Eurographics Conference on Visualization (EuroVis) (2014)
N. Elmqvist, M. Hlawitschka, and J. Kennedy (Editors)

Node Overlap Removal by Growing a Tree

Arlind Nocaj[1], Lev Nachmanson[2], and Sergey Bereg[3]
[1]University of Konstanz
[2]Microsoft Research, Redmond
[3]University of Texas at Dallas

Abstract

Node overlap removal is often a necessary step in many scenarios including laying out a graph, or visualizing a tag cloud. Our contribution is a new overlap removal algorithm that iteratively builds a Minimum Spanning Tree on a Delaunay triangulation of the node centers and removes the node overlaps by "growing" the tree. The algorithm is simple yet it produces high quality layouts and usually runs several times faster than the widely used PRISM algorithm.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

A configuration with overlapped nodes often appears after applying a graph layout algorithm [Tam07]. To remedy this an overlap removal algorithm is usually applied. The algorithm PRISM [GH10] is widely used for this purpose. Our contribution is a simple algorithm that we call Growing Tree, or GTree further on, running faster than PRISM and producing outputs of comparable quality. To make the comparison with PRISM easier we implemented GTree in the open source graph visualization software Graphviz, where PRISM is the default overlap removal algorithm.

There is vast research on node overlap removal. Some layout methods have been extended to take the node sizes into account [FS04, FR91, LYC09, LEN05, WM95, DKM06]. Another possibility is to solve the problem by a post-processing step, for which many different approaches exist [LMR98, GN98, IAG*08, SSS*12, HLSG07, GNCNT13, GNSRP*13]. The idea of another set of algorithms is to define pairwise node constraints and translate the nodes to satisfy the constraints [MELS95, HIMF02, MSTH03, HL03]. These methods consider horizontal and vertical problems separately, which may lead to a distorted aspect ratio [GH10]. In [DMS06] the overlap removal is reduced to a quadratic problem and is solved efficiently in $\mathcal{O}(n \log n)$ steps. According to [GH10], the quality and the speed of the method of [DMS06] is very similar to the ones of PRISM.

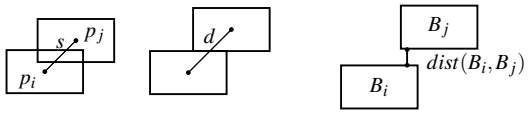
In PRISM [GH10, Hu09], a Delaunay triangulation on the node centers is used as the starting point of an iterative step. Then a stress model for node overlap removal is built on the edges of the triangulation and the stress function of the model is minimized using an iterative linear system solver. The high level structure of GTree is similar to PRISM. We also start with a Delaunay triangulation, but we use it in a different manner.

2. GTree Algorithm

Let us give some definitions. An input to GTree is a set of nodes V, where each node $i \in V$ is represented by a rectangle B_i with the center p_i. We assume that for different $i, j \in V$ the centers p_i, p_j are different too. If this is not the case, we randomly shift the nodes by tiny offsets. We denote by D a Delaunay triangulation of the set $\{p_i : i \in V\}$, and let E be the set of edges of D.

On a high level, a step of our method proceeds as follows. First we calculate the triangulation D, then we define a cost function on E and build a minimum cost spanning tree on D for this cost function. Finally, we let the tree "grow". The steps are repeated until there are no more overlaps. The last several steps are slightly modified. Now we explain the algorithm in more detail.

We define the cost function c on E in such a way that the larger the overlap on an edge becomes, the smaller the cost of this edge comes to be. Let $(i, j) \in E$. If the rectangles B_i and B_j do not overlap then $c(i, j) = dist(B_i, B_j)$, that is the distance between B_i and B_j. Otherwise, for a real number t

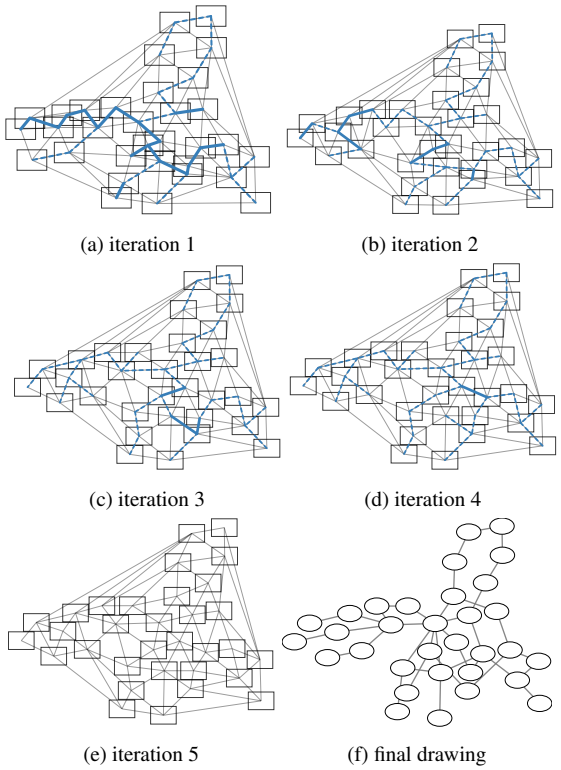


Overlapping nodes
$d = t_{ij}s,\ c_{ij} = -(t_{ij}-1)s$

Non overlapping nodes
$c_{ij} = dist(B_i,B_j)$

Figure 1: Cost funct. for edges of the Delaunay triangulation.

let us denote by $B_j(t)$ a rectangle with the same dimensions as B_j and with the same orientation, but having the center translated to the point $p_i + t(p_j - p_i)$. Now we find $t_{ij} > 1$ such that the rectangles B_i and $B_j(t_{ij})$ touch each other. Let $s = \|p_j - p_i\|$, where $\|\|$ denotes the Euclidean norm. We set $c(i,j) = -(t_{ij}-1)\cdot s$. See Figure 1 for an illustration. We set $t_{ij} = 1$ when B_i and B_j do not overlap. Having the cost function ready we compute a minimum spanning tree T on D for which $\sum_{e\in E'} c(e)$ is minimal, where E' is the set of edges of T. The cost is negative on the edges of E with overlap and is not negative on the rest of the edges. Therefore, the edges connecting overlapping nodes are most probably included into T.



(a) iteration 1 (b) iteration 2

(c) iteration 3 (d) iteration 4

(e) iteration 5 (f) final drawing

Figure 2: Growing a Tree: Bold blue tree edges capture most overlaps. Dashed tree edges restricted to be local. Overlap is completely resolved after few iterations by expanding the bold tree edges and shifting the dashed tree edges accordingly.

We can now resolve the overlaps by growing the tree, similar to the growth of a tree in the nature. Starting from the root node of T we grow the edges adjacent to the root, then continue to its children recursively. Let p' be the vector of new positions and $p'_r = p_r$ for the root r. Then growing an edge $(i,j) \in E$ at node i yields $p'_j = p'_i + t_{ij}(p_j - p_i)$ for j.

Different roots produce the same results modulo a translation of the plane by a vector. We iterate the high level step, starting from finding a Delaunay triangulation, then building a minimum spanning tree on it, and growing the tree (see Fig 2), while an overlap along an edge of the triangulation is found.

When there are no overlaps on edges of the triangulation, as noticed in [GH10], overlaps are still possible. We follow the same idea as PRISM and modify the iteration step. In addition to calculating the Delaunay triangulation we run a sweep-line algorithm to find all overlapping pairs and augment the Delaunay graph D with each such pair. As a consequence, the resulting minimum spanning tree contains non Delaunay edges catching the overlaps, and the rest of the overlaps get resolved. This stage usually requires much less time than the previous one.

Comparing GTree and PRISM: We run comparisons by using edge length dissimilarity, σ_{edge} [GH10], Procrustean similarity transformation, σ_{disp} [BG05], and calculating the area of the graph bounding box. GTree always needs more area than PRISM, but the two other quality measures are close. In Fig. 3, we compare the total CPU time using the same set of graphs as in PRISM [GH10,GN00] on a PC with Linux and an Intel Core i7-2600K CPU@3.40GHz. Overall GTree seems to be a good alternative to PRISM, due to its simplicity.

Acknowledgments This research was partially supported by DFG via grant GRK/1042.

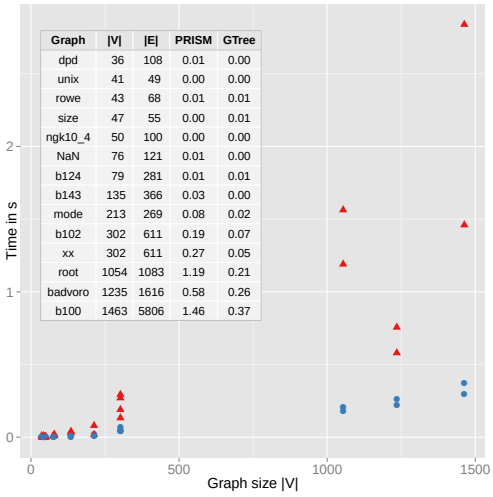


Figure 3: GTree is faster than PRISM for larger graphs.

References

[BG05] BORG I., GROENEN P.: *Modern multidimensional scaling: Theory and applications*. Springer, 2005. 2

[DKM06] DWYER T., KOREN Y., MARRIOTT K.: Ipsep-cola: An incremental procedure for separation constraint layout of graphs. *IEEE Trans. Vis. Comput. Graph. 12*, 5 (2006), 821–828. 1

[DMS06] DWYER T., MARRIOTT K., STUCKEY P. J.: Fast node overlap removal. In *Graph Drawing* (2006), Springer, pp. 153–164. 1

[FR91] FRUCHTERMAN T. M. J., REINGOLD E. M.: Graph drawing by force-directed placement. *Software - Practice and Experience 21*, 11 (1991), 1129–1164. 1

[FS04] FRIEDRICH C., SCHREIBER F.: Flexible layering in hierarchical drawings with nodes of arbitrary size. In *ACSC* (2004), Estivill-Castro V., (Ed.), vol. 26 of *CRPIT*, Australian Computer Society, pp. 369–376. 1

[GH10] GANSNER E. R., HU Y.: Efficient, proximity-preserving node overlap removal. *J. Graph Algorithms Appl. 14*, 1 (2010), 53–74. 1, 2

[GN98] GANSNER E. R., NORTH S. C.: Improved force-directed layouts. In *Graph Drawing* (1998), Whitesides S., (Ed.), vol. 1547 of *Lecture Notes in Computer Science*, Springer, pp. 364–373. 1

[GN00] GANSNER E. R., NORTH S. C.: An open graph visualization system and its applications to software engineering. *Softw., Pract. Exper. 30*, 11 (2000), 1203–1233. 2

[GNCNT13] GOMEZ-NIETO E., CASACA W., NONATO L. G., TAUBIN G.: Mixed integer optimization for layout arrangement. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI-Conference on* (2013), IEEE, pp. 115–122. 1

[GNSRP*13] GOMEZ-NIETO E., SAN ROMAN F., PAGLIOSA P., CASACA W., HELOU E., FERREIRA DE OLIVEIRA M., NONATO L.: Similarity preserving snippet-based visualization of web search results. 1

[HIMF02] HAYASHI K., INOUE M., MASUZAWA T., FUJIWARA H.: A layout adjustment problem for disjoint rectangles preserving orthogonal order. *Systems and Computers in Japan 33*, 2 (2002), 31–42. 1

[HL03] HUANG X., LAI W.: Force-transfer: A new approach to removing overlapping nodes in graph layout. In *ACSC* (2003), Oudshoorn M. J., (Ed.), vol. 16 of *CRPIT*, Australian Computer Society, pp. 349–358. 1

[HLSG07] HUANG X., LAI W., SAJEEV A., GAO J.: A new algorithm for removing node overlapping in graph visualization. *Information Sciences 177*, 14 (2007), 2821–2844. 1

[Hu09] HU Y.: Visualizing graphs with node and edge labels. *CoRR abs/0911.0626* (2009). 1

[IAG*08] IMAMICHI T., ARAHORI Y., GIM J., HONG S.-H., NAGAMOCHI H.: Removing node overlaps using multi-sphere scheme. In *Graph Drawing* (2008), Tollis I. G., Patrignani M., (Eds.), vol. 5417 of *Lecture Notes in Computer Science*, Springer, pp. 296–301. 1

[LEN05] LI W., EADES P., NIKOLOV N. S.: Using spring algorithms to remove node overlapping. In *APVIS* (2005), Hong S.-H., (Ed.), vol. 45 of *CRPIT*, Australian Computer Society, pp. 131–140. 1

[LMR98] LYONS K. A., MEIJER H., RAPPAPORT D.: Algorithms for cluster busting in anchored graph drawing. *J. Graph Algorithms Appl. 2*, 1 (1998). 1

[LYC09] LIN C.-C., YEN H.-C., CHUANG J.-H.: Drawing graphs with nonuniform nodes using potential fields. *J. Vis. Lang. Comput. 20*, 6 (2009), 385–402. 1

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout adjustment and the mental map. *J. Vis. Lang. Comput. 6*, 2 (1995), 183–210. 1

[MSTH03] MARRIOTT K., STUCKEY P. J., TAM V., HE W.: Removing node overlapping in graph layout using constrained optimization. *Constraints 8*, 2 (2003), 143–171. 1

[SSS*12] STROBELT H., SPICKER M., STOFFEL A., KEIM D. A., DEUSSEN O.: Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives. *Comput. Graph. Forum 31*, 3 (2012), 1135–1144. 1

[Tam07] TAMASSIA R.: *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2007. 1

[WM95] WANG X., MIYAMOTO I.: Generating customized layouts. In *Graph Drawing* (1995), Brandenburg F.-J., (Ed.), vol. 1027 of *Lecture Notes in Computer Science*, Springer, pp. 504–515. 1