Demonstration of a Preprocessor for the Spring Embedder

Paul Mutton, Peter Rodgers

University of Kent at Canterbury
{pjm2@ukc.ac.uk, P.J.Rodgers@ukc.ac.uk}

Abstract. Spring embedding is a widely used method for producing automated layouts of graphs. We present a preprocessor that improves the performance of the classical spring embedder, which can be used in conjunction with other optimization and approximation techniques. It creates an initial graph layout with edge lengths that are approximately equal and with a minimum node separation from which the spring embedder typically needs far fewer iterations to produce a well laid out graph.

Introduction. The spring embedder [1] is a widely used method for drawing general graphs. In its classical form, the layout is randomized before the iterative force process is simulated. We present a preprocessor for the spring embedder, discussed in more detail in [5]. Our preprocessor is a two phase process and is used before the spring embedder is initiated. The first phase attempts to keep connected nodes at a suitably close distance by an iterative method that produces edges lengths close to a user specified ideal. The second phase approximates an even node distribution by placing the nodes on a grid. The spring embedder, or some other optimized force directed variant, is applied after these two phases. In some specialized cases, such as grid or mesh like structures, or some graphs exhibiting strong clusters, the graph can be drawn with just the preprocessor and without requiring any subsequent spring embedding.

Previous work in preprocessing includes placing the graph in simple binary trees or meshes [2]. The Wave Front method [6] applies the spring embedder to a subgraph wave that sweeps through the graph. Other work on initial placement includes [4] which adds nodes to the layout incrementally, placing nodes at the barycenter of its three closest graph theoretical distance neighbours. This is followed by a local application of the force directed drawing process on the node, considering just its neigbours.

Brief Description of the Method. We use force models based on those of Fruchterman and Reingold, which defines k, the ideal minimum node separation. In this model, the repulsive force between nodes is $-k^2/d$ and the attractive force due to edges is d^2/k, where d is the distance between the two nodes.

The first phase of the preprocessor attempts to obtain an initial drawing where all edge lengths are approximately equal to ka, where $a \geq 1$, as the spring embedder appears to perform better when reducing the volume of the graph slightly. Preliminary experimentation suggests that $a = g^{1/3}$ is a reasonable value, where g is the mean degree of nodes in the graph. Prior to phase one, the nodes are randomly distributed within a cubic volume of side length $10^3 ka(|N|^{1/3})$, which is large enough to ensure good results, as the layout will rapidly shrink with the extropic application of the iterative first phase.

Each iteration of phase one involves visiting every node in the graph. For each node n, we examine the set of emanating edges. Each edge connects n to m_i and we calculate the position of point p_i such that the vector $m_i p_i = ka(u_i)$, where u_i is the unit vector of $m_i n$. Node n is moved to the mean position of all p_i of n. The graph rapidly shrinks into a more stable graph where each edge length is near to ka. Each iteration has linear time complexity, O(|E|).

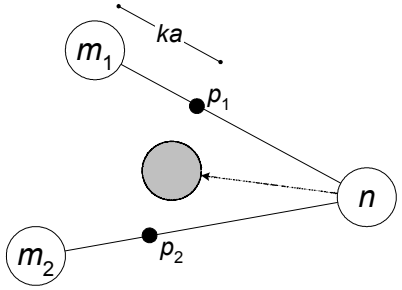

Figure 1: Node placement in the first phase

Phase one often leaves dense clusters of nodes very close together, so we now apply the second phase in order to achieve a minimum node separation. This places the nodes on a three-dimensional grid of unit size k. This is much quicker than the first phase, as it is not iterative. Application of both phases results in a new graph layout where edge lengths remain close to the phase one ideal value of ka, whilst ensuring all nodes have the final graph ideal minimum node separation of k. This yields the initial drawing to which we apply the spring embedder.

Summary of Results and Further Work. Preliminary results show the method is promising in improving the speed of the spring embedding process. The spring embedder finds an equilibrium for medium sized graphs of about 200 nodes approximately 10 times faster than with a random layout [5]. The time taken for the preprocessing is insignificant compared to the time required for spring embedding. Further experiments with optimized spring embedders also show improvements in performance. In some cases spring embedding was not required, as the preprocessor produced a good layout. Systematic testing with larger graphs and alongside other preprocessing methods is now needed to explore the benefits of the method.

The method has been implemented in Java with a focus on encapsulation and reuse; however, a radical performance increase could be gained by concentrating on optimising the code in an effective environment. Also requiring consideration is integration with other optimization and approximation techniques, as the method is amenable to integration with common force approximation methods such as Barnes-Hut tree coding methods, and with multilevel clustering methods.

References

1. P. Eades. A Heuristic for Graph Drawing. Congressus Numerantium 42. pp. 149-60. 1984.
2. A. Frick, A. Ludwig, H. Mehldau. A Fast Adaptive Layout Algorithm for Undirected Graphs. GD'94, LNCS 894. pp. 388-403. 1995.
3. T.M.J. Fruchterman, E.M. Reingold. Graph Drawing by Force-directed Placement. Software – Practice and Experience, Vol. 21(11). pp. 1129-1164. 1991.
4. P. Gajer, M.T. Goodrich, S.G. Kobourov. A Fast Multi-Dimensional Algorithm for Drawing Large Graphs. GD 2000, LNCS 1984. pp. 211-221. 2001
5. P.J. Mutton, P.J. Rodgers. Spring Embedder Preprocessing for WWW Visualization. Proceedings of 6th International Conference on Information Visualization, IV02-WGV. pp. 744-749. 2002.
6. A.J. Quigley, Large Scale Relational Information Visualization, Clustering, and Abstraction, PhD Thesis, University of Newcastle, Australia. August 2001.