Journal of Visual Languages and Computing 44 (2018) 72–88

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

View Points

Neighborhood beautification: Graph layout through message passing

Severino F. Galán[a,*], Ole J. Mengshoel[b]

[a] Artificial Intelligence Dept., UNED, C/ Juan del Rosal, 16, Madrid 28040, Spain
[b] Carnegie Mellon University, NASA Research Park, Moffett Field, CA 94035, USA

ARTICLE INFO

Article history:
Received 18 March 2016
Revised 15 June 2016
Accepted 29 November 2017
Available online 1 December 2017

Keywords:
Graph drawing
Aesthetic graph layout
Neighborhood interaction
Message passing

ABSTRACT

Graph layout algorithms are used to compute aesthetic and useful visualizations of graphs. In general, for graphs with up to a few hundred nodes, force-directed layout algorithms produce good layouts. Unfortunately, for larger graphs, they often get stuck at local minima and have high computational complexity. In this paper, we introduce a novel message passing technique for graph layout. The key idea of our message passing technique is that an aesthetic layout can be obtained if each node independently suggests aesthetic placements of its neighbors. In other words, every node sends messages to its neighbors, indicating new and better positions for them. As a result, the new technique, which we call *Neighborhood Beautification*, provides a new perspective that turns out to give a useful trade-off between the excellent layout quality reached by force-directed methods and the fast runtime achieved by algebraic methods. Neighborhood Beautification reduces, in many cases, the computational cost of force-directed algorithms, since only interactions between neighboring nodes are considered. Experimentally, we show that Neighborhood Beautification produces high-quality layouts for grid-like graphs but is outperformed by force-directed algorithms in the case of more complex graphs.

1. Introduction

Graphs are widely used in artificial intelligence, computer engineering, computer science, mathematics, and statistics to model objects and connections between them. They have been successfully applied in areas like automata theory, automatic planning, databases, electrical power networks, machine learning, network security, probabilistic graphical models, social networks, software engineering, and VLSI technology [26]. In order to easily understand and manipulate the data represented by graphs, aesthetic and useful visualizations, including layouts, of them are needed. Many graph layout methods have been developed [12,35,49]. Objectives that have been considered in developing such methods include [12,47]: minimization of edge crossings, uniformity of edge lengths, even distribution of nodes, and maximization of symmetries.

The present work deals with the problem of graph layout in the sense of nicely drawing undirected graphs whose edges are straight lines. This problem reduces to appropriately positioning nodes in a two dimensional plane. A popular approach to this problem is the force-directed technique, where the graph is associated with a system of interacting physical objects. This approach defines the energy of the system such that, when a minimum is achieved, the layout tends to be nice and useful. Given initial positions for the interacting physical objects, the system evolves by reducing its energy. Variants of this approach differ in how the energy is defined [11,14,20,37]. Force-directed graph layout can be combined with other methods for initializing the position of the nodes, for example, treemap-based methods [44].

Although force-directed methods are intuitive, easy to implement, and produce acceptable results for graphs of medium size (up to a few hundred nodes), they also have important disadvantages:

- The number of iterations needed to produce a good layout can be quite high. For example, since graphs are usually initialized with random positions for their nodes, in the case of large graphs the convergence to a global minimum is often too slow or even hard to achieve due to the presence of many local but non-global minima. Therefore, layout of large graphs requires the use of techniques in addition to force-directed methods. The multi-level technique [30,31,43,55] (see Section 2.2), which works regardless of the initial positions of the nodes and only requires the graph structure, is a convenient option for large graph drawing. Using this technique, a sequence of graphs $(G^0, G^1, \ldots, G^{l-1}, G^l)$ is first generated, so that G^0 is the original graph and G^{k+1} is obtained by grouping nodes of G^k. The coars-

* Corresponding author.
E-mail addresses: seve@dia.uned.es (S.F. Galán), ole.mengshoel@sv.cmu.edu (O.J. Mengshoel).

https://doi.org/10.1016/j.jvlc.2017.11.008
1045-926X/

ening continues until a graph G^l with only a small number of nodes is reached. Since the optimized layout for the coarsest graph can be more easily obtained, then two additional consecutive phases (placement and refinement) are performed iteratively. In the placement phase, the optimized layout of a coarser graph G^{k+1} is transformed to a finer layout G^k. In the refinement phase (or single-level layout), typically a force-directed method is used to improve the layout resulting from the placement phase.

- The computational complexity of each iteration of a force-directed method is usually $\mathcal{O}(M+N^2)$, where M and N are the number of edges and nodes in the graph respectively. This complexity derives from the fact that every node changes its position at each iteration, and this change depends on the attractive and repulsive forces exerted by the rest of the nodes. Whereas the attractive (mechanical) forces take place between neighboring nodes and are calculated in $\mathcal{O}(M)$ time for the whole graph, the repulsive (electrical) forces occur between any pair of nodes and are determined in $\mathcal{O}(N^2)$ time. Since the quadratic complexity of repulsive forces makes force-directed methods impractical for large graphs, these forces can be approximated by grouping sufficiently distant nodes, which reduces the complexity to $\mathcal{O}(N \cdot \log N)$ [4,28,34,48,52]. Alternatively, Fruchterman and Reingold [20] use a grid variant of their force-directed algorithm that ignores the repulsive forces between distant nodes. Although this variant calculates the repulsive forces in $\mathcal{O}(N)$ time for sparse graphs with uniform node distribution over the grid squares, for general graphs this calculation may become more expensive.

This work presents a novel message passing technique for graph layout named *Neighborhood Beautification*, which provides a new perspective on graph layout similar to how message passing has provided new perspectives on areas such as decoding [42] and compressive sensing [13]. The underlying idea of this novel technique is that a globally aesthetic layout of a graph can be achieved if every set formed by a node and its neighbors is independently laid out in an aesthetic way. An iteration of the neighborhood beautification method consists of three message passing phases that address the following aesthetic criteria in turn: minimize edge crossings, maximize edge length uniformity, and maximize angular resolution. In these three iterated phases, messages are passed from every node in the graph to each of its neighbors. A message contains information about the desired position for the receiving node from the viewpoint of the sending node under one of the three aesthetic criteria. Neighborhood Beautification can be applied as a single-level graph layout method, similar to force-directed algorithms, or as a graph layout method for the refinement phase of the multi-level technique.

The usefulness of a graph layout method is usually established by taking into account both its computational cost (in terms of time and memory) and the quality of the resulting layouts. The Neighborhood Beautification method takes $\mathcal{O}(M+N \cdot \Delta \cdot \log \Delta)$ time for each iteration in the worst case[1] (see Section 3.4), since only interactions between neighboring nodes are considered; therefore, the high complexity associated with the calculation of repulsive forces in force-directed algorithms is reduced in many cases. In the experiments of this paper, we find that although the Neighborhood Beautification method in general needs more iterations than force-directed algorithms, it usually offers shorter computation time. As far as the layout quality is concerned, Neighborhood Beautification provides promising results for grid-like graphs compared to force-directed algorithms, although for complex graphs the quality of layouts produced by force-directed algorithms is superior in general. In this work, we extensively evaluate the performance of the Neighborhood Beautification method in terms of running time and layout quality.

Since Neighborhood Beautification relies on message passing between neighbors, a distributed implementation [2,25] could be constructed that is enabled by the graph structure. Yet, in this work we do not implement a distributed version of Neighborhood Beautification, but leave it to future work. Further, Neighborhood Beautification is implemented in this work as a sequential algorithm. As mentioned in Section 5, an interesting future research direction is its implementation in the context of parallel computing [33,45,50].

The rest of this paper is organized as follows. Section 2 reviews several widely used methods for single-level and multi-level layout of graphs. Section 3 explains our Neighborhood Beautification method in detail. Section 4 describes a series of experiments for evaluating the Neighborhood Beautification method. Finally, Section 5 summarizes the main results of our work and enumerates future research directions.

2. Related work

We consider graphs whose nodes have unconstrained positions and whose edges are straight lines. Existing algorithms for aesthetically laying out such graphs can broadly be classified as single-level or multi-level. Single-level methods exclusively operate on the original graph to be laid out, whereas multi-level methods use additional auxiliary graphs obtained from transforming the original graph. Single-level graph layout can be carried out in several ways: force-directed [11,14,20,37], incrementally [10,51], or algebraically [8,32,39,40]. Although the algebraic methods are significantly faster than the force-directed methods, they frequently produce graph layouts of inferior quality compared to graph layouts obtained using force-directed methods [27, Section 1.3.3].

Due to their importance and widespread use, this section reviews two force-directed approaches: (1) the spring-embedder model, proposed by Eades [14] and later modified by Fruchterman and Reingold [20], and (2) Kamada and Kawai's model [37]. We also review the multi-level technique for graph layout [30,31,55], an approach successfully used for large graphs, and where a force-directed method is used as a subroutine.

2.1. Force-directed graph layout

To tackle the graph layout problem, force-directed algorithms are inspired by a physical analogy. These algorithms consider forces acting on the nodes and iteratively transform the graph layout from its initial configuration (usually with random positions for the nodes) until a configuration is reached where the net force acting on each node is null. Equivalently, they associate a potential energy with a layout and aim to find a layout whose energy is locally optimal. Kobourov has written an excellent and detailed survey of different force-directed algorithms for graph layout [38].

2.1.1. The spring embedder model

The spring embedder model was defined by Eades [14] as a system of electrically charged rings connected by springs. The mechanical forces provoke the attraction between every pair of neighboring nodes, whereas the electrical forces make every pair of non-neighboring nodes repel each other.

The attractive force exerted by a spring is modeled by the following formula:

$$F_{\text{att}} = c_1 \cdot \log\left(\frac{d}{c_2}\right), \tag{1}$$

[1] The symbol Δ denotes the maximum degree of the graph.