Clustered edge routing

Citation for published version (APA):

Document status and date:
Published: 16/03/2015

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
1 Introduction

Graphs are an important tool to express relational data. The classic method to depict graphs is a node-link diagram where vertices (nodes) are associated with each object and edges (links) connect related objects. Node-link diagrams represent graphs in the most direct way. However, they quickly appear cluttered and unclear, even for moderately sized graphs. If the positions of the nodes are fixed – because they represent geo-referenced data or are laid out according to functional requirements – then suitable link routing is the only option to reduce clutter.

We present a novel link clustering and routing algorithm which respects (and if desired refines) user-defined clusters on the links. Our input is a node-link diagram with fixed node positions and optionally a user-defined clustering on the links and/or a set of disjoint polygonal obstacles. Our clustering method is based on a well-separated pair decomposition (WSPD) and we route link clusters individually on a sparse visibility spanner. To completely avoid ambiguity we draw each individual link and guarantee that clustered links follow the same path in the routing graph. (We call the edges of the routing graph ‘edges’ and use ‘links’ for the input from the node-link diagram.) Our algorithm also ensures that clusters are not drawn close to nodes and do not cross obstacles (see Fig. 1).

Holten and van Wijk [11] formalized four edge compatibility measures which indicate how similar links are. In Section 2 we argue that the clusters induced by a WSPD consist of compatible links according to the measures of Holten and van Wijk. Users can hence simply vary the separation constant of the WSPD to find a balance between few clusters of less compatible links and many clusters of very compatible links.

In Section 3 we show how to route the link clusters defined by the WSPD along the greedy sparsification of the visibility graph. On the complete graph a greedy sparsification has several desirable properties, such as a provable angle constraint as well as low total weight. Under realistic input assumptions we can prove that these properties still hold for the sparsification of the visibility graph. In Section 4 we describe our complete routing algorithm, including preprocessing, link ordering, and crossing minimization.

Related work. Our work is closely related to the approach by Pupyrev et al. [2, 16] who use a routing graph based on a Yao-spanner of the visibility graph. The edges of a Yao-spanner can get arbitrarily close and angles can be arbitrarily small. Pupyrev et al. hence go through an extensive and computationally expensive iterative optimization step to improve it. Dwyer and Nachmanson [9] also use visibility graphs, albeit approximate ones, to route edges. They describe two approaches. The first uses a spatial decomposition and requires node movement. It can hence not be used to draw graphs with fixed node positions. The second approach also uses the Yao-spanner. All these techniques route links by using shortest paths on the routing graph. Link clustering is hence simply induced by the routing graph and does not necessarily satisfy any similarity measures. Furthermore, neither of these methods supports user-defined link clusters.

Dwyer et al. [8] integrate link routing techniques into a force-directed layout. Their method requires a feasible initial routing and moves vertices.

Various techniques reduce link clutter by bundling links which are “close”. Gansner et al. [10] use a circular graph layout and route links either on the inside or the outside of the circle. Holten and van Wijk [11] describe a force-directed approach and use the aforementioned compatibility measures to determine the strengths of forces. Cui et al. [6] propose a geometry-based approach which uses a control mesh. Lambert et al. [13] use a combination of the Voronoi diagram and a quadtree as a multi-resolution grid for routing links. Bundling methods generally draw bundled links on top of each other. Hence it can be difficult to decide unambiguously if two nodes are connected. Luo et al. [14] propose a method which is ambiguity-free, but bundled links need to share a common node.
2 Clustering links via a WSPD

The well-separated pair decomposition (WSPD) was introduced by Callahan et al. [5]. Two point sets \( A \) and \( B \) are well-separated if they can be enclosed in two circles of equal diameter which are “far apart” relative to their diameter. More precisely, point sets \( A \) and \( B \), with bounding boxes \( R(A) \) and \( R(B) \), are said to be \( s \)-well-separated for some separation constant \( s > 0 \) if \( R(A) \) and \( R(B) \) can be enclosed in two disjoint equal diameter circles \( C_A \) and \( C_B \) and the distance between \( C_A \) and \( C_B \) is at least \( s \) times the diameter of \( C_A \). The WSPD of a set of points \( P \) with separation constant \( s \) is a sequence of \( m \) pairs \( \{A_i,B_i\} \) of nonempty subsets of \( P \) such that

1. for each \( 1 \leq i \leq m \), \( A_i \) and \( B_i \) are well-separated with respect to \( s \).
2. for any two distinct points \( p \) and \( q \) there is exactly one pair \( (A_i,B_i) \) such that \( p \) is in one set and \( q \) is in the other.

The number of well-separated pairs \( m \) is also called the size of the WSPD. If the separation constant \( s \) is indeed constant we can compute a WSPD of size \( O(n) \) on a point set in the plane in \( O(n \log n) \) time, see [15] Chapter 9. Every well-separated pair \( (A_i,B_i) \) induces a link cluster: each link with one endpoint in \( A_i \) and the other endpoint in \( B_i \) is part of the cluster.

**Link compatibility measures.** Holten and van Wijk introduced four measures which concern the angle, scale, and position of a pair of links, as well as the visibility between them. Consider the \( s \)-well-separated pair \( \{A,B\} \). We examine the compatibility measures on any two links \( e = (p_0,p_1) \) and \( f = (q_0,q_1) \) with \( p_0,q_0 \in A \) and \( p_1,q_1 \in B \). We assume that the link \( e \) is at most as long as the link \( f \). We use \( D \) to denote the diameter of the circles \( C_A \) and \( C_B \).

**Angle compatibility.** Links in the same cluster should have a similar angle. We define the angle \( \alpha \) between two non-parallel links as the smallest angle between the lines induced by the links. The angle of parallel links is 0.

**Lemma 1** The angle \( \alpha \) between \( e \) and \( f \) is bounded by \( \alpha \leq 2 \cdot \tan^{-1} (\frac{1}{s}) \) for \( s \geq 1 \).

**Proof.** The figure shows a worst case configuration of \( e \) and \( f \) with respect to \( \alpha \). We have \(|(p_0,t)| \leq 0.5D \) and \(|(t,r)| \geq 0.5sD \) as rough bounds. We can now bound \( \alpha \) by \( 2 \cdot \tan^{-1} (\frac{1}{s}) \). □

**Scale compatibility.** Links in the same cluster should have similar length.

**Lemma 2** The difference in length of \( e \) and \( f \) is at most \( 2 \cdot D \). The length ratio of \( f \) to \( e \) is bounded by \(|f| \leq \frac{|e|}{s} \). \( |q_1| \leq |q_0| \)

**Position compatibility.** Links which are close to each other should be more likely to end up in the same cluster. Holten and van Wijk measure “close to each other” by considering the distance between the midpoints \( p_m \) and \( q_m \) of links \( e \) and \( f \) in relation to the average link length of \( e \) and \( f \).

**Lemma 3** The difference in position of links \( e \) and \( f \) with midpoints \( p_m \) and \( q_m \) is \( |(p_m,q_m)| \leq D \). The ratio of the difference in position to the average length is bounded by \( \frac{|p_m,q_m|}{|p_0,q_1|} \leq \frac{1}{s} \).

**Visibility compatibility.** Let \( q_m \) be the point on the line induced by \( e \) that when projected onto \( f \) coincides with its midpoint \( q_0 \). The visibility compatibility of \( e \) with \( f \) is defined by the normalized distance between the midpoint of \( e \) \( (p_m) \) and \( q_m \). To normalize this distance we divide by the length of the segment \( q_0q_1 \) which when projected onto the line induced by \( f \) coincides with \( f \).

**Lemma 4** Let \( q_m \) and \( q_m \) be the points on the line induced by \( e \) which, when projected onto \( f \), coincide with \( q_0 \), \( q_1 \), and \( q_m \). The visibility compatibility is bounded by \( \frac{|p_m,q_m|}{|p_0,q_1|} \leq \frac{1}{s} \) for \( s > 1 \).

**Proof.** Let \( \alpha \) be the angle between the lines induced by \( e \) and \( f \) which using Lemma 1 we can bound as \( \alpha < 2 \tan^{-1} (\frac{1}{s}) \). Let \( \frac{p_0}{p_m} \) be the projection of \( p_m \) onto the line induced by \( f \). We have \(|\langle p_m,q_m \rangle| \leq |(p_m,q_m)| \leq D \) by the triangle inequality. This implies \(|(p_0,q_m)| \leq D \). From the definition of \( s \)-well-separated we have \(||q_0,q_1|| \geq sD \), which implies that \(|q_0,q_1|| \geq sD \). We now have \(|q_0,q_1|| \leq sD \). □

Increasing the separation constant \( s \) of the WSPD improves all four compatibility measures. Users can hence vary \( s \) to find a balance between few clusters of less compatible links and many clusters of very compatible links. If the user has specified clusters we test if they are also spatially clustered. If this is not the case, that is, if the endpoints of the clustered links are not well-separated, we can refine the user-specified cluster into compatible clusters using the WSPD.
3 The routing graph

We add a small polygonal obstacle around each node, merge those obstacles which are “too close”, and enclose merged obstacles within their convex hull. We use a combination of Voronoi and additional in-cell edges to connect nodes with their obstacle vertices.

Let \( d_G(p,q) \) denote the shortest-path distance between two vertices \( p \) and \( q \) in a graph \( G = (V,E) \). A geometric \( t \)-spanner \((t > 1)\) of \( G \) is a graph \( G' = (V,E' \subseteq E) \) such that for any two vertices \( p,q \in V \), \( d_G(p,q) \leq t \cdot d_G(p,q) \). The so-called greedy spanner is constructed by considering all edges \( e = (p,q) \in E \) in non-decreasing order and adding them to \( E' \) if and only if \( d_G(p,q) > t \cdot d_G(p,q) \). For our routing graph we use the greedy spanner to sparsify the visibility graph. We also include all original obstacle edges.

The visibility graph can be computed in \( O(n^2 \log n) \) time using a sweepline approach (see [7] Chapter 15). The greedy spanner can be computed in \( O(n^2 \log n) \) time [3]. We use the \( O(n^2 \log^2 n) \) algorithm of Bouts et al. [4] which is faster in practice.

The greedy sparsification of the complete graph is very sparse, it guarantees a lower bound for the angles between adjacent edges in realistic input assumptions) a similar proof holds for our routing graph.

Let \((v,u),(v,w)\) be two edges in the greedy spanner with angle \( \alpha \) at \( v \) and let \((v,u)\) be considered before \((v,w)\) by the algorithm \((|vu| \leq |vw|)\). For \( \alpha < {\pi \over 4} \) we have that \( |uw| < |vw| \) and hence there is a \( t \)-path between \( u \) and \( w \) when considering \((u,v)\). Furthermore, we have \( t > 1/(\cos \alpha - \sin \alpha) \). Hence, if \( \alpha \) is too small in relation to \( t \), we have \( |vu| + t|uw| < t|vw| \) contradicting that \((v,w)\) is in the greedy spanner and implying a lower bound on \( \alpha \).

We now assume square axis-aligned obstacles with sidelength \( k \), which are separated by at least \( {1 \over 2}tk \). Hence the sparsification of the visibility graph must include all obstacle edges since no other vertices are sufficiently close to form a \( t \)-path between their vertices. Let \( \delta(u,w) \) denote the length of the shortest path between \( u \) and \( w \) in the visibility graph. Because of our input assumptions \( \delta(u,w) \) is at most a small constant factor larger than \( |uw| \) (see figure). Hence the same argument as above applies, with slightly worse constants.

By adjusting the dilation \( t \), the user can find a balance between a more direction preserving or a cleaner, more abstract routing graph.

4 Routing and ordering links

The routing should respect the clustering. Using the nearest pair of nodes (one from each set of the corresponding well-separated pair) we determine merge points. All links in a cluster are routed via their merge points to ensure that they share a common sub-path.

To avoid unambiguity we draw links individually along the routing graph edges. Inspired by [16] we use bi-arcs to allow them to smoothly change directions at vertices and to ensure that they cross at large angles. Clustered links are drawn together to form a ribbon of links. This leads to a two-level ordering problem. We order both ribbons and the links within them to minimize crossings. We can order the links optimally in linear time since they do not pass through nodes [16].

Because of the tree-like structure of the ribbons ordering them among each other is NP-complete. We prove this by reduction from the 1-sided crossing minimization problem [12]: Given a two-layered (bipartite) graph \( G = ([L_0,L_1],E) \) and an ordering \( x_0 \) of vertices on layer \( L_0 \), is there an ordering \( x_1 \) of \( L_1 \) such that there are at most \( k \) crossings?

Definition 1 (Ribbon ordering problem) Given the ribbons routed through vertex \( v \). Is there an ordering along its edges with at most \( k \) crossings?

Lemma 5 Ribbon ordering is NP-Complete.

Proof. Consider an arbitrary instance of the one-sided crossing minimization problem \( G = ([L_0,L_1],E) \) where \( L_0 \) is the fixed layer. We denote the fixed vertices with \( \bar{v}_1 \ldots \bar{v}_{|L_0|} \) and those in \( L_1 \) by \( v_1 \ldots v_{|L_1|} \). We construct an instance of the ribbon ordering problem as illustrated in Fig. 2. We add edges \( e_1 \ldots e_{|L_0|} \) on the left side corresponding to \( L_0 \) and one edge on the right side. We add ribbons \( r_1 \ldots r_{|L_1|} \) to this edge corresponding to \( L_1 \). For each edge in \( (\bar{v}_i,v_j) \in E \) we add a link from ribbon \( r_i \) to \( e_j \).

We can now solve the original instance by finding an order of ribbons on the right edge. Since ribbon ordering is clearly in NP this proves NP-completeness. □

This close relation to 1-sided crossing minimization allows many of its heuristics to be adapted to ribbon ordering. For our implementation we adapted the

\[
\begin{align*}
\text{Figure 2: A 1-sided crossing minimization problem and the corresponding ribbon ordering instance. The four left edges act as a fixed layer.}
\end{align*}
\]
Barycenter heuristic [12] which works by calculating the “average ranking” for each ribbon.

We demonstrate our method on the TLR4 dataset which represents interactions between different biomolecules. Barsky et al. [1] used their Cerebral software to find node positions and used a simple spline based heuristic to draw links (see Fig. 3(a)). Fig. 3(b) shows the result of our algorithm on their node layout. Clusters were defined by experts and refined where needed using a separation of 1. The sparsification was computed for a dilation of $t = 1.8$.

Acknowledgements. The authors would like to thank Tim Dwyer for fruitful discussions and Tamara Munzner for the Cerebral use case.

References


