Layered Graph Force-driven Vertex Positioning

Radek Mařík
Faculty of Electrical Engineering, Czech Technical University, Technicka 2, Prague, Czech Republic

Keywords: Force-driven Layout, Layered Graph, Multitree, Spanning Tree, Phylogenetic Network, Genealogical Network.

Abstract: We propose a new method of node positioning for huge layered graphs specified by layers and fixed ordering of nodes within layers. We assume that the assignments of nodes to the layers and the order of nodes within the layers are provided by other suitable methods capable of processing multitree like networks. The node positioning method is based on the force-driven approach with barrier-like repulsive forces that avoids the quadratic complexity of traditional methods. We demonstrate achievements on several datasets containing up to millions of people or species. The proposed layout method of layered graphs that are close to acyclic multitrees creates aesthetically acceptable layouts in linear time.

1 INTRODUCTION

Although it has been more than 55 years since Tutte introduced barycentric embedding, research of graph visualization techniques remains a highly active field attracting much attention (Tutte, 1963). Working with tree-like structures such as genealogical or phylogenetic networks is no exception in this sense. We will use a genealogical graph as an exemplar of general tree-like networks to simplify our further discussion. Applications to other tree-like network domains are provided in the section dedicated to experiments.

In many situations the resulting network layout is produced by state of the art tools as required (a brief overview of such tools is provided in Section 2). Nevertheless, the tools can often only process networks up to several thousand nodes. The tools and methods that are able to cope with millions of nodes and edges are not able to cover additional constraints, such as the layering of node children into the same layer and keeping the order of nodes within a given layer as expected in genealogical network visualizations.

In this paper, we propose a new node positioning method that keeps both assignments of nodes to layers and the order of nodes within each layer. The method is very fast and therefore can operate with networks having more than a million nodes and edges. The method is based on the force-driven approach with barrier-like repulsive forces that avoids the quadratic complexity of traditional methods.

Thus, in this paper we focus on the third most critical aspect discussed by Sugiyama in (Sugiyama et al., 1981), also related to the third step of the main algorithm proposed in (Gansner et al., 1993), node positioning:

1. determination of generations (layers, node ranks),
2. enforcing node orders within the layers,
3. setting the actual layout coordinates of nodes (node positioning),
4. design of edges.

In other words, we assume that node layers and orders in the layers are fixed in an optimal way (they should not be changed) and it is necessary to accomplish only node positioning. The rest of the steps, such as layering and ordering, is performed by other methods able to operate within huge networks, e.g. such as those proposed for multitree-close networks in (Marik, 2017).

The remainder of the paper is organized in the following way. In the next section, a brief overview of related tools and methods is given. Then in Section 3, we provide a summary of important mathematical concepts used in this paper. We continue with a description of the proposed method's technical details and its steps in Section 4. Finally, we discuss achieved results tested on datasets with up to millions of nodes in Section 5.

2 RELATED METHODS

In the following paragraphs, we provide a brief overview of the methods related to the method proposed in this article. Sections 2.1 and 2.2 review

Mařík, R.
Layered Graph Force-driven Vertex Positioning.
DOI: 10.5220/0006624703010308
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 3: IVAPP*, pages 301-308
ISBN: 978-989-758-289-9

methods related to layering and ordering that serve as preprocessing steps to node positioning. Having this context established, node positioning methods are then described in Section 2.3.

2.1 Multitree Close Networks Layering and Node Ordering

Tree based drawing methods for phylogenetic/genealogical graphs have been one of the standard techniques for centuries. Ancestor trees, descendant trees and Hourglass charts are part of the traditional tools employed by the majority of freeware, shareware, or commercial tools, for example Gramps (Gramps, 2016), MyHeritage (MyHeritage, 2016), TreePlus (Lee et al., 2006), GraphViz (Gansner and North, 2000), and Phylo.io (Robinson et al., 2016). There are also other more space-efficient representations such as fan charts or H-charts (Tuttle et al., 2010; Kieffer et al., 2016). As any pure tree representation enables any ordering of node predecessors/successors, it is possible to specify the type of ordering, such as children ordered by their birth dates. Furthermore, tree representations can be laid out in such a way that family members are grouped together. The tools are not able to process huge networks, which are not pure trees, with millions of nodes while still satisfying constraints on node layering and ordering.

However, the situation with family member grouping changes significantly if the assumptions of one main person and direct ancestors/descendants are dropped. In a number of cases, it is highly beneficial if the entire network of families, or at least a significant part, can be displayed in one layout. Then we often deal with structures close to multitrees and face difficult issues linked with edge crossing and preferences on node clustering (Warfield, 1977; Sugiyama et al., 1981) with occasional cycles usually caused by mistakes in datasets. Therefore, the standard techniques for planar graph layouts (Lempel et al., 1967; Hopcroft and Tarjan, 1974; Shih and Hsu, 1999) including planarization techniques (Resende and Ribeiro, 2001; Chimani et al., 2008) are not suitable in all cases because of a higher than linear complexity.

As we assume multitree-like networks, we would like to stress the significance of layers, so we consider a layout design targeting layered drawing (Healy and Nikolov, 2013). A layering and ordering method targeting huge tree-close networks with up to several millions of nodes that ensures some node layer and order constraints was introduced in (Marik, 2017). A two-level approach when a given quasi-tree (a graph having $O(|V|)$ biconnected components) is decomposed into a tree of biconnected components where biconnected components are drawn with a force-directed approach with subtrees placed onto concentric rings was proposed in (Archambault et al., 2006). Both approaches (Archambault et al., 2006; Marik, 2017) use a spanning tree to drive the high level layout, but they differ in optimization criteria. The majority of algorithms for node layering are derived from the topological order computation, $O(|V|+|E|)$ time complexity (Cormen et al., 2009).

2.2 Constrained Graph Layout

There is also a class of methods that tackle graph layout while satisfying different types of constraints targeting, for example, node separation, drawing cycles, and non-overlapping clusters (Dwyer and Koren, 2005; Dwyer, 2009). Although the methods are applicable to more general networks than those close to multitree treated in this paper, they have a higher complexity (often quadratic) than linear. The experiments presented networks of up to 10,000 nodes only. The given node layers and the order of nodes within layers assumed in this paper can be expressed using about $O(|V|^2)$ node separation constraints that leads to an unfeasible problem for networks with the number of nodes of magnitude $|V| \approx 10^6$.

2.3 Node Positioning

In this paper, we assume that layers of nodes and a node order within the layers are given and fixed. Therefore, we focus only on a new node positioning force-driven method. At first, we refer to the traditional reference by Tutte (Tutte, 1963). His paper focuses mainly on 3-connected planar graphs. Tutte considered barycentric graph embedding. The paper is also referenced as the first "force-directed" algorithm because suitable vertex positions can be found by solving a system of linear equations, so that it exhibits features of force-directed methods (Battista et al., 1999). Sugiyama in (Sugiyama et al., 1981) proposed vertex positions calculated using quadratic programming with an objective function combining distances among vertices from neighboring layers and distances from lower and upper layer barycenters that is minimized. In 1984, Eades (Eades, 1984) introduced the spring-embedder model in which graph vertices are denoted by a set of rings, each pair of rings is connected by a spring and the spring is associated with two types of forces: attraction forces and repulsive forces. Fruchterman and Reingold (Fruchterman and Reingold, 1991) added the even vertex distribu-

tion criterion to the previous two, aiming for the same length of edges and a layout symmetry and also experimented with a number of force forms. They selected a model in which nodes at a distance d are attracted to each other by the attractive force f_a following the quadratic form and they are pushed apart by the repulsive force f_r

$$f_a(d) = d^2/k, \quad f_r(d) = -k/d^2, \tag{1}$$

where k is the optimal distance between vertices defined as

$$k = C\sqrt{\frac{\text{area}}{|V|}} \tag{2}$$

Both algorithms (Eades, 1984) and (Fruchterman and Reingold, 1991) compute attractive forces between adjacent vertices and repulsive forces between all pairs of vertices. Thus, these classical algorithms exhibit a quadratic complexity and do not keep a given node ordering. We will propose a suitable modification of the force-directed method to position ordered nodes in layers in almost linear time.

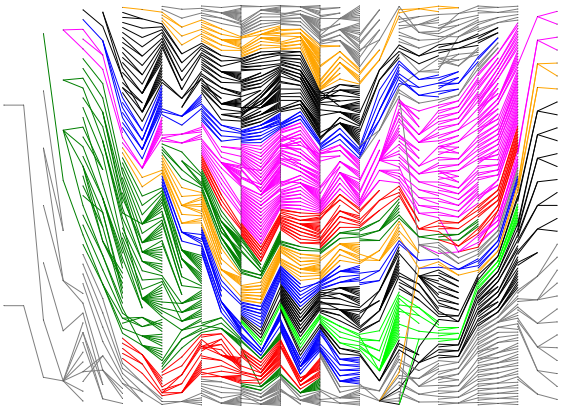


Figure 1: A private family tree (Mykiska's network). A visualization of a sample private family tree with 2192 individuals and 765 marriages created using the uniform spacing method. The layout is created very quickly (below 0.5 second with a Python script on DELL XPS 13 using an Intel i7 2GHz processor). The visualization demonstrates the unpleasant zig-zag layout that might be difficult to read for large networks. The layers are drawn vertically from left to right.

3 TREE-LIKE NETWORKS

To begin, we need to demarcate the structures we treat in this paper. We aim for domain data that cannot be modeled as pure tree structures. We assume directed tree-like networks that can form hierarchies, occasionally exhibit sharing of subtrees, and ones in which strongly connected components very rarely occur. We do not attempt to assess to which level such a structure deviates from a tree although we are aware of some quantification methods (Dorogovtsev and Mendes, 2003; Gu and Sun, 2010). We follow the usual graph theory terminology (Diestel, 2005; Bondy and Murty, 2008).

A **layering** $L = (L_1, L_2, \ldots, L_h)$ of a graph, $G = (V, E)$ is an ordered partition of V into non-empty *layers* L_i such that adjacent vertices are in different layers, i.e. if $\{u, v\} \in E$, where $u \in L_i$ and $v \in L_j$, then $i \neq j$ (Brandes and Köpf, 2002). An **ordering** of a layered graph is a partial order $\prec$ of V such that either $u \prec v$ or $v \prec u$ if and only if $L(u) = L(v)$.

4 NODE POSITIONING

We assume that the partitions of the vertices into layers and node orders in each graph layer are precomputed and fixed, e.g. by a method in (Marik, 2017) following the first two phases proposed by Sugiyama. The order of nodes in layers is the main force driving the resulting graph layout. However, there is still a large number of possibilities in which to assign 2D coordinates to the nodes. In the following paragraphs we discuss and propose two techniques for node position calculations that also satisfy the constraints on node orders. As the order of nodes in layers can be determined in $O(|V|)$ time for multitree-like networks we focus only on techniques that scale in a similar way. All the following methods assume that graph layers are placed uniformly in a horizontal direction. Additionally, the nodes are linked with straight-lined edges.

We start with a simple positioning technique that produces results that might not be easily readable. It is based on uniform spacing. Then, we provide a technique inspired by force-directed methods. We show that there is a combination of special attractive and repulsive forces that enables the readjustment of node positions quickly, even for millions of nodes.

4.1 Uniform Spacing

A very simple technique is based on uniform spacing. Nodes in each graph layer are placed with a uniform distance in a vertical direction. A layout generated using the ordering method proposed in the paper (Marik, 2017) and using uniform node spreading in both directions is shown in Fig 1. Family clans (subtrees of the undirected spanning tree of the processed network) with more than 150 nodes in presented visualizations are emphasized by different colors. An ideal layout would result in edges creating colored horizon-

tal strips with only a minimum number of crossings. Rapid zig-zag changes (waves) in color strips in vertical directions make graph reading more difficult. On the other hand, the computation can be performed in linear time $O(|V|)$ and it provides a kind of top-level overview for very large graphs with millions of nodes.

4.2 Force-directed Node Positioning

Although node ordering based on uniform spacing produces useable results for very large graph data, its zig-zag form makes reading difficult. Another method we have investigated is based on force-directed node positioning.

We assume again that nodes are layered and ordered within layers. Therefore, we need a method that is capable of reducing the wavy format while retaining the order of nodes in layers.

Force-directed node positioning uses two types of forces: attractive and repulsive ones. Our goal is to make waves straight so we define attractive forces between nodes linked by edges and belonging to different layers. Thus, attractive forces push linked nodes to each other which in turn decreases the range of the zig-zag wavy form of strips (subtrees). We use attractive forces following the quadratic form introduced by Fruchterman and Reingold (Fruchterman and Reingold, 1991), see Equation (1). We also need to use repulsive forces to keep vertices apart. We propose a local barrier-like force instead of $f_r(d)$ according to Equation (1) leading to the quadratic complexity. The resulting force for each node must hold the node ordering and the two node separation constraints. Forces are applied iteratively while nodes move significantly.

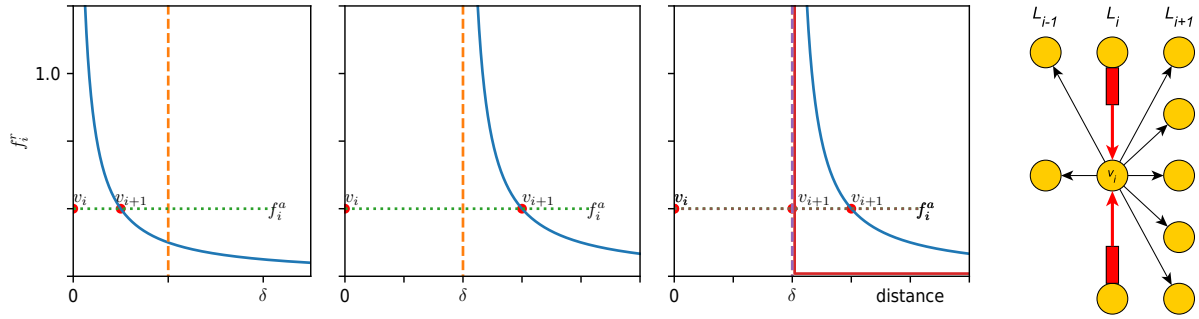
The diagram in Figure 2(d) depicts forces schematically. Both types of forces are projected in the direction along a given layer (a vertical in our case). The optimum distance k is defined as a distance of nodes in a layer with the largest number of nodes $L_{\max}$ if the nodes are placed uniformly, i.e. $k = \delta = C/|L_{\max}|$, C is the predefined height of the layout. If the actual definition of the forces is taken as proposed in (Fruchterman and Reingold, 1991) then the actual distance of nodes might converge to a very small fraction of δ because there are only one or two sources of the repulsion. In addition, a number of neighboring nodes can generate attraction in one direction and the nodes are prescribed to stay in the same order. The variant a) in Figure 2 captures an instance in which the vertices v_i and v_{i+1} in the same layer are pushed so close to each other that the total sum of all projected attractive forces f_i^a shown as a given value and pulling the vertex v_i is in equilibrium with the repulsive force f_i^r generated by the vertex v_{i+1}. In real layouts the resulting distance of both vertices might be a tiny fraction of the minimum separation constraint δ. A modification of repulsive forces compensating such cases can be defined to generate an infinite repulsive force at distance δ by shifting the repulsive force field beyond the minimum separation distance as is shown in variant b). However, in both cases, displacements of nodes in each iteration become very small after a few iterations and then the algorithm converges very slowly. The repulsive force from the neighbors of a given node is necessary to propagate forces through the network in order to reach their equilibrium.

However, there is another way to propagate repulsive forces. We can imagine a repulsive force with a profile in which a repulsive force between two neighbor nodes $v_j^{(i)}, v_k^{(i)} \in L_i$ is 0 if their distance $d_{jk} = |x(v_k) - x(v_j)| \geq \delta$ and ∞ if $d_{jk} < \delta$. In other words, such a profile models a barrier at distance δ, see Figure 2 variant c). If $d_{jk} < \delta$, we say that the barrier between v_j and v_k is *active*. While this variant of repulsive force reflects the shifted modification discussed above, case b), it cannot be easily used for force propagation. Nevertheless, having neighboring nodes in the layer order, we can approximate the repulsive force of a given node by the attractive forces of its neighboring nodes if their barriers are active. In fact, we only consider the two direct neighboring nodes in the rest of the paper.

We will show that if repulsive forces are modeled as barriers then the problem leads to a solution combining barycenters of vertex clusters that can be performed iteratively again in linear time. In fact, repulsive forces are modeled as decision procedures realizing barriers. In each iteration, each node is moved to its equilibrium position or the nearest barrier if such a position is not reachable.

As vertices are strictly tied to a given layer, we can only focus on projections of forces towards the direction of the layer. Thus, we can denote the union of the lower and upper neighbors of the given vertex v as $N_v = N_v^- \cup N_v^+$. Furthermore, we drop the upper index of the k layer in the following steps for vertices $v_i = v_i^{(k)} \in L_k$ with the exception of $v_j \in N_{v_i}$. The coordinate of vertex v_i will be denoted as $x_i = x(v_i)$. Let f_i^a be a total attractive force pulling the vertex v_i and having the quadratic form $f_i^a(x_i) = 1/\delta \sum_{v_j \in N_{v_i}} (x_j - x_i)^2$. The optimum position $\mu_i^I = \arg\min_{x_i} f_i^a$ of the vertex v_i can be found as an minimum of the function f_i^a using $\frac{df_i^a}{dx_i} \stackrel{!}{=} 0$. Thus, the resulting equilibrium position

$$\mu_i^I = 1/|N_{v_i}| \sum_{v_j \in N_{v_i}} x_j \qquad (3)$$



(a) (b) (c) (d)

Figure 2: Repulsive force variants. Three different variants of repulsive force models. Nodes v_i and v_{i+1} of the same layer mark the resulting position of v_{i+1} relatively to the position of v_i. f_i^r is the sum of all repulsive forces. a) The variant proposed by Fruchterman and Reingold breaking the minimum separation δ constraint. b) Its shifted modification satisfying the minimum separation constraint. c) The barrier variant proposed and used in this article. d) Forces on a vertex. Attractive (black) forces from the lower and upper vertex neighbors pull the vertex v_i towards their barycentric position. Repulsive (red) forces from the vertex predecessor and successor in the layer L_i push the vertex v_i away to maintain the minimum separation constraint δ. The vertices might be equipped by a solid separation barrier schematically drawn as red rectangles. Only force projections to the layer L_i need to be treated because of the layer constraint.

is the barycentric position of v_i. The vertex v_i should be placed into the equilibrium position μ_i^I if no barrier to the predecessor v_{i-1} or the successor v_{i+1} is active. If a barrier is active then we need to combine two or three force sources as we have a node and at most two of its neighbor barriers active. Let us assume that the active barrier is between the vertex v_i and its successor v_{i+1}. The position of v_i is x_i and the position of v_{i+1} is $x_{i+1} = x_i + \delta$. The total attractive force f_{i+1}^a pulling the successor v_{i+1} pushes also the vertex v_i as the repulsive force. Thus, if only this one barrier is active, then the total force pulling the vertex v_i is

$$f_i^{Is}(x_i) = f_i^a(x_i) + f_{i+1}^a(x_{i+1}) = f_i^a(x_i) + f_{i+1}^a(x_i + \delta) \tag{4}$$

In this case equilibrium can be reached at the position

$$\mu_i^{Is} = \frac{|N_{v_i}|}{|N_{v_i}| + |N_{v_{i+1}}|}\mu_i^I + \frac{|N_{v_{i+1}}|}{|N_{v_i}| + |N_{v_{i+1}}|}(\mu_{i+1}^I - \delta) \tag{5}$$

In other words, the equilibrium position μ_i^{Is} is a convex combination of no barrier equilibrium positions with the successor displaced by $-\delta$. Similarly, we can obtain the equilibrium position of v_i, only if the barrier with the predecessor is active

$$\mu_i^{pI} = \frac{|N_{v_i}|}{|N_{v_i}| + |N_{v_{i-1}}|}\mu_i^I + \frac{|N_{v_i-1}|}{|N_{v_i}| + |N_{v_{i-1}}|}(\mu_{i-1}^I + \delta) \tag{6}$$

If both the barriers with the predecessor and the successor are active, we start with the total force pulling the vertex v_i

$$f_i^{pIs}(x_i) = f_{i-1}^a(x_{i-1}) + f_i^a(x_i) + f_{i+1}^a(x_{i+1}) = \tag{7}$$

$$= f_{i-1}^a(x_i - \delta) + f_i^a(x_i) + f_{i+1}^a(x_i + \delta) \tag{8}$$

for which the equilibrium position of the vertex v_i is

$$\mu_i^{pIs} = \frac{|N_{v_i-1}|(\mu_{i-1}^I + \delta) + |N_{v_i}|\mu_i^I + |N_{v_i+1}|(\mu_{i+1}^I - \delta)}{|N_{v_{i-1}}| + |N_{v_i}| + |N_{v_{i+1}}|} \tag{9}$$

Furthermore, we need to determine what case of active barriers occurs to calculate the new optimum position x_i' of the vertex v_i. This can be accomplished using a simple decision procedure:

1. if $\mu_{i-1}^I \le \mu_i^I - \delta$ and $\mu_i^I + \delta \le \mu_{i+1}^I$, then $x_i' := \mu_i^I$;
2. if $\mu_{i-1}^I > \mu_i^I - \delta$ and $\mu_i^{pI} + \delta \le \mu_{i+1}^I$, then $x_i' := \mu_i^{pI}$;
3. if $\mu_{i-1}^I \le \mu_i^{Is} - \delta$ and $\mu_i^I + \delta > \mu_{i+1}^I$, then $x_i' := \mu_i^{Is}$;
4. otherwise $x_i' := \mu_i^{pIs}$.

We update the vertices positions layer by layer following the order of vertices in the given layer. We perform two such sweeps in the forward direction and then two sweeps in the backward direction (both layers and orders) to speed up node position changes propagation. The initial positions of vertices are determined using the uniform spacing strategy. As we move only one node, although its new position is the result of all its neighboring nodes, we also need to satisfy the order of vertices. We again use another decision procedure to determine a new feasible position x_i'' for the vertex v_i

1. if $x_i' \le x_{i-1} + \delta$ then $x_i'' := x_{i-1} + \delta$,
2. if $x_i' \ge x_{i+1} - \delta$ then $x_i'' := x_{i+1} - \delta$,
3. otherwise $x_i'' := x_i'$.

The new feasible positions might result in position oscillations. Thus, the new position $x_i^\star$ of the vertex

Table 1: Sample datasets and their basic network statistics: a node number $|V|$ of the complete network , a people number $|V_P|$ of the complete network, a marriage number $|V_M|$ of the complete network, a node number $|V_{max}|$ of the maximum component, an edge number $|E_{max}|$ of the maximum component, a number of strongly connected components $|SCC|$ of a size larger than 1, a number of layers $|L|$, a number of source nodes $|V_{src}|$.

Dataset	$\|V\|$	$\|V_P\|$	$\|V_M\|$	$\|V_{max}\|$	$\|E_{max}\|$	$\|SCC\|$	$\|L\|$	$\|V_{src}\|$
Mykiska's network	2952	2192	765	2913	2917	0	27	609
USA presidents	3186	2145	1042	1589	1602	0	77	480
WeMightBeKin	52783	38486	14297	52672	54210	0	82	12716
ITIS	945352	472676	65799	615342	615341	0	36	1
Stobie's network	996055	706794	289268	995522	1038192	2	187	218593
MGP	200832	188351	0	188351	213865	0	52	7678

v_i is determined conservatively as a convex combination of the current and feasible positions as $x_i^\star = 0.6x_i'' + 0.4x_i$. The constants 0.6 and 0.4 were determined experimentally, but the approach is not sensitive to their modifications. The constant values mean that the correction of the vertex position is reflected partially by displacement of the current vertex and partially by its neighbors. We stress again that the strict and solid barriers with the attractive forces assumed to have the quadratic form enable the combining of barycenters of vertices only. The related sums of coordinate values and their counts are sufficient to calculate new positions.

5 IMPLEMENTATION, EXPERIMENTS, AND DISCUSSION

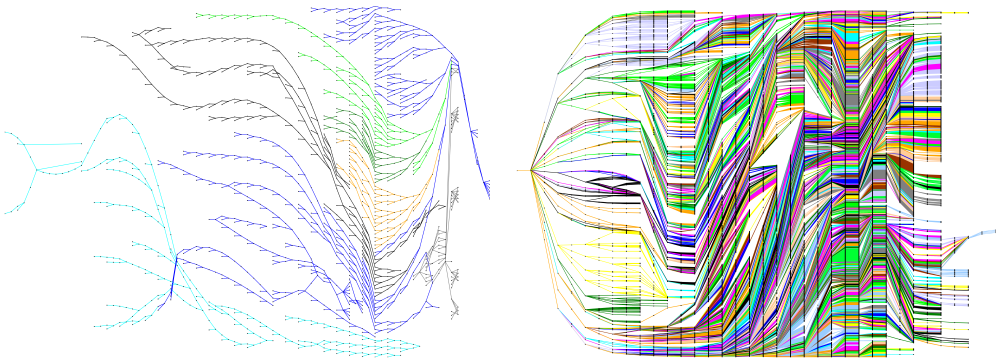
The algorithm was implemented as an experimental non-optimized Python script with a number of additional procedures evaluating the layout process and used on an ultrabook DELL XPS 13 with 16GB of RAM using an Intel i7 2GHz processor. The layering and ordering is very fast, taking from mere seconds up to a dozen minutes for networks with a million vertices. The positioning procedure uses 10×4 iterations lasting a similar amount of time.

Although tree-like networks might be observed in a number of domains, such as telecommunications, forensic and cause-effect networks, in this paper we performed experiments mainly using phylogenetic and genealogical datasets exhibiting multitree properties. We selected 20 datasets to evaluate the proposed methods (Pruitt, 2017; Leskovec and Krevl, 2017; GoogleFFT, 2017). Example datasets and their statistics are shown in Table 1. Except for one, all datasets we used in our experiments had no cycles. Stobie's dataset (Stobie, 2017) contains 2 small strongly connected components. Some datasets represent genealogical networks, the ITIS (Integrated Taxonomic Information System) dataset is a snapshot of the Catalog of Life in the GEDCOM format (ITIS, 2017). The ITIS dataset exhibits a special feature: the total number of 'person' and 'marriage' records covers only 57% of the entire graph nodes.

Another dataset consists of 3057 people from a database created by Egyptologists (Dulíková, 2016). Experiments with families from the database did not exhibit any layout deficiencies as the families are quite simple and no larger than 50 family members. Similar results were obtained for the rest of the datasets which are GEDCOM files downloaded from the Internet containing between 400 and 1,000,000 individual records, such as Stobie's network, the Mathematics Genealogy Project (MGP) (MGP, 2017), and the ITIS dataset (ITIS, 2017).

For example, the diagram in Figure 3(a) depicts the result of an almost hierarchical graph structure capturing a genealogical network of US presidents (Pruitt, 2017). The performance of the method can be demonstrated on the phylogenetic network layout in Figure 3(b) with 615342 nodes as an overview of taxonomic information on plants, animals, fungi, and microbes as developed by the Integrated Taxonomic Information System (ITIS) (ITIS, 2017). One can easily observe the overall structure of such a multitree.

Example layouts shown in this section dedicated to experiments demonstrate that the wavy form of the resulting layout is only partially removed. As has been observed with a number of different force-directed layout method variants, the convergence of a vertex's position after dozens of iterations is very slow. The slow convergence is attributable to several reasons. First, if the position of lower neighbors is opposite to the position of upper neighbors (layers), then the current vertex is not moved because it stays in the optimum position. Thus, the vertices of wavy subgraphs are dragged into new positions only by unbalanced forces for vertices located near the wave extrema. Furthermore, there are stable configurations of vertices even when their local layout is bent. A so-



(a) (b)

Figure 3: a) Genealogy of US presidents. A resulting layout of a network with an almost hierarchical structure. While the rapid zig-zag subtree layouts disappeared, the large range of wavy formats for subtrees is still present. Thus, the node positions are still not optimal as the locally force-driven algorithm is caught in a local extreme. b) An ITIS snapshot. A snapshot of a phylogenetic network with almost one million nodes. One can easily identify that the entire network consists of one large submultitree on the right side and a forest of smaller ones on the left side of the diagram.

lution might be found in a special treatment of entire subtrees that we are currently investigating.

6 CONCLUSIONS

In this work we proposed a new method for layered graph node positioning, i.e. layered tree-like network layouts with constraints on node order with regard to their partition into layers and their order in layers. We restricted and demonstrated its application to tree-like networks as we are aware of methods that are capable of layering such large graphs with almost linear complexity. The produced graph layouts are more acceptable for the user if they deal with large networks combining many trees into a single acyclic graph. We are not aware of any other tool that is able to layout multitree-like networks with millions of nodes and edges while maintaining the given layering and ordering of nodes. The layout can be computed using an unoptimized Python script during a processing time in which other tools would utilize for drawing networks of two order magnitude smaller. Thus, the experiments clearly demonstrate a significant improvement in graph comprehension. They also indicate that results provided by existing state of the art tools are quite far from computing an optimum layout with regard to the number of edge crossings, constraints on node layers and orders, at least for special types of graphs such as genealogical ones.

The proposed method enables graphical highlights that assist with any type of initial analysis and data processing treating datasets with a large number of entities. Technical details on how a suitable software tool providing additional utilities like zooming, panning, searching, filtering can be easily implemented are beyond the scope of this article. It is also obvious that the wavy character of the layout can still be improved. We are investigating possibilities that would assign node positions based on forces taking into account entire subtrees of the undirected spanning trees.

ACKNOWLEDGEMENTS

Sponsored by the project for GAČR, No. 16-072105: Complex network methods applied to ancient Egyptian data in the Old Kingdom (2700–2180 BC).

Datasets on officials from the Old Kingdom of Egypt were kindly provided by Veronika Dulikova and Miroslav Barta at the Czech Institute of Egyptology, the Faculty of Arts, Charles University in Prague. The author thanks the Mathematics Genealogy Project for providing data from its database for use in this research.

REFERENCES

Archambault, D., Munzner, T., and Auber, D. (2006). Smashing peacocks further: Drawing quasi-trees from biconnected components. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):813–820.

Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G. (1999). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Englewood Cliffs, NJ.

Bondy, J. and Murty, U. (2008). *Graph Theory*. Springer.

Brandes, U. and Köpf, B. (2002). *Graph Drawing: 9th International Symposium, GD 2001 Vienna, Austria, September 23–26, 2001 Revised Papers*, chapter Fast

and Simple Horizontal Coordinate Assignment, pages 31–44. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chimani, M., Junger, M., and Schulz, M. (2008). Crossing minimization meets simultaneous drawing. In *2008 IEEE Pacific Visualization Symposium*, pages 33–40.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Diestel, R. (2005). *Graph Theory*. Springer.

Dorogovtsev, S. N. and Mendes, J. F. F. (2003). *Evolution of Networks, From Biological Nets to the Internet and WWW*. Oxford University Press.

Dulíková, V. (2016). *The Reign of King Nyuserre and Its Impact on the Development of the Egyptian state. A Multiplier Effect Period during the Old Kingdom.* PhD thesis, Charles University in Prague, Faculty of Arts, Czech Institute of Egyptology.

Dwyer, T. (2009). Scalable, versatile and simple constrained graph layout. *Computer Graphics Forum*, 28(3):991–998.

Dwyer, T. and Koren, Y. (2005). Dig-cola: directed graph layout through constrained energy minimization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 65–72.

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160.

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

Gansner, E. R., Koutsofios, E., North, S. C., and Vo, K. P. (1993). A technique for drawing directed graphs. *IEEE Transactions nn Software Engineering*, 19(3):214–230.

Gansner, E. R. and North, S. C. (2000). An open graph visualization system and its applications to software engineering. *Softw. Pract. Exper.*, 30(11):1203–1233.

GoogleFFT (2017). Google: Famous family trees. `https://groups.google.com/forum/\#\!forum/famous-family-trees`.

Gramps (2016). Gramps. genealogical research software. `https://gramps-project.org/`. Accessed: 5.6.2016.

Gu, Y. and Sun, J. (2010). A tree-like complex network model. *Physica A: Statistical Mechanics and its Applications*, 389(1):171 – 178.

Healy, P. and Nikolov, N. S. (2013). *Handbook of Graph Drawing and Visualization*, chapter Hierarchical Drawing Algorithms, pages 409–453. CRC Press.

Hopcroft, J. and Tarjan, R. (1974). Efficient planarity testing. *Journal of the ACM*, 21(4):549–568.

ITIS (2017). ITIS - Integrated Taxonomic Information System. `https://www.itis.gov/downloads/index.html`. Retrieved February, 10, 2017, from the Integrated Taxonomic Information System on-line database, `http://www.itis.gov`.

Kieffer, S., Dwyer, T., Marriott, K., and Wybrow, M. (2016). Hola: Human-like orthogonal network layout. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):349–358.

Lee, B., Parr, C. S., Plaisant, C., Bederson, B. B., Veksler, V. D., Gray, W. D., and Kotfila, C. (2006). Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426.

Lempel, A., Even, S., and Cederbaum, I. (1967). An algorithm for planarity testing of graphs. In Rosenstiehl, P., Gordon, and Breach, editors, *Theory of Graphs*, pages 215–232, New York.

Leskovec, J. and Krevl, A. (2017). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Marik, R. (2017). *Complex Networks & Their Applications V: Proceedings of the 5th International Workshop on Complex Networks and their Applications (COMPLEX NETWORKS 2016)*, chapter Efficient Genealogical Graph Layout, pages 567–578. Springer International Publishing, Cham.

MGP (2017). Mathematics genealogy project, department of mathematics, north dakota state university. `https://www.genealogy.math.ndsu.nodak.edu/index.php`. Accessed: February 2017.

MyHeritage (2016). Myheritage. `https://www.myheritage.cz`. Accessed: 5.6.2016.

Pruitt, P. D. (2017). Great sites for links to genealogy software. `http://famousfamilytrees.blogspot.cz/2011/12/`. Accessed: February 2017.

Resende, M. G. C. and Ribeiro, C. C. (2001). *Encyclopedia of Optimization*, chapter Graph planarization, pages 908–913. Springer US, Boston, MA.

Robinson, O., Dylus, D., and Dessimoz, C. (2016). Phylo.io: Interactive viewing and comparison of large phylogenetic trees on the web. *Molecular Biology and Evolution*.

Shih, W.-K. and Hsu, W.-L. (1999). A new planarity test. *Theoretical Computer Science*, 223(1-2):179–191.

Stobie, T. (2017). Thomas stobie's genealogy pages. `http://freepages.genealogy.rootsweb.ancestry.com/~stobie/`. Accessed: February 2017.

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

Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society, Third Series*, 3(13):743–768.

Tuttle, C., Nonato, L. G., and Silva, C. (2010). Pedvis: A structured, space-efficient technique for pedigree visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1063–1072.

Warfield, J. N. (1977). Crossing theory and hierarchy mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(7):505–523.