bacteria.

The gene regulatory network (GRN) of a bacterium describes interactions between a gene product (a regulator protein) and its target genes. Genes are specific pieces of DNA that form the blueprint for the creation of proteins. A regulator gene, a gene that codes for a regulator protein, can either promote (further) or inhibit (impede) other genes. Sometimes, a gene both inhibits and promotes, depending on environmental conditions. This regulation (or control) between genes is described by a GRN, where genes and regulations are represented as nodes and edges, respectively.

Because GRNs describe part of a bacterium's internal mechanics, domain experts study this type of network intensively, by making use of standard network visualization and layout methods. However, the specific structural properties of GRNs make visualization difficult, even when the network is small. This difficulty is caused by the following characteristics [2]:

Low in-degree Every gene is regulated by only a few other genes. Therefore, all nodes of the network have a low in-degree.

Scale-free out-degree There are few genes that regulate many others, and many genes that regulate few others. Therefore, the network's out-degree distribution follows a power law.

Few cycles The network has few cycles because genes rarely (indirectly) regulate each other both ways.

- *Kasper Dinkla is with Eindhoven University of Technology, e-mail: k.dinkla@tue.nl.*
- *Michel A. Westenberg is with Eindhoven University of Technology, e-mail: m.a.westenberg@tue.nl.*
- *Jarke J. van Wijk is with Eindhoven University of Technology, e-mail: vanwijk@win.tue.nl.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

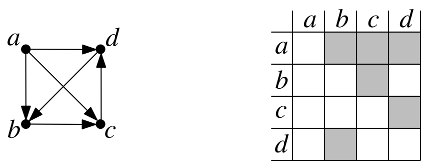


Fig. 2. Two widely-used visualization techniques. Example of a node-link diagram (left) and an adjacency matrix of the same network (right).

In this paper, we propose an approach that exploits these structural characteristics, by creating an adjacency matrix that is cut up and rearranged to take up less space. Our novel network depiction benefits bacterial GRN analysis, and has the following properties:

Compactness This enables a detailed overview of the entire network.

Localization of motifs This enables quick detection of subnetworks of interest.

Consistent arrangement This facilitates interaction while preserving visual orientation.

2 Related Work

Much research in the area of network visualization has been performed [22, 42]. In this section, we restrict our overview to related work directly relevant to our proposed approach.

The node-link diagram and adjacency matrix are the most popular for network visualization. Node-link diagrams are—in general—used to depict sparse networks (with a low edge to node ratio), while adjacency matrices are an alternative for dense networks (with a high edge to node ratio) [14]. Node-link diagrams are intuitive, and most people are able to interpret such diagrams without receiving further explanation. Therefore, network visualization tools in the biological domain often use this representation [24, 28, 38, 43]. However, when a network is dense, this intuitiveness is overshadowed by the visual clutter caused by the many edges that have to be drawn. In this case, an adjacency matrix provides a more ordered and therefore easier to interpret depiction, even though an observer will have to become accustomed to the visualization [1, 41].

Optimal drawing of node-link diagrams, corresponding to aesthetic criteria, and the combinatorial aspects of deriving diagrams that adhere to these criteria, is the focus of the *Graph Drawing* domain [5]. Eades was the first to propose a force simulation to lay out graphs [11], after which more elaborate force models were developed [13]. Such force-based layouts are simple and effective. Aesthetic criteria are often used to judge the effectiveness of node-link diagrams and to derive algorithms that generate high-quality diagrams [5]. The number of intersections in a diagram is an important criterion, and therefore many techniques focus on reducing these intersections [9, 10, 39]. Giving edges a more complex geometry than a straight line, enabling the routing of edges, is a way to avoid intersections as well [27].

A simple example of the exploitation of specific network characteristics is that of tree-layout algorithms [34]. Such algorithms generate layouts that emphasize the inherent tree structure. Similarly, DAGs (short for Directed Acyclic Graphs) are given special treatment with Sugiyama's algorithm [40], where the layered structure of a DAG is exploited and emphasized.

The layout of an adjacency matrix is fairly rigid, i.e., nodes and edges are restricted to rows and columns. Most attention has been given to ordering nodes in such a way that patterns in the network structure become apparent. This is known as *rearrangement clustering*, where nodes are placed close to each other in the arrangement when their distance, according to some similarity measure, is small as well [32]. Such approaches take on more complex forms in biclustering [26], which tries to optimize horizontal and vertical ordering of nodes simultaneously. The use of adjacency matrices and corresponding rearrangement techniques is popular in the biological domain, where they appear in the form of *heat maps* [37].

Combining node-link diagrams and adjacency matrices has been explored as well. One way of doing this is to couple a node-link diagram and an adjacency matrix depiction of the same network by coordinated views [20]. Another way is to create a true hybrid, like the NodeTrix technique [21], where dense parts of a network are detected, isolated, and visualized as matrices, which are then reconnected with links that represent the sparse sections of the network. This is another example of a technique that exploits structural characteristics, namely the presence of dense clusters.

An adjacency matrix can also be used when a network forms a DAG. Then, when nodes are ordered to respect the DAG layers, only the top-right triangle of the matrix will contain edge depictions. If the DAG has few edges that skip multiple layers, the resulting matrix can be arranged to form a more space-efficient staircase-like depiction. This property is effectively used in *Quilts* [25], by giving special treatment to the layer skipping edges, turning them into nodes referenced with color codes, which enables further compaction of the adjacency matrix. *Genea Quilts* [6] are a variant of Quilts that exploit properties specific to genealogical networks to achieve similar compression.

Many analysis tasks involve following a path along the edges of a network. In node-link diagrams, edges can be traced easily from node to node, provided they are not overly obfuscated by intersections with other edges. Node duplication involves the placement of multiple copies of the same node, and distributing the edges over these duplicates [19]. This enables greater reduction of edge crossings, creating a cleaner depiction, at the cost of paths becoming harder to trace over the nodes that have been duplicated. This problem also occurs in adjacency matrices, because every node is represented horizontally as well as vertically, for outbound and inbound edges, respectively. Therefore, following a path in an adjacency matrix is notoriously hard [14].

In a biological setting, node duplication is often used for node-link diagrams of metabolic networks [8, 31], where certain nodes have such a high degree that they are either split into many duplicate representations or removed altogether. This is done because some compounds, like water, are involved in so many metabolic processes that they do

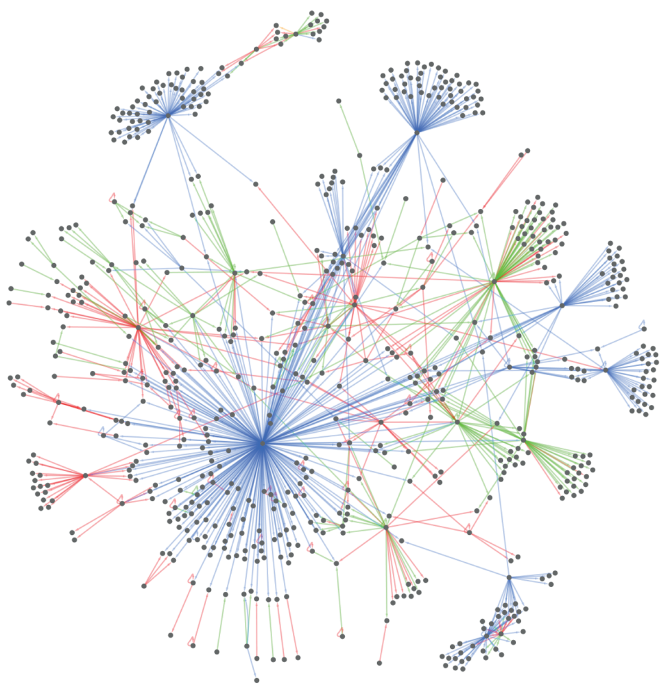


Fig. 3. Node-link diagram of the same GRN as in Fig. 1. Generated with Cytoscape [38], using a standard spring-embedder layout algorithm. Genes are depicted as dark gray nodes and regulations as colored links: green, red, orange, and blue for promotion, inhibition, both, and unspecified, respectively.

not represent information of particular interest to an analyst. The tasks that an analyst has to perform, aided by a network visualization, is of great influence to such trade-offs. In some cases, tracing paths is of no concern and representations more akin to adjacency lists are used, such as animal pedigrees [16]. Here, paths are hard to follow, but it is much easier to compare two nodes (and their attributes) that share an edge. Dealing with the richness of information attached to nodes and edges of networks often leads to these kind of solutions in visual analytics. *Pathline* is another example of this [33], where metabolic pathways are shaped into a vertical linear ordering, making it harder to interpret the pathway, but easier to compare corresponding time-series data that can now be stacked on top of each other in a consistent manner.

Any network visualization is prone to scalability issues. Large graph layout algorithms [17] often result in visualizations that are difficult to interpret. In such circumstances, compression of the network, either lossless or lossy, can be beneficial. Here, lossless compression refers to a visualization that still encodes all network information, but with more complex encoding schemes that require additional effort by an analyst to interpret. For example, visually nesting nodes with edges identical to other nodes (identical neighborhoods) in a joint node can drastically reduce the number of edges that have to be drawn. However, the routing of edges to nested nodes makes the structure of nested nodes more complex. Grouping together children of trees is a simple version of this [15], and an extension of this encoding to multiple levels of nesting has been applied for the analysis of protein interaction networks, forming so-called *Power Graphs* [35]. Lossy compression is usually done by clustering of nodes into meta-nodes, thereby also collapsing edges into meta-edges, where the presence of individual nodes and edges may be lost in the visualization [12]. These kind of approaches are used in interactive settings, where meta-nodes may be expanded and collapsed at will [4].

3 Motivation and Concept

Node-link diagrams are the customary method of inspecting gene regulatory networks in the biological domain, see Fig. 3 for an example. Global features of the network are visible in the diagram, such as the strong interconnectedness of the network, few genes that regulate many other genes, and many genes that are regulated by few genes. More detailed features, such as the immediate neighborhood of a gene, are obfuscated by many links causing substantial overlaps. Yet, context and details are both important to biologists: they require a clear overview of the immediate neighborhood of a gene, but also need to follow paths and determine indirect regulations. Our aim is to facilitate both in the same visualization.

As an alternative visualization, it would be possible to use an adjacency matrix. However, as is clear from Fig. 3, the network contains many nodes but relatively few edges: it is sparse. The resulting adjacency matrix would therefore take up a lot of display space (approximately 700 by 700 rows and columns, respectively), but it would be mostly empty. This makes it hard to see both network context and details at the same time. The specific characteristics of a GRN—low in-degree, power law out-degree, and few and small cycles [2]–however, allow us to introduce a variant of an adjacency matrix that we call a *Compressed Adjacency Matrix* (CAM).

The first two characteristics can be seen clearly in Fig. 3. It is important to note that they cause a strong interconnectedness of the network, which makes it difficult to interpret a node-link representation. However, in combination with the third characteristic, they imply that a GRN has a highly DAG-like structure that consists of few layers. Moreover, because of the second characteristic, there are relatively few *roots*, nodes with no inbound edges; few *hubs*, nodes with both inbound and outbound edges; and many *leaves*, nodes with no outbound edges. This is essential to the construction of our CAM, and they are crucial to obtain its regular structure (see Fig. 1).

The interpretation of a CAM is straightforward, but, like a standard adjacency matrix, a CAM is somewhat less intuitive than a node-link diagram. Fig. 4(c) illustrates how to read a CAM: The outbound and inbound edges of a node are found by looking to the right and upwards, respectively, while using the underlying thick lines as guides.

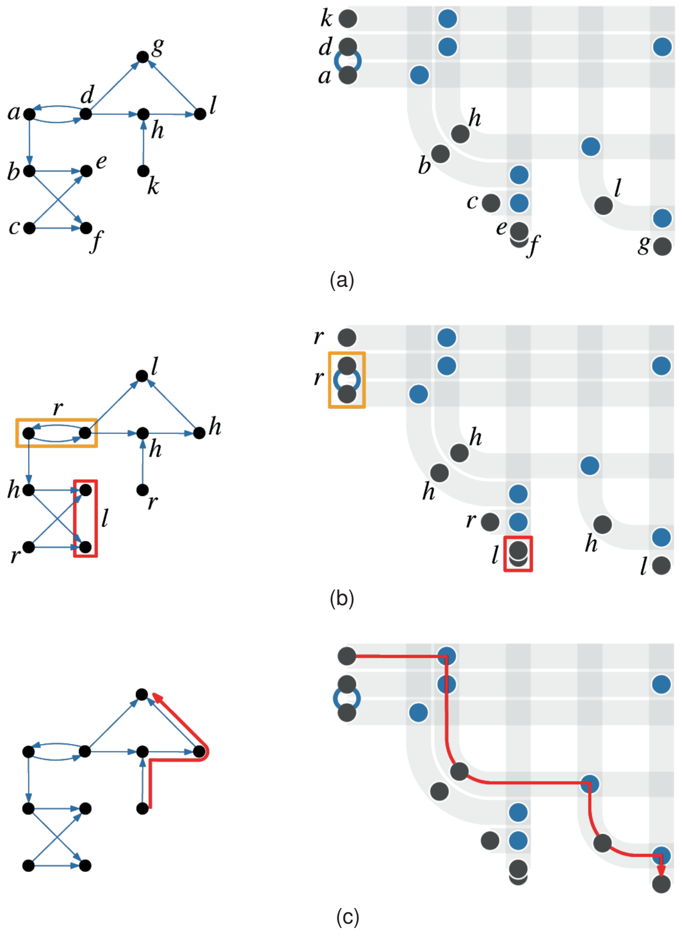


Fig. 4. Node-link diagrams of a simple network on the left and corresponding CAMs on the right: (a) Nodes are uniquely labelled and edges colored blue in both representations, the CAM shows edges as arcs or dots. (b) Cycles are grouped in the layout of a CAM (orange), and nodes with identical neighborhoods are stacked (red), roots are placed before edges (labelled r), hubs diagonally (labelled h), and leaves at the bottom (labelled l). (c) To follow a path in a CAM, one starts at a node and traces a line (red) to the right until an edge is hit, then one traces a line down until a node is hit again.

From Fig. 4, several beneficial properties of CAMs become apparent. First, the entire neighborhood of a node is easy to find by following its underlying line. Second, nodes with identical neighborhoods are localized as stacks that save space (lossless compression). Third, the few and small cycles in the network are localized and give the impression of an actual cycle, due to the edges of the cycles being drawn as arcs. Fourth, the layers of the network's DAG-like structure, and regulations between these layers, are compartmentalized and visible as contiguous surfaces. Finally, paths can be followed by tracing the underlying thick lines through nodes, which is usually easy in node-link diagrams as well, but not in standard adjacency matrices.

4 Approach

Though CAMs have a tidy appearance and standard adjacency matrices are easy to derive from CAMs, the conversion of a network to a CAM is not trivial and consists of six steps. First, the network is decomposed into weakly connected components, after which every component is treated as a separate network for the remaining steps. Second, nodes with identical neighborhoods are grouped. Third, strongly connected components are detected and grouped to form a DAG. Fourth, the nodes of the DAG are partitioned into layers such that all edges have the same direction with respect to the layers. Fifth, the lay-

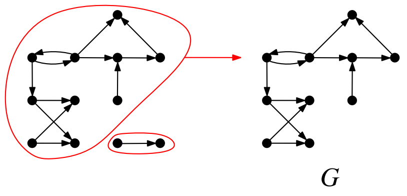


Fig. 5. Example network (left) with its weakly connected components marked in red. In this case, the largest component is taken as G (right).

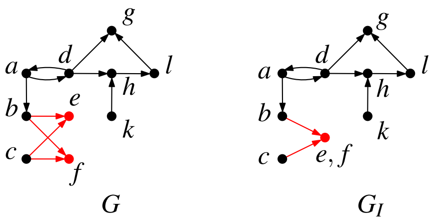


Fig. 6. Example of G (left) and the G_I (right) that is derived by grouping vertices with identical neighborhoods. The vertices and edges in red are joined into a larger group.

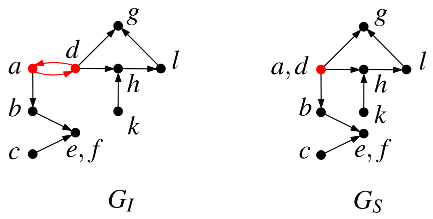


Fig. 7. Example of G_I (left) and the G_S (right) that is derived by grouping SCCs. The red vertices and edges are part of an SCC that is larger than a single vertex.

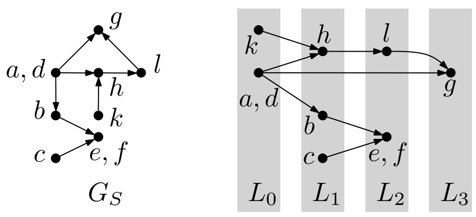


Fig. 8. Example of G_S (left) and its layering (right).

ers are turned into blocks that form the backbone of the CAM, where nodes are partitioned into five classes that dictate their arrangement in the CAM. In addition, grouped strongly connected components are split into separate nodes again while maintaining the vertex arrangement dictated by the blocks. Finally, the blocks are concatenated to form a cascade from which node positions and the CAM visualization are derived in various styles. We describe each step in detail and illustrate it with a running example.

4.1 Network

The network is a directed graph $G = (V, E)$, where V is the set of vertices (nodes) and E is the set of *directed* edges between vertices of G. We assume that G is weakly connected, i.e., every vertex is reachable from any other vertex when edge directions are ignored. If a network is not weakly connected, it is decomposed into weakly connected components with a connectivity search. Each weakly connected component is then taken as G and converted to a CAM (see Fig. 5). The individual CAMs are concatenated in the final visualization.

The vertices in our network have a number of attributes, such as the gene's name and annotation of gene function. The edges have a single attribute that describes the type of interaction between a gene and its target, namely *promotion*, *inhibition*, *both*, and *unspecified*. However, the exact nature of the attributes is irrelevant to the construction of a CAM.

4.2 Identical neighborhood grouping

Biological networks contain a lot of vertices with an identical neighborhood. Grouping these vertices reduces the complexity of the network, while still leaving open the possibility to properly convey each vertex of a group in the final visualization. To this end, we define a grouped version $G_I = (V_I, E_I)$ of G, where V_I is the set of vertices of G_I that represent non-overlapping subsets of V with identical neighborhoods, and E_I is the set of directed edges of G_I.

Here, it is important to note that we take the edge attribute (describing the type of regulation) into account: only edges with the same value for the attribute are combined into one. An example is shown in Fig. 6, in which the vertices e and f both receive input from vertices b and c, thus their neighborhoods are considered identical. Such vertices are stacked on top of each other in the final visualization, cf. Fig. 4(b).

4.3 Strongly connected component grouping

We need to remove all cycles of G_I to turn it into a Directed Acyclic Graph. Along the same lines of the Sugiyama algorithm [40], we could temporarily invert those edges that break a cycle, to be reverted again in the final visualization with a special edge depiction. However, a better alternative is available to us because we know that G and therefore G_I has few and small cycles. The small cycles imply the existence of small *strongly connected components (SCC)*, i.e., sub-graphs where any vertex is reachable from any other vertex.

The SCCs of G_I are a partition of V_I. By grouping every SCC into a single vertex, as shown in Fig. 7, we create a DAG $G_S = (V_S, E_S)$. The set of vertices V_S represents SCCs of G_I, and the set E_S is the set of directed edges between vertices in G_S. The edges E_S are not necessarily the same as the edges E_I. They represent an aggregation, which means that edges of E_I with different attribute values can be represented by a single edge of E_S. This is not important for the construction of the CAM, because it relies only on the structure of G_S.

4.4 Layers

Graph G_S is a DAG and can therefore be regarded as a layered structure (see Fig. 8). We determine its layers L_i, $i = 0, 1, \ldots, m$, which form a partition of V_S, in a similar way as Sugiyama's algorithm [40]:

The edges that run between layers all have the same direction. Also, no edge runs between two vertices of the same layer. The vertex configuration is as compact as possible: if a vertex $v \in L_i$ has outbound edges, then there exists $v' \in L_{i+1}$ and an edge from v to v', or there exists $v' \in L_{i-1}$ and an edge from v' to v. In other words, vertices immediately precede their successors in the DAG if there are any successors, or immediately succeed their predecessors otherwise. This is important for the derivation of blocks from layers in a later stage.

4.5 Blocks

The layered node-link diagram of Fig. 8 already conveys the structure of the network quite well. However, for large networks, the many edge intersections between layers make it hard to interpret. We therefore transform the layers into blocks, which we arrange in such a way that empty space can be used effectively to obtain a compact visualization.

Layers map directly to blocks, i.e., a block B_i is derived from its corresponding layer L_i. Block B_i consists of sequences of vertices, H_i and V_i, that specify the horizontal and vertical ordering of L_i's vertices in the CAM, respectively. An appropriate ordering of the vertices in H_i and V_i is crucial to obtain the desired compactness. We achieve this by partitioning the vertices of L_i into five classes:

Leaf Vertex in L_i without successors.

Short root Vertex in L_i that has a successor but no predecessors and all successors are leaves in L_{i+1}.

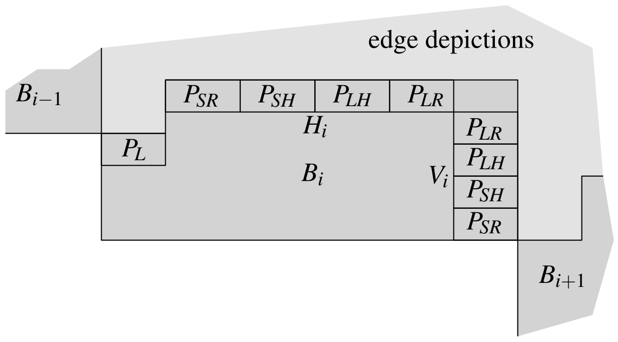


Fig. 9. The composition of B_i and its placement with respect to its neighboring blocks.

Long root Vertex in L_i that has a successor but no predecessors and is not a short root.

Short hub Vertex in L_i that has a predecessor and successor, and all successors are leaves in L_{i+1}.

Long hub Vertex in L_i that has a predecessor and successor, but is not a short hub.

The partition that we get for L_i then consists of leaves P_L, short roots P_{SR}, long roots P_{LR}, short hubs P_{SH}, and long hubs P_{LH}. Each of these sets can be ordered individually, based on attributes or some other characteristic. This ordering affects the readability of the CAM, and we will come back to this in Section 4.6. Once we have the partition, we can construct H_i and V_i by concatenation:

$$H_i = [P_L,\ P_{SR},\ P_{SH},\ P_{LH},\ P_{LR}]$$
$$V_i = [P_{SR},\ P_{SH},\ P_{LH},\ P_{LR}]$$

The construction of H_i and V_i allows us to make a spatial configuration of B_i as illustrated in Fig. 9. The blocks $B_1, B_2, ..., B_m$ form a cascade with enough space above it for edge depictions (see Fig. 10(b)). Most vertices of L_i occur twice in this configuration. The vertices of H_i have depictions above them to represent inbound edges from predecessors (in the light gray zone). Similarly, vertices in V_i have depictions to their right to represent outbound edges to successors. This explains why P_L is missing from V_i: it consists of leaves that have no outbound edges.

The presence of P_{SR} in H_i is optional. Removing it will result in a more compact but less consistent CAM, which we will discuss in Section 4.6. Likewise, the configuration of B_{i-1} and B_i is made more compact by shifting down P_L. This is possible, because vertices of P_{SR} and P_{SH} in B_{i-1} have only outbound edges to P_L of B_i. Therefore, no space is required for edge depictions beyond P_L.

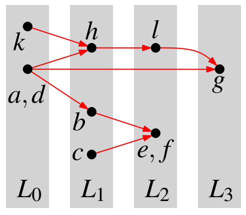
From the arrangement that we now have, we can actually construct a standard adjacency matrix, see Fig. 10(c). This can be done by shifting the vertices that belong to some H_i upwards to a common horizontal axis, and shifting those vertices that belong to some V_i to a vertical axis at the left (while respecting their block-induced arrangement).

Finally, the grouped strongly connected components V_S are flattened within H_i and V_i. The sets of identical neighborhood vertices are maintained in the layout, and drawn as stacks of vertices. In addition, edges in E_I that connect vertices of a strongly connected component are represented by arcs, which forms the cycle shown in Fig. 4.

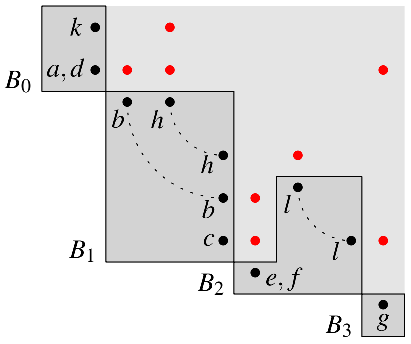
For the small example network, clearly not much compression can be achieved. However, the teaser image and the image discussed in Section 5 show that our approach is effective for compressing large GRNs.

4.6 Visualization

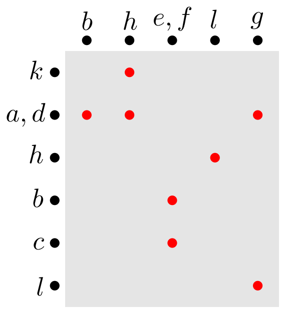
Our approach creates a layout that consists of inbound and/or outbound positions for every vertex if it has an in and/or outbound neighborhood, respectively. This provides us with enough information to create a CAM visualization. As in Fig. 10, edge depictions are placed in a grid, outbound from the vertex to its direct left and inbound to the vertex directly below it. We give a special treatment to hubs because they have inbound and outbound edges by definition. In standard adjacency matrices, they therefore appear both at the left and at the top of the matrix, spaced wide apart, making it difficult to trace a path. In contrast, we place each hub at the middle of an arc that extends from the hub's inbound neighborhood to its outbound neighborhood. This special treatment of hubs is enabled by their carefully chosen arrangement (cf. Fig. 9).



(a)



(b)



(c)

Fig. 10. (a) Layers of G_S. (b) Blocks of G_S that form a CAM. (c) Standard adjacency matrix arranged according to the blocks of G_S. All edge depictions are in red. Here e,f and g are leaves, c is a short root, k and a,d are long roots, b and l are short hubs, and h is a long hub.

Styles

Like for a standard adjacency matrix, the grid drawn in the background is a visual aid that more strongly associates a vertex depiction with edge depictions of its in- and/or outbound neighborhoods (and vice-versa). Even with this aid, adjacency matrices are hard to interpret when they are large. We have therefore experimented with various visual styles to improve interpretation. A sample of style combinations is shown in Fig. 11.

The first style is a plain grid, where grid cells are separated by solid lines (see Fig. 11(a)). These lines help to inspect the neighborhood of a vertex, but they have to be given a dark color to be visible due to their small width. This creates many strong brightness transitions in the visualization, impeding its aesthetics.

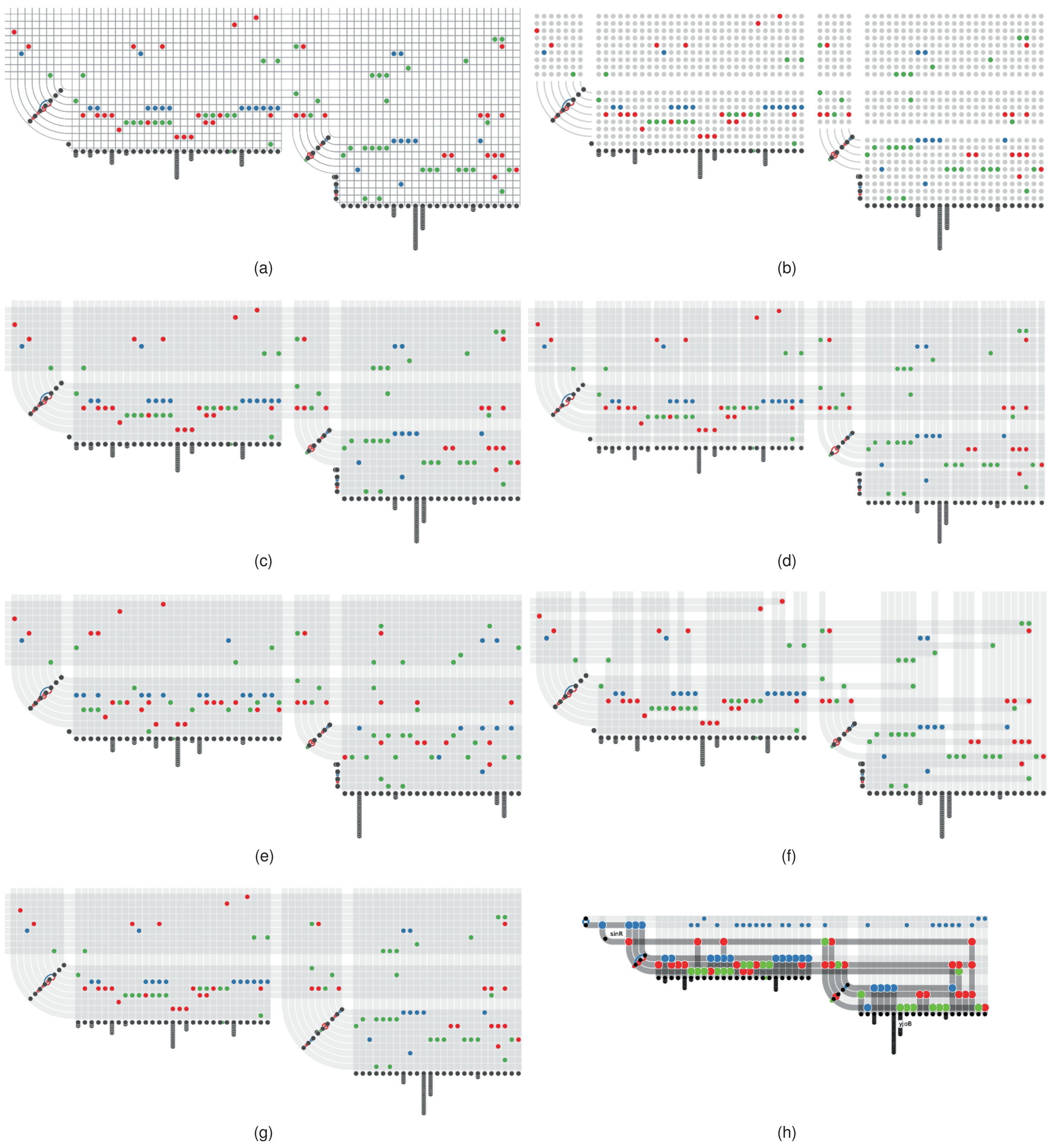


Fig. 11. A sample of CAM configurations, applied to a section of Fig. 1: (a) Thin separating lines. (b) Non-regulations as gray dots. (c) Thick underlying lines. (d) Periodic spacing. (e) Leaves have no rearrangement clustering on neighborhood similarity. (f) Thick underlying lines are shortened. (g) Roots are treated as hubs. (h) Filtered and highlighted on two selected genes.

The second style circumvents the need for explicit lines by introducing gray edge depictions for those vertex pairs that have no edges between them (see Fig. 11(b)). This generalization creates a grid pattern that guides the observer as well. However, the presence of these non-edge depictions draws away attention from the actual edges, making it harder to get an impression of the network's connectivity. Moreover, a connection between the neighborhoods of hubs still has to be made explicit, which is done with a thick arc that does not integrate well with the visual style of the rest of the CAM.

In our opinion, the third style is the most effective and therefore default style of the prototype. Instead of drawing lines between the edge depictions, thick and translucent lines are drawn behind them (see Fig. 11(c)). Vertical and horizontal lines appear darker where they intersect due to their translucency, as though they are plies that are stacked on top of each other. This makes every line of a vertex stand out and better to trace, e.g., it is easy to follow a line down from an edge, along an arc, to the depiction of the hub that the edge is directed to, and even further along the arc, into the hub's outbound neighborhood. Thus, paths in the network are visually apparent and can be followed. Note that this is not possible in a standard adjacency matrix. The other styles also enable path following, but the neighborhood of a vertex does not appear as a clear contiguous area.

Even with the aid of a grid, it is still hard to find all edges of a vertex's neighborhood because it is easy to accidentally skip a line over a long distance. This is similar to drifting off direction when there are no lines guiding you. One way to suppress this effect is to add more structure to the grid, providing additional landmarks to guide an observer. A periodic change of grid cell color is commonly used to get this effect in adjacency matrices and tables. However, we want to restrict the use of colors, because we already use colors to encode edge types. Instead, extra space can be added between vertices to create a visual grouping of cells. This makes skipping a line harder, but also causes the visualization to be less tidy (see Fig. 11(d)). Likewise, space between blocks and different classes of vertex can be added if desired.

We have also experimented with minimizing line length while making sure the lines still contain all edges (see Fig. 11(f)). This makes it easier to determine the extent of a neighborhood. It also makes the visualization less structured, and implies the presence of edges (possibly non-existing) at line intersections, because the immediate neighborhood is darker. Similarly, the inbound neighborhood line of a root is hidden by default because it wastes space (it has no edges to guide the observer to). Including these lines may be desirable, however, to get a more consistent visualization (see Fig. 11(g)).

Color

All visual components of the CAM are gray scale except for the edges. The number of grays used is kept at a minimum, because this draws attention to the edges. This also leaves the rest of the color space available to color code edge types such that they are easy to distinguish. Green and red colors encode regulations that promote and inhibit, respectively, because it is a custom in the target domain. Orange encodes regulations that both promote and inhibit, because it is approximately an intermediary for green and red, yet can still be distinguished as a separate color. Blue encodes unspecified regulations, because it has a neutral connotation.

Vertices are colored dark gray—not black—to make them less dominant. They are also given halos matching the background color (white) to improve contrast, and to make them distinguishable when partially stacked. Likewise, edges and text labels are given halos.

Arrangement

Various vertex arrangements are implemented in the prototype, i.e., the sets in the partition of a layer L_i (cf. Section 4.5) can be ordered in various ways. Simple arrangements like sorting by in- and out-degree are possible, in addition to rearrangement clustering by neighborhood similarity. Similarity clustering is widely used to bring out edge patterns in standard adjacency matrices [20], and by comparison of Fig. 11(c) and (e) it is also beneficial to CAMs of GRNs. The use of arrangements is configurable in the prototype, but by default we arrange leaves by inbound neighborhood similarity, and hubs and roots by out-degree. Switching arrangements can be done interactively, where vertices transition smoothly to their new positions.

Vertex arrangement in CAMs is more restricted than in an adjacency matrix due to vertices being part of blocks and their partitions. It is also possible that leaves with similar neighborhoods end up in different layers of the DAG and thus in a different block because one inbound edge is different, causing a large distance between the leaves in the final visualization.

Interaction

Hovering over vertices and edges is possible for highlighting and obtaining additional information, i.e., gene name and function description for hovered vertices, and inbound and outbound gene names for hovered edges. In addition, two types of connectivity search are supported: highlighting of direct neighborhood and highlighting of the entire up- and down-directed section of the network, i.e., all genes that (in-)directly influence or are influenced by the hovered gene (see Fig. 11(h)). Moreover, multiple vertices can be selected such that any highlighting is kept in place when the vertices are no longer being hovered.

When a vertex or edge is highlighted, its size, color saturation, and brightness is increased, such that it stands out from its surroundings. The background grid lines are darkened as well to provide extra guidance. As shown in Fig. 11(h), the enlarged edges also help to distinguish between those edges that are part of the connectivity search, and those that are part of the neighborhood of a connected vertex.

Highlighted vertices and edges attract attention, but the remainder of the network may inhibit their inspection. The prototype therefore allows to filter those vertices and edges that are not highlighted. This causes a considerable reduction in size of the CAM, making it easier to interpret (see Fig. 11(h)). Moreover, it enables interactive navigation of the network, because nodes can be added or removed from the selection, increasing or decreasing parts of the network that are highlighted and therefore visible. To better facilitate this navigation, the transition between filtered CAMs is animated, and the underlying block arrangement is maintained to preserve the observer's visual orientation. This means that the entire network is always the basis for the generated CAM, so the DAG structure that leads to layers and blocks is stable, and vertices do not switch blocks, regardless of filtering.

5 Discussion

We will now compare CAMs to standard visualization techniques that are often used for GRN analysis. Fig. 12 and 13 show the CAM and the node-link diagram of a GRN of the bacterium *Escherichia coli*, which consists of approximately 1300 genes and 2800 regulations [36].

Node-link diagrams are more intuitive than CAMs, but in case of Fig. 13, there are so many edge intersections that the diagram is hardly readable. Some depictions are even fully occluded, leading to a loss of information. More advanced node-link encodings [7], or layout techniques that are specific to scale-free networks [3], may create more insightful visualizations. However, many edge intersections cannot be avoided due to the interconnectedness of the GRN. Edge-bundling techniques could alleviate this further [23], but at the risk of additional information loss.

The main disadvantage of CAMs is inefficient use of space. Node-link diagrams can be more efficient in this aspect, because the positioning of nodes is less restricted. Even after compression, it can be seen in Fig. 12 that large sections of the CAM remain unused, and that it has a high aspect ratio. However, the neatly-arranged visualization and lack of edge overlaps provided by a CAM outweigh these disadvantages.

GRN analysts have specific needs that have to be taken into account as well, which involve the search of patterns. These patterns come in the form of subnetworks with a specific structure. Some instances of these subnetworks are called *motifs*, which are subnetworks that are statistically over-represented in GRNs, and are known to have a specialized function [2]. We therefore consider the most important subnetwork structures, corresponding motifs, and related generic network tasks [29].

Genes should be regarded as part of a dynamical system to understand why specific subnetwork structures have specialized function. Genes have a so-called *level of expression* that is conceptually related to the extent at which the gene is involved in protein production or the regulation of other genes. It is also possible to talk about *regulation signals*, because regulations are dependent on gene expression levels

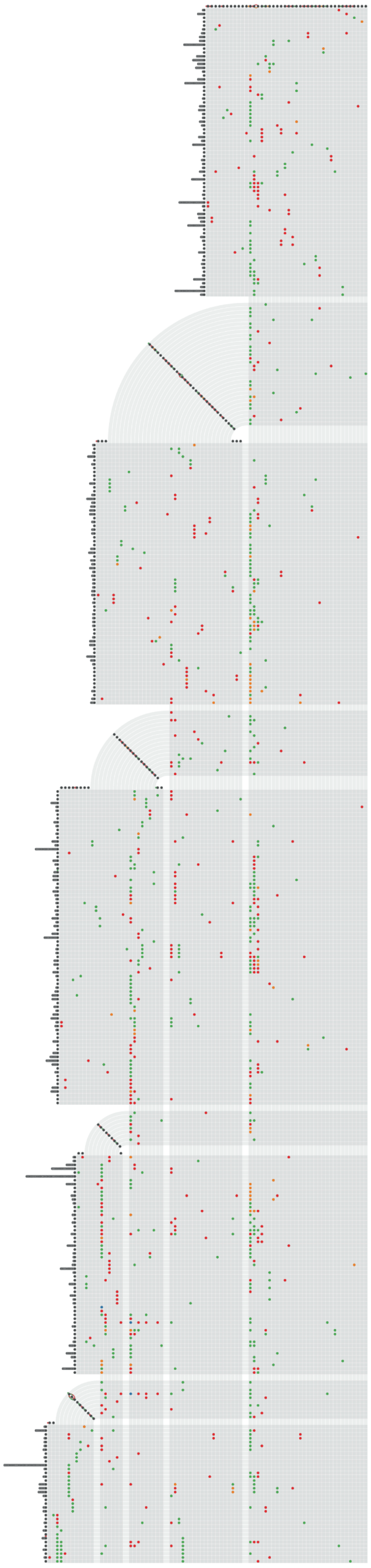


Fig. 12. Rotated CAM of the GRN of bacterium *Escherichia coli*.

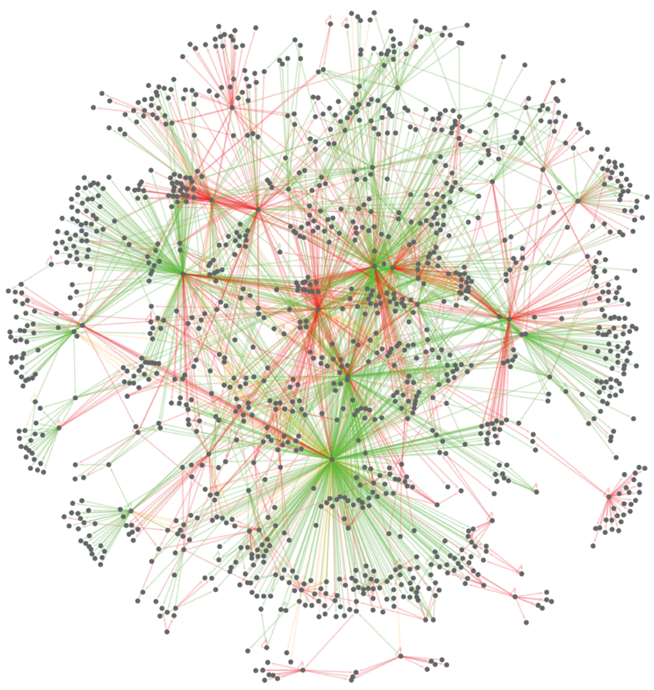


Fig. 13. Node-link diagram of the GRN of *Escherichia coli*, laid out with a spring-based method in Cytoscape as commonly used by GRN analysts.

and are therefore time dependent. The motifs that we discuss can thus be regarded as signal processing components, acting as signal delays, filters, and pulse generators.

Self-edge Single vertex with an edge directed to itself.

This structure is of particular interest in GRNs because a self-inhibiting gene (or the *negative auto-regulation* motif) likely shows a faster response to inbound regulation signals, and is more stable when receiving fluctuating signals [2]. Likewise, a self-promoting gene (or the *positive auto-regulation* motif) shows slower response to inbound signals. Self-edges are easily spotted in both Fig. 12 and Fig. 13 as curves that bend back into a node. This also makes self-edges localized such that they are easily spotted as a part of another subnetwork of interest.

Out-fan Vertex with edges directed towards a set of vertices.

Inspecting an out-fan of a vertex requires finding its neighborhood, the *adjacency* task of [29]. This is relevant for gaining insight about *regulators*, which are genes with a large outbound neighborhood, referred to as a *regulon*. These regulators play a dominant role in a bacterium's response to environmental conditions, acting as master switches for parts of a GRN. A regulator is easily spotted in a node-link diagram by the many edges that converge at its position, creating the appearance of an actual fan. Likewise, a regulator is easily spotted in a CAM because it is either a root or a hub, and these vertex classes are easily distinguished. A regulator also tends to be in a higher layer and has large strips of edge depictions to its right that is emphasized by neighborhood similarity clustering. In addition, the ordering of hubs and roots by out-degree places regulons close to each other per layer, making them easier to spot. Surveying the entire neighborhood of a regulon, however, is difficult in node-link diagrams because direct neighbors can be positioned anywhere, and individual edges are hard to spot because of the many edge intersections. While neighbors have more regular positions in a CAM, making them easier to find, they can be spread out over longer distances as well.

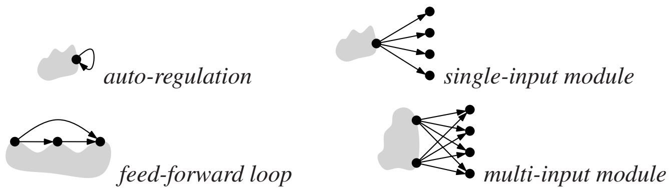


Fig. 14. Illustrations of GRN motifs, generalized of regulation types. Gray areas represent the remainder of a network.

The *single-input module* motif is a form of out-fan where the genes of a regulon are affected by only one regulator (see Fig. 14). Here, a temporal arrangement of gene activations occurs by varying the regulation strength to each gene of the regulon, in essence creating a sequential program that is executed when the regulator becomes active. Single-input modules are spotted in a node-link diagram by their dense arrangement around a regulator, provided that it is not obfuscated by edges that are not part of the module. Stacked leaves in a CAM, with only one inbound edge, correspond directly to a single-input module. For example, it can be seen that *Escherichia coli* has many such modules of varying size spread out over all layers, while *Bacillus subtilis* (see Fig. 1) has relatively few large modules that are mostly part of the first layers.

In-fan Vertex with edges directed towards it from a set of vertices.

The observations for the out-fan are symmetric to those of the in-fan. However, the in-fan is of relevance to the *multiple-input module* motif, which is an extension of the single-input module where multiple regulons have strong overlap (see Fig. 14). This enables the execution of different sequential programs with the same genes. The special arrangement of vertices in CAMs clearly reveal multiple-input modules, where the out-degree ordering of roots and hubs pushes regulators close together and the neighborhood similarity arrangement of leaves show overlap—and differences—between regulons as thick stripes of edge depictions. Regulon overlap can be detected in node-link diagrams as two fans that diverge to the same set of nodes if only two regulators are involved. However, involvement of more than two regulators results in patterns that are hard to discern. Moreover, it is harder to compare two or more regulons in the node-link diagram of Fig. 13 than in the CAM of Fig. 12.

Path Multiple vertices connected such that they form a directed chain.

A path in a GRN is effectively an indirect regulation of the gene at the end of the path by the gene at the beginning of the path. Likewise, a regulator may have greater effect on the entire GRN than initially estimated from its regulon, because it affects even more genes via paths through its regulon. This stipulates the desire for focus (e.g., fans) and context (e.g., paths). Tracing a path is called *follow path* in [29].

There may exist multiple paths between two genes. Such regulatory paths tend to pass signals at different speeds, especially when the lengths of the paths are unequal. The relevance of this follows from *feed-forward loop* motifs. Feed-forward loops consist of three genes with one direct and one indirect regulation (see Fig. 14). Its behaviors are the filtering of pulses (brief inbound regulation signals), the conversion of long inbound regulation signals to outbound pulses, and the shortening or elongation of response to inbound signals similar to auto-regulation.

Following paths is already difficult in the node-link diagrams of Fig. 3 and 13, and spotting two paths between two nodes even more so. This is not easy in CAMs either, but feed-forward loops can still be discerned because the nodes of the paths follow each other in the layering and therefore create consistent patterns. These patterns are made explicit by highlighting and filtering, as seen in Fig. 11(h).

Cycle Multiple vertices connected such that they form a cycle.

Cycles are also known as *feed-back loops* but they are uncommon in GRNs, and therefore not considered to be motifs. However, genes that are part of cycles have strongly associated behavior, because these cycles tend to be small. For example, two genes that inhibit each other form a cycle that functions as a form of indirect auto-regulation on both genes. In that sense, genes that are part of a cycle behave in unison.

The presence of cycles is clearly lost in large node-link diagrams, where the nodes of a cycle can have large distance between them, and the links of the cycles are obfuscated by other links. Cycles are localized in CAMs and have the appearance of actual cycles, making them easier to spot. Moreover, the complementary nature of genes that are part of a cycle also becomes apparent, because their neighborhood edges are placed close to each other in the matrix. This enables easy comparison of their neighborhoods (see the right-most cycle in Fig. 12).

Clearly, there are other visualization techniques beyond node-link diagrams that can be considered, foremost of which are adjacency matrices and more advanced visual encodings. Adjacency matrices, however, are almost identical to CAMs but take up a lot more space and do not facilitate following a path. The more advanced visual encodings focus on networks that have structural characteristics different from those of GRNs. NodeTrix [21], for example, is effective for networks that have somewhat isolated clusters, which does not match the strongly interconnected nature of GRNs that follows from their scale-free out-degree distribution. Moreover, Quilts [25] are designed for DAG-like networks but rely on a tidy layer structure where few edges skip layers. As becomes clear from Fig. 12, GRNs have many edges that skip layers, mainly outbound from important regulators, that make Quilts impractical. We have also tried other node-link layout algorithms, but these gave the same, or worse, results as the one shown in Fig. 13.

6 Conclusion

We have presented a new approach for the visualization of GRNs and demonstrated its strengths and weaknesses with respect to finding sub-network structures that are of importance to GRN analysts. Moreover, CAMs have clear benefits over standard adjacency matrices, such as the ability to follow paths, which in some aspects is even easier than in node-link diagrams. The feasibility of applying the CAM technique in practice is shown with a prototype that supports various interactive techniques often used for standard adjacency matrices. The combination of adjacency matrix specific techniques, such as rearrangement clustering, and node-link diagram properties, such as no node duplication, make for a good alternative to current GRN visualizations.

Future work includes the integration of CAMs into the visual analysis process of GRN analysts. This involves the mapping of additional data to vertices, such as gene expression time series, where we can exploit the linear arrangement of vertices in a CAM to improve attribute comparison tasks. We also want to investigate further matrix compression, for example, by arranging leaves by neighborhood size and pushing sections of them upwards where possible. In addition, the layers of a CAM could be made to branch out, in accordance with a possible branch-like structure of the network itself.

For a CAM to achieve an extensive compression, the visualized network has to fulfill specific requirements. So far we have not encountered other types of network that fulfill these requirements. However, we plan to investigate methods to make CAMs applicable to a broader class of networks. Case in point is an alternate method to deal with cycles, i.e., inverting edges instead of grouping strongly connected components. This means that the few and small cycles requirement can be dropped, but likely at the cost of less intuitive visualizations.

Acknowledgments

Kasper Dinkla is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.004.

REFERENCES

[1] J. Abello and F. van Ham. Matrix Zoom: A visual interface to semi-external graphs. In *IEEE Proc. Symposium on Information Visualization*, pages 183–190, 2004.

[2] U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. CRC press, 2006.

[3] R. Andersen, F. Chung, and L. Lu. Drawing power law graphs using a local/global decomposition. *Algorithmica*, 47(4):379–397, 2007.

[4] D. Archambault, T. Munzner, and D. Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Trans. Visualization and Computer Graphics*, 14(4):900–913, 2008.

[5] G. Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1998.

[6] A. Bezerianos, P. Dragicevic, J.-D. Fekete, J. Bae, and B. Watson. GeneaQuilts: A system for exploring large genealogies. *IEEE Trans. Visualization and Computer Graphics*, 16(6):1073–1081, 2010.

[7] K. Boitmanis, U. Brandes, and C. Pich. Visualizing internet evolution on the autonomous systems level. In *Graph Drawing*, volume 4875 of *Lecture Notes in Computer Science*, pages 365–376. Springer Berlin / Heidelberg, 2008.

[8] D. Croft, G. O'Kelly, G. Wu, R. Haw, M. Gillespie, L. Matthews, M. Caudy, P. Garapati, G. Gopinath, B. Jassal, S. Jupe, I. Kalatskaya, S. Mahajan, B. May, N. Ndegwa, E. Schmidt, V. Shamovsky, C. Yung, E. Birney, H. Hermjakob, P. D'Eustachio, and L. Stein. Reactome: A database of reactions, pathways and biological processes. *Nucleic Acids Research*, 39(suppl 1):D691–D697, 2011.

[9] W. Didimo, G. Liotta, and S. Romeo. Topology-driven force-directed algorithms. In *Graph Drawing*, volume 6502 of *Lecture Notes in Computer Science*, pages 165–176. Springer Berlin / Heidelberg, 2011.

[10] T. Dwyer, Y. Koren, and K. Marriott. Ipsep-cola: An incremental procedure for separation constraint layout of graphs. *IEEE Trans. Visualization and Computer Graphics*, 12(5):821–828, 2006.

[11] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[12] N. Elmqvist, T. Do, H. Goodell, N. Henry, and J.-D. Fekete. Zame: Interactive large-scale graph visualization. In *IEEE Proc. Pacific Visualization Symposium*, pages 215–222, 2008.

[13] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[14] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE Proc. Symposium on Information Visualization*, pages 17–24, 2004.

[15] M. Graham and J. Kennedy. Exploring multiple trees through DAG representations. *IEEE Trans. Visualization and Computer Graphics*, 13(6):1294–1301, 2007.

[16] M. Graham, J. Kennedy, T. Paterson, and A. Law. Visualising errors in animal pedigree genotype data. *Computer Graphics Forum*, 30(3):1011–1020, 2011.

[17] D. Harel and Y. Koren. A fast multi-scale method for drawing large graphs. In *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 235–287. Springer Berlin / Heidelberg, 2001.

[18] J. Heer and D. Boyd. Vizster: Visualizing online social networks. In *IEEE Proc. Symposium on Information Visualization*, pages 32–39, 2005.

[19] N. Henry, A. Bezerianos, and J.-D. Fekete. Improving the readability of clustered social networks using node duplication. *IEEE Trans. Visualization and Computer Graphics*, 14(6):1317–1324, 2008.

[20] N. Henry and J.-D. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Trans. Visualization and Computer Graphics*, 12(5):677–684, 2006.

[21] N. Henry, J.-D. Fekete, and M. McGuffin. NodeTrix: a hybrid visualization of social networks. *IEEE Trans. Visualization and Computer Graphics*, 13(6):1302–1309, 2007.

[22] I. Herman, G. Melançon, and M. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. Visualization and Computer Graphics*, 6(1):24–43, 2000.

[23] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Visualization and Computer Graphics*, 12(5):741–748, 2006.

[24] Z. Hu, D. Ng, T. Yamada, C. Chen, S. Kawashima, J. Mellor, B. Linghu, M. Kanehisa, J. Stuart, and C. DeLisi. VisANT 3.0: New modules for pathway visualization, editing, prediction and construction. *Nucleic Acids Research*, 35(suppl 2):W625–W632, 2007.

[25] Juhee Bae and B. Watson. Developing and evaluating quilts for the depiction of large layered graphs. *IEEE Trans. Visualization and Computer Graphics*, 17(12):2268–2275, 2011.

[26] G. Kerr, H. Ruskin, M. Crane, and P. Doolan. Techniques for clustering gene expression data. *Computers in Biology and Medicine*, 38(3):283–293, 2008.

[27] G. Klau and P. Mutzel. Optimal compaction of orthogonal grid drawings. In *Integer Programming and Combinatorial Optimization*, volume 1610 of *Lecture Notes in Computer Science*, pages 304–319. Springer Berlin / Heidelberg, 1999.

[28] S. Kozhenkov, Y. Dubinina, M. Sedova, A. Gupta, J. Ponomarenko, and M. Baitaluk. BiologicalNetworks 2.0 – an integrative view of genome biology data. *BMC Bioinformatics*, 11(1):610, 2010.

[29] B. Lee, C. Plaisant, C. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proc. AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, pages 1–5. ACM, 2006.

[30] Y. Makita, M. Nakao, N. Ogasawara, and K. Nakai. DBTBS: Database of transcriptional regulation in *Bacillus subtilis* and its contribution to comparative genomics. *Nucleic Acids Research*, 32(suppl 1):D75–D77, 2004.

[31] L. Matthews, G. Gopinath, M. Gillespie, M. Caudy, D. Croft, B. de Bono, P. Garapati, J. Hemish, H. Hermjakob, B. Jassal, A. Kanapin, S. Lewis, S. Mahajan, B. May, E. Schmidt, I. Vastrik, G. Wu, E. Birney, L. Stein, and P. DEustachio. Reactome knowledgebase of human biological pathways and processes. *Nucleic Acids Research*, 37(suppl 1):D619–D622, 2009.

[32] W. McCormick, P. Schweitzer, and T. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972.

[33] M. Meyer, B. Wong, M. Styczynski, T. Munzner, and H. Pfister. Pathline: A tool for comparative functional genomics. *Computer Graphics Forum*, 29(3):1043–1052, 2010.

[34] E. Reingold and J. Tilford. Tidier drawings of trees. *IEEE Trans. Software Engineering*, SE-7(2):223–228, March 1981.

[35] L. Royer, M. Reimann, B. Andreopoulos, and M. Schroeder. Unraveling protein networks with power graph analysis. *PLoS Computational Biology*, 4(7):e1000108, 2008.

[36] H. Salgado, S. Gama-Castro, M. Peralta-Gil, E. Daz-Peredo, F. Snchez-Solano, A. Santos-Zavaleta, I. Martnez-Flores, V. Jimnez-Jacinto, C. Bonavides-Martnez, J. Segura-Salazar, A. Martnez-Antonio, J. Collado-Vides, and J. Collado-Vides. RegulonDB (version 5.0): K-12 transcriptional regulatory network, operon organization, and growth conditions. *Nucleic Acids Research*, 34(suppl 1):D394–D397, 2006.

[37] P. Saraiya, C. North, and K. Duca. An evaluation of microarray visualization tools for biological insight. In *IEEE Proc. Symposium on Information Visualization*, pages 1–8, 2004.

[38] P. Shannon, A. Markiel, O. Ozier, N. Baliga, J. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[39] P. Simonetto, D. Archambault, D. Auber, and R. Bourqui. ImPrEd: An improved force-directed algorithm that prevents nodes from crossing edges. *Computer Graphics Forum*, 30(3):1071–1080, 2011.

[40] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man and Cybernetics*, 11(2):109–125, 1981.

[41] F. van Ham. Using multilevel call matrices in large software projects. In *IEEE Proc. Symposium on Information Visualization*, pages 227–232, 2003.

[42] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[43] M. Westenberg, S. van Hijum, O. Kuipers, and J. Roerdink. Visualizing genome expression and regulatory network dynamics in genomic and metabolic context. *Computer Graphics Forum*, 27(3):887–894, 2008.