Combining DagMaps and Sugiyama Layout for the Navigation of Hierarchical Data

Guy Melançon, Pierre-Yves Koenig, Charles Bohan, Bérengère Gauthier

Combining DagMaps and Sugiyama Layout for the Navigation of Hierarchical Data

Pierre-Yves Koenig, Guy Melançon
CNRS UMR 5506 LIRMM, INRIA Futurs / CNRS UMR 5506 LIRMM
Montpellier France
{Pierre-Yves.Koenig, Guy.Melancon}@lirmm.fr

Charles Bohan, Bérengère Gautier
Maison de la géographie, Montpellier France
{Charles.Bohan, Berengere.Gautier}@mgm.fr

Abstract

This paper presents a novel technique for exploring and navigating large hierarchies, not restricted to trees but to directed acyclic graphs (DAGs). The technique combines two different visualizations emphasizing different points of view the user can adopt when analyzing the content of the hierarchy. Usual hierarchical (node-link) layout reflect the relative position of nodes in the hierarchy while a variation of a treemap emphasizes node attributes. The classical treemap algorithm has been adapted in order to deal with DAGs. Linking the two views enables the user to visually and interactively explore elements of the hierarchy with respect to selected attributes, while being able to locate the node in the DAG.

1. Introduction

Hierarchical data naturally appears in numerous applications. In their simplest form, hierarchies organize into trees. Tree structures such as classifications (knowledge classification systems, species and/or taxonomies, for instance), phylogenetic trees, company organization charts, are but a few examples. When dealing with richer hierarchical organization, cross links allow elements to belong to several non distinct classes. Inheritance relations of classes in object oriented programming is a typical example. Classifying concepts extracted from documents, for instance, can sometimes require to "duplicate" a concept depending on its possible interpretations. The "network" concept, for example, can refer to social network analysis or to low level computer hardware and both concepts may be required to index a collection of documents. Such a classification naturally leads to the construction of a DAG.

The mathematical structure that naturally models this situation is a directed acyclic graph (a *DAG*). That is, nodes are ordered as *ancestor* nodes or *child* nodes just as with trees, with the exception that nodes may have multiple parents. Nodes with no ancestors are called *source* nodes, while those without any child will be called *sinks*. When drawing the DAG, source nodes are often placed at the top and are said to have level 0. The level of a node is then set to the length of a longest path connecting it to a source node. In doing so, we make sure all edges go downwards (see Figure 1).

The case study we shall explore in this paper is made up of companies linked to one another through subsidiary links. That is, company c_1 links with company c_2 if c_2 is a subsidiary of c_1. Clearly, this graph structure is a DAG: subsidiaries themselves have subsidiaries, and a subsidiary may be held by several "ancestor" companies. The dataset also collects attributes measuring how much of a subsidiary is held by each of its parent companies. The top image in Figure 1 shows a part of a DAG describing links between companies and their subsidiaries. Node (and link) color and size map to different attributes that can be computed from the DAG (see section 2).

Natural questions emerge when studying data modeled with DAGs. The level of a node v in the DAG roughly correspond to how "general" it is. More precisely, it correspond to how much of the DAG it spans, or how many nodes can be reached from v going downwards. For companies, this amounts to how much a company controls its subsidiaries (and subsidiaries of its subsidiaries, etc.). This is even more true when one takes edge or node attributes such as com-

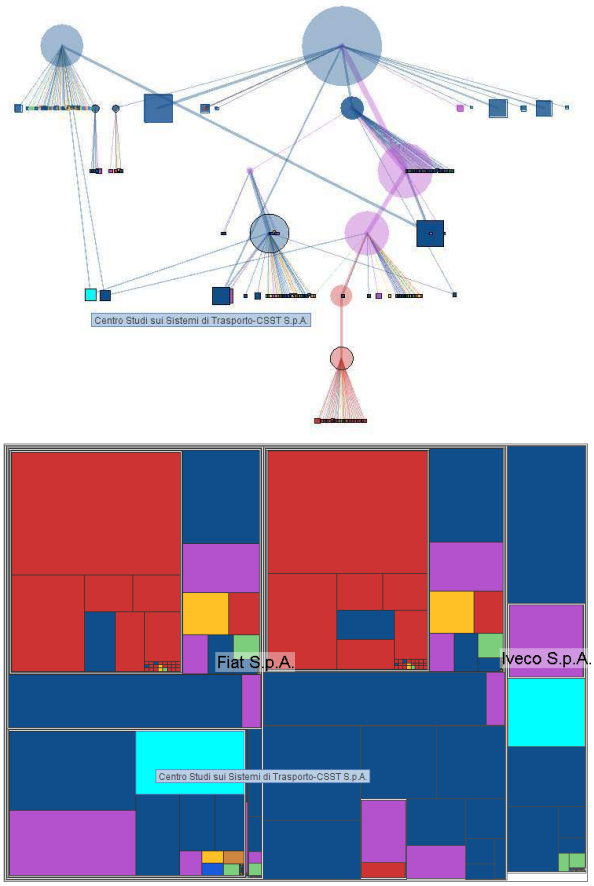


Figure 1. Traditional Sugiyama node-link layout (top) and corresponding DagMap view of a DAG (right). Node size and color correspond to node attributes such as companies' assets and headquarters' location.

panies' assets into account. In our case, the DAG structure also reveals how much two (or more) companies' strategies or interests overlap over a set of subsidiaries. The technique we have set up combines a node-link view of the DAG explicitly showing links between companies and visually showing where a company sits in the whole hierarchy. This type of view however is not optimal for showing and comparing node attributes. We have thus designed a DagMap view, which builds a TreeMap [9] out of a DAG in order to emphasize node attributes such as the total capital of a company, making it easy to compare companies based on this measure. Color coding semantic attributes such as the country (or region) for the company's headquarters revealed to be efficient in helping geographers study how companies build strategies to escape from income tax, compete against competitors or collaborate with others. For instance, our visualization eases the identification of small subsidiaries as being "tax haven" in the DagMap view while the node-link view helps understand how they link to ancestor companies or to other subsidiaries.

The remaining of the paper is organized as follows. We first briefly describe our case study, to motivate the use of the combined DagMap and node-link layout. We then provide details on how the DagMap is computed out of the DAG. Coming back to the case study, we explain how the node-link view and the DagMap view are bound to one another through user interaction. Related work is then discussed before the paper concludes on future work.

2. Case study

Part of this work was conducted in collaboration with geographers through the National French Research Program SPANGEO[1]. Data was obtained through the Orbis database maintained by the Bureau van Dijk office[2]; part of the data was also collected manually by one of the author geographer (C. Bohan). The full dataset concerns about 140 000 companies emerging as subsidiaries of 597 main companies covering numerous industrial sectors. Companies (and subsidiaries) are linked to one another by about 250 000 links. Geographers are not however interested in studying the full dataset but rather concentrate on specific sectors of activity (NACE code). Alternatively, part of the whole DAG emerging from selected major companies (source nodes) can be computed and explored separately. The case study we are concerned with here spans over subsidiaries of a major European car constructor.

The visualization help geographers reveal how companies obey territorial logics or on the contrary develop their activity based on other concerns. Strategies differ from one group to another (Peugeot has a completely different organization than that of Renault for instance), but also differ from one sector to another (food companies indeed usually sell their products locally; the car industry is organized differently, with parts produced in lower cost countries, cars assembled in another, while vehicles can be shipped wherever markets exist). The picture becomes even more complex when companies expand their activities over several industrial sectors. Typically, companies will develop subsidiaries in the financial sector in order to have control over financial flows generated by their overall activity, for instance. As a consequence, commercial and financial strategies intertwin over several industrial sectors.

The combined DagMap and node-link visualization provides an efficient exploratory device in order to develop a view over these phenomena. The right image in Figure 1

[1] See the URL `s4.parisgeo.cnrs.fr/spangeo/spangeo.htm`
[2] See the URL `www.bvdep.com/en/orbis.html`

illustrates the DagMap view of subsidiaries of the car industry emerging from the Fiat international group. Cells have been colored according to whether subsidiaries are installed in Europe (blue – darker blue is used to indicate that a company belongs to one of the 15 founder European countries), Asia (yellow), North America (orange), South America (green) or Africa (gray). This color code was designed with the help of geographers; they moreover required a special color code for subsidiaries suspected to act as tax haven (magenta). The DagMap relies on a squarified treemap algorithm [1] where cell areas correspond to companies' assets.

The DagMap actually encodes a DAG, and not simply a tree. This translates into the possibility of seeing, in the treemap itself, whether a specific subsidiary depends on a single ancestor company, or on the contrary that it is held by more than one competing companies. When clicking on a cell, the user can readily see whether the underlying element expands over other cells in the treemap. This visual feedback provide a quick measure of how much a subsidiary is present in the network and help analyze companies' strategies. This is the case for the subsidiary "Centro Studi sui Sistemi di Trasporto" (CSST for short) which depends on both "Iveco S.p.A." and "Fiat Auto Holdings B.V.". Looking at the node link diagram we indeed see how it sits between these two companies. We moreover observe that "Fiat Auto Holdings B.V." does not hold its CSST subsidiary directly but rather through "Fiat Auto S.p.A.". It is not our role here to make any conclusion about these relationships between these companies and their subsidiaries. We rather wish to underline the usefulness of this combined view, as it has been reported to us during our interviews with our final users.

3. DagMaps: extending TreeMaps to directed acyclic graphs (DAGS)

Treemaps appear as an excellent alternative to traditional (node-link) layout for trees [9], emphasizing attributes of leaf nodes as opposed to relative position of nodes in the tree. Indeed, internal nodes of the tree are apparent only through the nesting of cells and can moreover be equipped with several attributes such as node size or color (compare images in Figure 1). Many improvements on the original design of treemaps have been suggested [12, 1]. Traditional tree layout (see [2, 7]) draw internal nodes as well as leaf nodes and thus explicitly show the relative positions of nodes in the tree. These properties also remain with traditional node-link layout for DAGs [3] .

We shall now explain how the basic treemap algorithm can be adapted in order to deal with DAGs. Roughly speaking, the DAG is unfolded into a tree by duplicating nodes where necessary. That is, a node v with multiple father

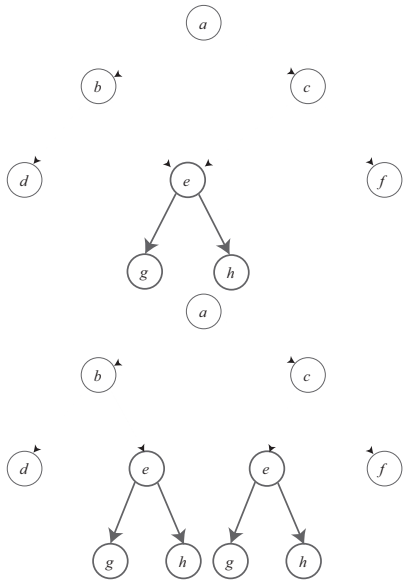


Figure 2. A tree is obtained from a DAG by duplicating nodes (and labels) with multiple ancestors.

nodes is duplicated as many times as necessary in order that each of its father has its own copy of v. Put formally, let $G = (V, E)$ be a DAG. Given a node $v \in V$, denote by $F(v)$ the father nodes of v in G. In other words $F(v) = \{u \in V | (u, v) \in E\}$. Now, assume for a moment the subgraph H induced from the set of descendant nodes of $v \in V$ (nodes reachable from v going downwards, including v itself) is a *tree*. For each father node $u \in F(v)$ we clone the subtree H into a distinct subtree H_u and attach it below u. In doing so, nodes $u \in F(v)$ now have distinct descendants (as far as v is concerned). See Figure 3. Performing this type of transformation (we denote as $\mathcal{T}$) bottom-up from sink nodes towards the sources will eventually output a tree $T = \mathcal{T}(G)$.

Conversely, let $T = (V, E)$ be a tree where nodes have labels, that is T comes equipped with a labelling $\lambda : V \to \mathcal{L}$. Suppose moreover that T is such that if two nodes $u, v \in V$ have equal labels $\lambda(u) = \lambda(v)$, then the subtrees $T_u = (V_u, E_u)$, $T_v = (V_v, E_v)$ extending from u and v are isomorphic (same tree structure) and corresponding nodes have equal labels. That is, there is a bijective correspondence $\phi : V_u \to V_v$ satisfying $\lambda(x) = \lambda(\phi(x))$ for all $x \in V_u$, moreover preserving edges: (x, y) is an edge in E_u if and only if $(\phi(x), \phi(y))$ is an edge in E_v. We can then define on T an equivalence relation on nodes where $u \equiv v \iff \lambda(u) = \lambda(v)$. The quotient set $V/\equiv$ can then be equipped with a graph structure G where classes $[u]$, $[v]$ connect (in G) if two nodes $x \in [u], y \in [v]$ connect in

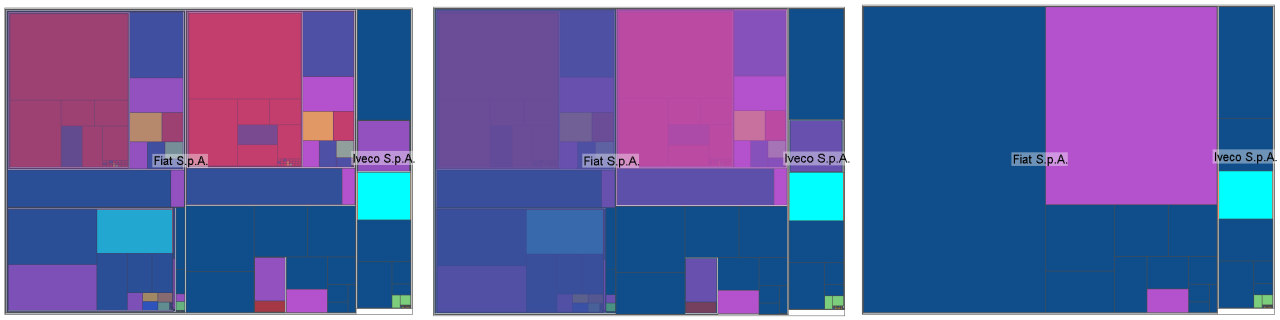


Figure 3. Selecting subsidiaries, the user immediately visualizes "regions" of the hierarchy where it is involved (the CSST subsidiary appears twice as a pale turquoise cell). Using a slider, the user can dim the cells of lower level subsidiaries, emphasizing visual cues of higher level companies' activity.

T. As can be easily seen, the graph computed from T is a DAG, and the previous unfolding process unfolds G back into T (see Figure 3).

This bijective correspondence moreover needs to be implemented in order to track user interaction on the DagMap and map it back to the original DAG (or over the Sugiyama layout when synchronizing views).

The treemap algorithm proceeds recursively, cutting the cell associated with a node into subcells for each of the child nodes, moreover ordering the child nodes $u_1, \ldots, u_k$ according to their number of leaf nodes (in $T_{u_1}, \ldots, T_{u_k}$). This also holds true with the DagMap: the number of sink nodes accessible from node u (in G) equals the number of leaf nodes in the subtree T_u (in $T = \mathcal{T}(G)$). As a consequence, because we may suspect companies with many subsidiaries to have greater asset, cells with greater area are placed in the top left part of the treemap.

4. Exploring the hierarchy: coordinating DagMaps with classical Sugiyama layout

The use of the DagMap alone revealed to be insufficient when exploring the whole hierarchy of companies and subsidiaries, as we explain below. The exploration focuses on the discovery of different commercial strategies and companies, and its relation to the territory. Note that the visual exploration only comes as a first step towards a full explanation of such a phenomenon. Typically, users built their analysis combining standard (text-based) exploration with Google search trying to build hypothesis, then back to a classical node-link visualization with questions in mind, iterating a sense-making loop [10] engaging various intellectual postures (from wild guesses to strong and documented assertions).

The complexity of the analysis however partly comes from the necessity of quickly perceiving the level and place of a node in the hierarchy, while visualizing attributes such as a company's asset and localization (country/continent)[3]. Simply mapping companies' assets and territorial membership using node size (and thus node area) and color in a traditional layout (top image in Figure 1) proved to be inefficient, apparently because it did not easily allow the visual comparison of node area. Not mentioning the fact that the traditional layout displays at least twice as much visual information as the DagMap since ancestor nodes are mapped onto colored disks as well), or edge crossings and occlusions. Note also that a lot of space is left unoccupied with the Sugiyama layout, as is usual with almost all node-link layouts. As a consequence, the space required to display all nodes with areas reporting companies assets (in our case) requires a lot more space than with the treemaps and argues unfavorably for a Sugiyama-alone visualization.

Another difficulty comes from the fact that a company's strategy is embodied over several levels in the hierarchy. For instance, Asian or South American subsidiaries of Fiat only appear at the lowest level (Figure 4), suggesting that delocalization actually take place through intermediate subsidiaries. The identification of this type of scenario required specific visual cues. We introduced an artefact helping the user to momentarily lessen the visual impact of lower level subsidiaries, in order to see higher level strategies form before digging down to lower levels. Indeed, although lower level subsidiaries might be Asian companies, the strategy is revealed by observing whether control is held by European companies, for instance. To this end, the user can dim lower level companies and vary the opacity of the corresponding

[3] As a matter of fact, the original data is not exactly a DAG: there might indeed be links from a subsidiary controlling part of an ancestor company, thus introducing cycles in the network of links. This however only happens exceptionally, and is considered as marginal by our expert users, who actually discarded them.

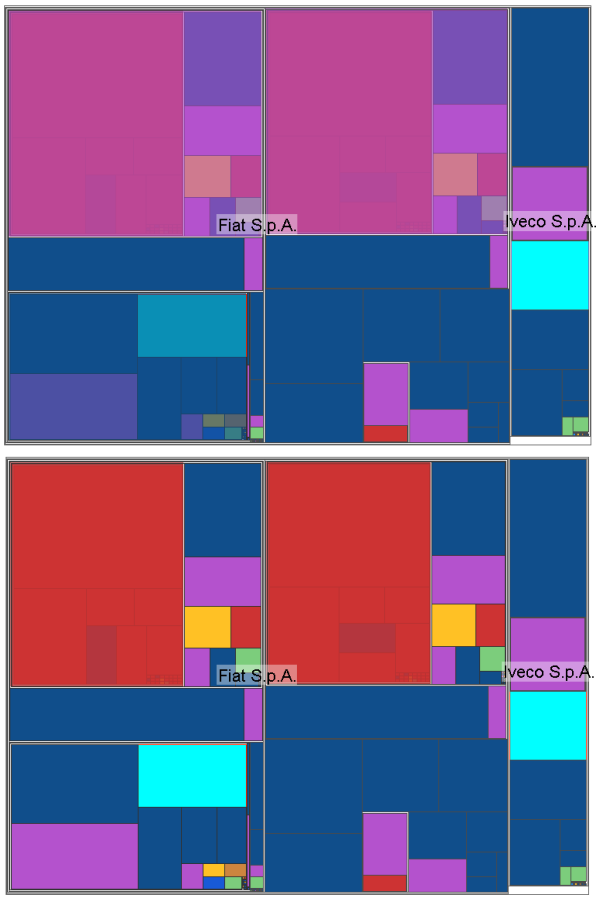


Figure 4. Two different dynamic cues can be used to explore the hierarchy structure. Lower level subsidiaries can be temporarily dimmed out. Moreover, the level at which the dimming effect operates can itself be varied.

cells using a slider (compare images in Figure 4) – this dynamic cue proved to be an efficient way to recover the hierarchy's structure without having to leave the DagMap view. Dimming out all but the highest level companies may readily confirm that control is held by European headquarters. Depending on where subsidiaries are located, the user is then capable of forming hypotheses about strategies being based on territorial logic, as opposed to pure financial strategies, for instance.

5. Related work

Extending treemaps from trees to general graphs has already been studied by Fekete et al. [4], extracting a spanning tree out of a graph, then drawing edges on top of the treemap. Incidentally, Holten's edge bundle technique [6] can be used to improve the readability of such a drawn-over-treemap.

We believe our approach to be more adequate to the visualization of hierarchical data (DAGs) for two reasons. In our case, edges do not connect any two cells but do correspond to inheritance relations which somehow carry specific meaning. Users needed to go back to the Sugiyama layout to consult about inheritance relationships (and hierarchical level of companies), swapping from the treemap to the Sugiyama layout to grasp that information before going back to the treemap. Quickly blinking at the Sugiyama layout appeared to be a fast and efficient way of maintaining their mental map. This quick blink actually makes use of a crucial positional attribute, which translates into edges being all drawn downwards. This would not be the case if edges were simply drawn over a treemap.

Also, extracting a spanning tree would privilege a subtree (or part of the DAG), arbitrarily deciding that a subsidiary belongs more to one owner company as opposed to another owner company. This choice however can only be made based on arguments that precisely should emerge from the exploration and visualization of the datasets. We should keep in mind that the analysis is conducted here in order to confront territories (continent) with financial strategies (control of subsidiaries, implantation of subsidiaries in other parts of the worlds, etc.). The DagMap indeed lays out useful and complementary information on different part of the screen while linking corresponding cells through user interaction. In a sense, the unfolding of the DAG, together with our visualization, provides an exploratory framework opened to any explanatory hypothesis.

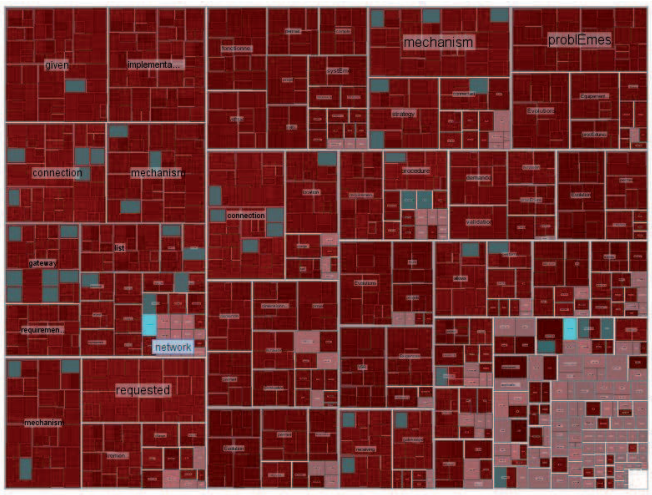
6. Future work

The DagMap is undergoing users' critics as it is used on a daily basis by the geographers. Obviously, other datasets, and other types of datasets can be explored with the help of the DagMap. Typically, object oriented software visualization should benefit from such a visualization since inheritance is a central issue in the design and/or re-design of such systems.

Dags also naturally appear when performing clustering, where clusters are allowed to overlap. Indeed, the hierarchy primarily indicates how clusters nest; common child nodes then roughly correspond to intersection of clusters. Again, DagMaps can be used to observe how much elements spread over the whole clustering.

The image below shows an example where overlapping clusters of concepts (keywords) distribute as higher level cells in a DagMap. The fact that clusters overlap is somewhat natural, as keywords can typically have different meaning. The highlighted cells (grey and turquoise) show how the "network" concept spans across all clusters.

It might be useful in this case to explore clusters at various levels of details, using indices such as Furnas's DOI and API [5] or van Ham van and Wijk's DOA [11], which naturally extend to DAGs.



We definitely should proceed to a formal evaluation of the combined DagMap-Sugiyama technique, testing it against various scenarios (Fekete et al. [4] drawn-over-treemap, Sugiyama alone, DagMap alone, etc.). We should expect, for instance, the DagMap-Sugiyama visualization to work best with DAGs containing but only few cross edges. Experimentations should help determine how much "few cross edges" is. For now, the claimed benefits of our DagMap-Sugiyama technique is solely based on user "interviews". The technique and the tool was actually designed together with users (expert geographers) who were able to informally evaluate our prototype:

Exploring the Fiat dataset with the combined DagMap-Sugiyama visualization, we were able to observe (and confirm) different strategies that were already identified by Porter [8], moreover developing on a continental level with Fiat, which probably obeys a "car culture" differentiation. Also, (Fiat) Europe seems to traditionally control the hierarchy and emergent South American countries sit on lower levels thus confirming a classical center-periphery schema (Porter).

Further analysis is needed to confirm what has been discovered using the DagMap-Sugiyama visualization. Namely, that the lower levels of the hierarchy obey a global model, which could link to the development of the "world car" explaining why subsidiaries from different world regions end up in the same DagMap cell. Conversely, higher levels of the hierarchy concentrate control on European soil, then involving a pyramidal logic in the whole structure: headquarters at the top of the pyramid, relying on financial subsidiaries (2nd level) right before a structuration of markets following cultural (continental) difference (3rd level mainly involving European firms), then an undifferentiated "world car" market at the bottom spread over emergent countries.

Acknowledgements. This work was funded through the French ANR SPANGEO project. We wish to thank our colleague Céline Rozenblat from University of Lausanne for useful comments during all stages of this work, and Laurent Perrier from University of Montpellier who greatly helped to collect the Orbis dataset.

References

[1] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In W. d. Leeuw and R. v. Liere, editors, *Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '00)*, pages 33–43, Amsterdam, 2000. Springer-Verlag.

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

[3] P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 13(4):424–437, 1990.

[4] J. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Overlaying graph links on treemaps. In *IEEE Information Visualization 2003 Symposium Poster Compendium*, pages 82–83. IEEE Press, 2003.

[5] G. W. Furnas. Generalized fisheye views. In *Human Factors in Computing Systems CHI '86*, pages 16–23. ACM Press, 1986.

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

[7] M. Kaufmann and D. Wagner, editors. *Drawing Graphs, Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer.

[8] M. Porter. *L'avantage concurrentiel : comment devancer ses concurrents et maintenir son avance*. Paris : Intereditions, 1986.

[9] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[10] J. J. Thomas and K. A. Cook. Illuminating the path: The research and development agenda for visual analytics. In *Illuminating the Path: The Research and Development Agenda for Visual Analytics*, pages 33–68. IEEE Computer Society, 2006.

[11] F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. In T. Munzner and M. Ward, editors, *IEEE Symposium on Information Visualisation*, Austin, TX, USA, 2004. IEEE Computer Science press.

[12] J. J. van Wijk and H. van de Wetering. Cushion treemaps: visualisation of hierarchical information. In G. Wills and D. Keim, editors, *IEEE Symposium on Information Visualization (InfoVis '99)*, page 7378. IEEE CS Press, 1999.