Partial Edge Drawings

Birgit Schreiber
Matr.: 1227566
Curriculum: 566
e1227566@student.tuwien.ac.at

December 23, 2017

Contents

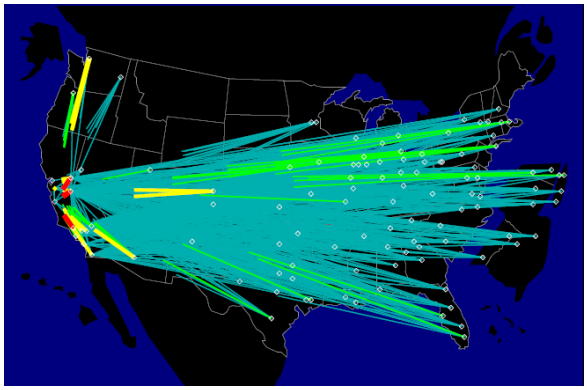
Abstract

For graph visualization one important problem is to avoid visual clutter to make easy understandable drawings for humans. An approach to avoid clutter in form of edge crossings in straight line drawings is to draw only two stubs of an edge and drop the middle part. This is called partial edge drawing and recently several papers investigated this approach. This paper aims to give a basic understanding of partial edge drawings and a small overview of known bounds and algorithm for this kind of drawings. In addition the proof for one bound is presented in more detail. Also two algorithms for creating partial edge drawings from the paper "Progress on Partial Edge Drawings" [8] are explained in order to convey a deeper understanding of partial edge drawings.

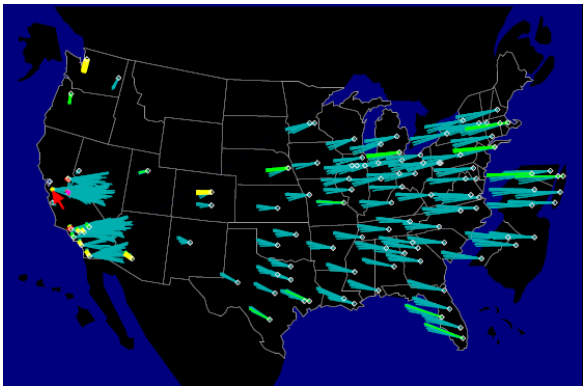
1 Introduction

For numerous applications, like network data - and information visualization, it is crucial to present a graph in an understandable way. When visualizing non-planar graphs visual clutter is one of the main problems. Edge crossings for example seriously affect the readability of non-planar drawings of graphs. There are several approaches to reduce visual clutter and produce aesthetically pleasant and good readable drawings. In this paper Partial Edge Drawings are discussed whose goal is to completely remove edge crossings in straight line drawings. Edge intersections are removed by dividing every edge into three parts and dropping the middle part. The drawing only relies on the so called subs of an edge.

This idea was first introduced by Becker et al. in 1995 [6]. In their paper they searched for a way to visualize the overload the AT&T long distance telephone network in the U.S. experienced when an earthquake occurred. They used line thickness and colour to convey information about the overload of links. If overload occurred only in one direction of a link they only drew the half line. This resulted in the graphic shown in figure 1a. It can be seen that this display is ineffective because most of the country is obscured by the long lines. By only using short stubs of each link they were able to deliver the same information clearer. In figure 1b the same data as before is shown but as partial edge drawing. This is a motivation example how partial edge drawings can benefit graph related data representation.



(a) network-wide overload after the earthquake [6]



(b) network-wide overload after the earthquake as PED, same data as in figure 1a is represented [6]

Figure 1: Motivating Example

Rusu et al. used this concept in 2011 and investigated partially drawn links, inspired by the principles of Gestalt. The Gestalt principles of 'closure' and 'good continuation' play a crucial role for PED and uses the fact that human brains intuitively complete gaps in drawings. Now gaps in the links have to be filled by the human eye to achieve reliable results and answers [3].

Since then various paper investigated partial edge drawings which can be found in the section related work, section 3.

2 Definitions

2.1 Partial Edge Drawings (=PED)

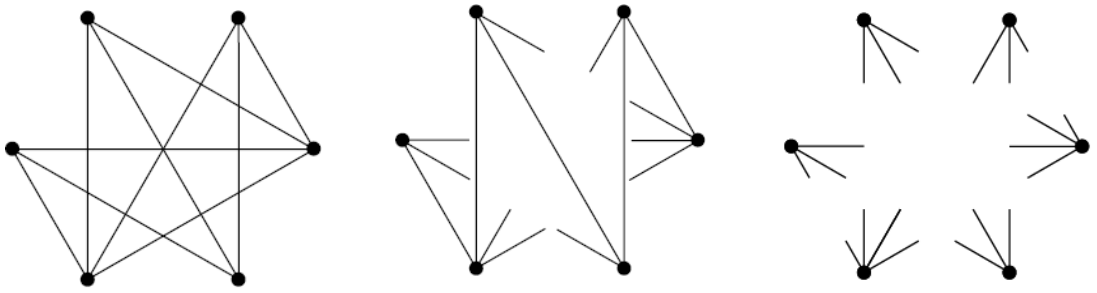
Partial edge drawings are straight line drawings which do not allow any edge crossings. Lets recall straight line drawings map vertices of a graph to points on a two dimensional plane and edges are represented by line segments which connect the two corresponding points. In a PED every edge is only drawn partially. An edge is split in three parts and only the two parts incident to the end points, the so called stubs, of the segment are drawn. Different types of PED can be seen in figure 2.

2.2 Symmetric PED (=SPED)

In a symmetric partial edge drawing the two stubs of an edge are required to have equal length. An example is shown in figure 2b. The symmetry can be used to determine if a link exists between two vertices, by checking for a stub of the same size on the other vertex.

2.3 Homogeneous SPED (=δ-SHPED)

A partial edge drawing is called homogeneous if the stub length is a given fraction δ of the total edge length. The fraction has to be the same for all edges in the drawing. A 1/4-SHPED is displayed in figure 2c. In a δ-SHPED the length of the edge can be concluded from the stub size. The 1/4-SHPED arose as standard. The amount of drawn ink equals the amount of creased ink in a 1/4-SHPED. Many proofs and concepts for δ-SHPED start from $\delta = 1/4$ and are then generalized.



(a) with crossings (b) as SPED (c) as SHPED

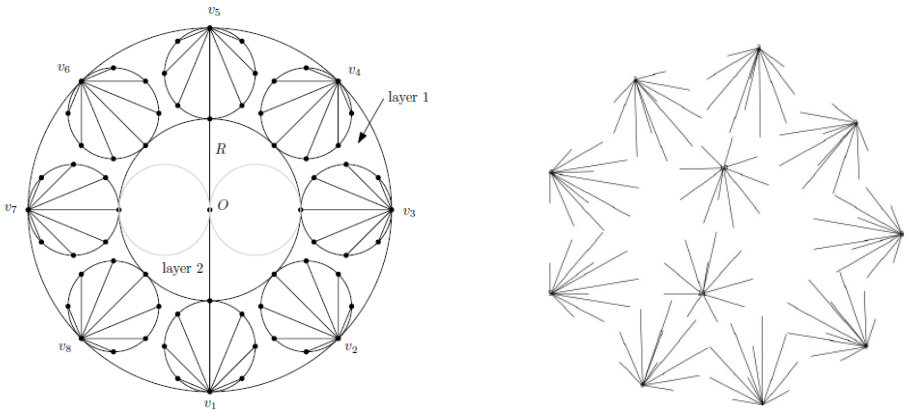
Figure 2: Different drawings of a 6-vertex graph, all using the same vertex position

3 Related Work

Mad at Edge Crossings? Break the Edges!

The first formal definition of PED was given by Bruckdorfer et al. in 2012 [2]. The also introduced different types of PED as SPED and SHPED, which are explained in the previous section. They considered graphs with and without given embedding. For graphs without embedding they showed that a complete graph K_n, thus every graph with n vertices, always admits a $\frac{1}{\sqrt{4n/\pi}}$-SHPED. They also investigated which graphs admit an SHPED when the factor δ is given and came up with the following bound. For $\delta \in [0, 1/2]$, the graph K_n has a SHPED with a factor *delta* if $n \leq n(\delta) = \sum_{i=1}^{m} n_i$ and $m = \lfloor \frac{1}{2\delta} \rfloor$, $\lfloor \frac{\pi}{arcsin(\frac{\delta}{1+\delta-2\delta i})} \rfloor$. They gave an example that showed how with their equation it can be shown that for $\delta = 1/4$, the graph K_{11} has a 1/4-SHPED and gave an construction for such a graph, the idea of the construction and the result can be seen in 3. For graphs with a given embedding they formulated the problem MaxSPED, which aims to maximize the total stub length in a SPED. They gave a integer linear program for MaxSPED. A quick recall, integer linear programming is a program

in which some or all of the variables are restricted to be integers and the constrains and objective functions are linear. Integer linear programming is known to be NP-hard. They presumed that the problem MaxSPED also is NP-hard. This was later shown to be true with a reduction to Planar3SAT [1].



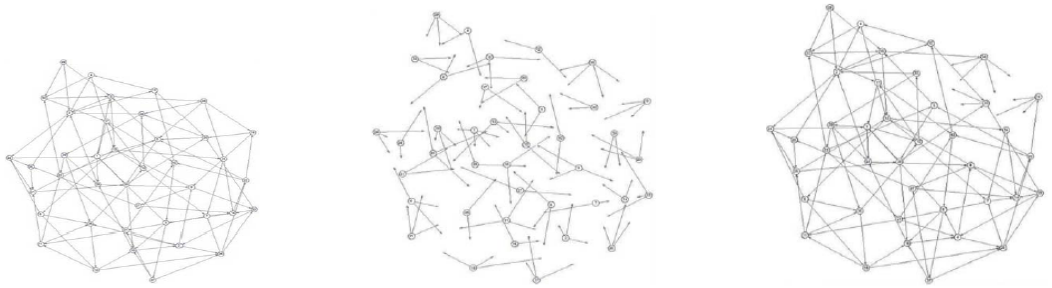
(a) Computing the number of vertices for $\delta = 1/4$

(b) SHPED of K_{11} with $\delta = 1/4$ by construction

Figure 3: Example form Bruckdorfer et al.[2]

Partial link drawings for nodes, links, and regions of interest

Burch et al. integrated the idea of PED into a graph visualization tool as an interaction feature [5]. Additionally to representing the entire graph in the partially drawn link style they allow a user to apply this feature to several nodes or regions of interest. They developed a graph visualization tool first reads a graph dataset from a file. Then a graph layout is chosen and the node-link diagram is generated and displayed to the user. Then several interaction possibilities are available. Such as selecting a region of interest and shortening the stubs to make the diagram more readable. Figure 4 shows an application example of the described tool.



(a) Traditional link-node diagram

(b) Applying the PED feature globally with stub length of 50%

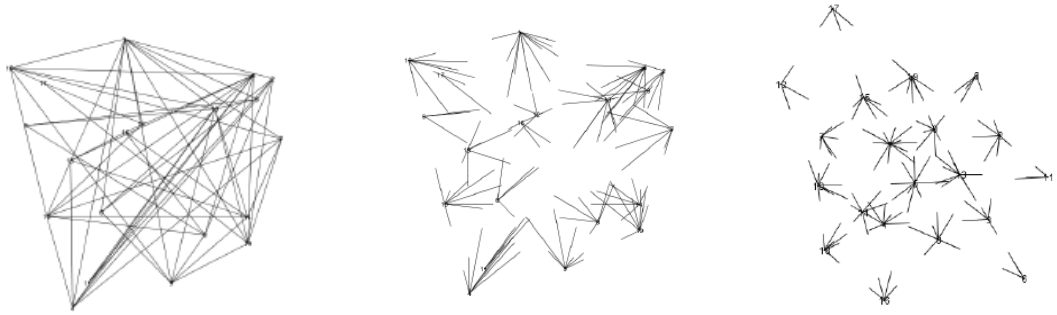
(c) Applying the PED feature to a region of interest (top right vertices) with stub length of 50%

Figure 4: Example from Bruch et al. [5]

A practical approach for 1/4-SHPEDs

In 2015 Bruckdorfer et al. presented their paper "A Practical Approach for 1/4-SHPEDs" [7]. In order to keep homogeneity they allowed crossings in PED and called a 1/4-SHPED which permits crossings 1/4-nSHPED (n=nearly). The goal was to help the reader to identify an adjacency by approximating

the distance due to four times the sub length, by sacrificing the crossing free property of PED. They developed a force-directed algorithm that aims to produce good readable 1/4-nSHPED. They used the idea behind spring embedder for their algorithm. Spring embedder is a force directed placement of vertices that aims for symmetry by spread vertices uniformly for a drawing. Spring embedder itself does not always yield a good result for 1/4-nSHPED. The algorithm they developed uses forces that push vertices away from stubs to resolve crossings. The time complexity of their algorithm lies in $O(nm \log m)$. Figure 5 shows the application of this algorithm.



(a) Traditional straight-line drawing with 383 crossings

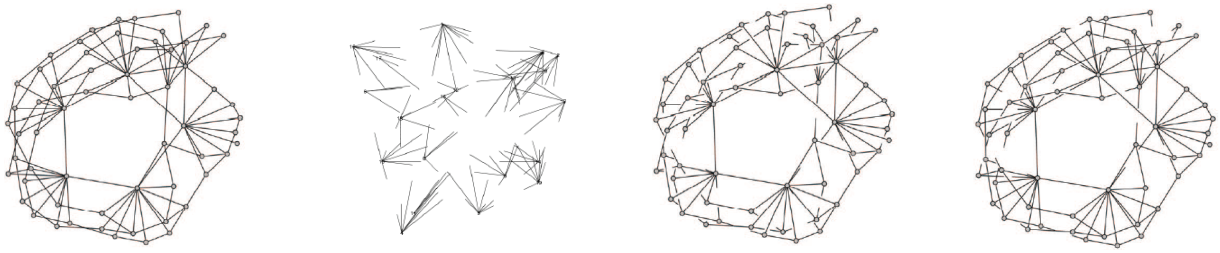
(b) Directly making 1/4-nSHPED of 5a admits 71 crossings

(c) 1/4-nSHPED calculated with their new algorithm having 4 crossings

Figure 5: Example from Bruckdorfer et al. [7]

Partial Edge Drawing: Homogeneity is more Important than Crossings and Ink

That homogeneity is more important than having a crossing free PED was shown by Binucci et al. in 2016 [4]. In some graphs the amount of ink that has to be removed is large to avoid all crossings. They presented heuristics in their paper to produce symmetric PEDs that are either crossing- free or where crossings that form large angles ($< 90\,\text{deg}$) are allowed. The crossing free SPED they produced are called PL-SPEDs and LAC-SPED are SPEDs which only forbid crossings at sharp angles. They compared the PED obtained from their heuristic in a user study to standard 1/4-SHPED and concluded that homogeneity is more important in terms of readability than maximizing ink or avoiding crossings. Figure 6 shows the different PED introduced by Binucci et al.



(a) Traditional straight-line drawing

(b) Standard 1/4-SHPED

(c) PL-SPED

(d) LAC-SPED

Figure 6: Example from Binucci et al.[4]

Progress on Partial Edge Drawings

In 2017 Bruckdorfer et al. gave further bounds for δ-SHPED, they investigated especially 1/4-SHPED. They showed that it is not possible to draw a K_16 which is in a one-sided convex position as 1/4-SHPED. The proof for this bound is described in more detail in the section 4. Then they generalized

this for $\delta \in [0, 1/2]$ and graphs in one-sided convex position and obtained the bound that a graph K_n can not be drawn as δ-SHPED on a one-sided convex position if $n > 10\frac{1}{\delta}\ln\frac{1}{\delta}$. They also showed that it is not possible for any $n > 164$ to draw a 1/4-SHPED. They also investigated some special graph classes and gave the following results.

- The complete bipartite graph $K_{n,n}$ has a δ-SHPED if $n \leq \lfloor\frac{1}{\delta}\rfloor \cdot \lfloor\lfloor\frac{\log 1/2}{\log(1-\delta)}\rfloor\rfloor \in \Phi(\frac{1}{\delta^2})$ where $\lfloor\lfloor x\rfloor\rfloor$ denotes the largest integer that is strictly less than x.
- For any integers $n > 0$ and $k < \log\delta/\log(1-\delta) \in \theta(\frac{1}{\delta}\log\frac{1}{\delta})$ the complete bipartit graph $K_{2k,n}$ has a δ-SHPED.
- A graph of bandwidth k has a $1/(2/\sqrt{2k})$-SHPEd if $2 \leq k \leq n$ and $\sqrt{k}$ and $n/\sqrt{k}$ are integers. If additionally $n/\sqrt{k}$ is even, the k-circulant graph C_n^k has a $1/(6\sqrt{k})$-SHPED.

They gave an algorithm to solve MaxSPED of a given 2-planar drawing efficiently. And provided a 2-approximation algorithm for the dual problem MinSPED, which tries to minimize the amount of erased ink. Both of this algorithms are explained in detail in the section 5.

3.1 Overview

Bounds

[h] Graph	Lower Bound	Upper bound	Reference
K_n (for $\delta = 1/4$)	$n \geq 16$	$n \leq 164$	[Bruckdorfer et. al., 2017]
K_n (for $0 < \delta < 1/2$)	$n > 10\frac{1}{\delta}\ln\frac{1}{\delta}$	-	[Bruckdorfer et. al., 2017]
K_n	$n \geq \lfloor\frac{1}{r\pi\delta^2}\rfloor$	$n \leq \frac{4}{\delta^3}\ln\frac{1}{\delta}$	[Bruckdorfer & Kaufmann, 2012] [Granacher, 2013]

Algorithms

- Integer linear programming for MaxSPED [2]
- Graph visualization tool with interactive PED feature [5]
- Force directed algorithm for 1/4-nSHPED [7]
- Algorithm for PL-SHPED and LAC-SPED [4]
- Algorithm for 2-planar MaxSped [8]
- 2-approximation for MinSPED [8]

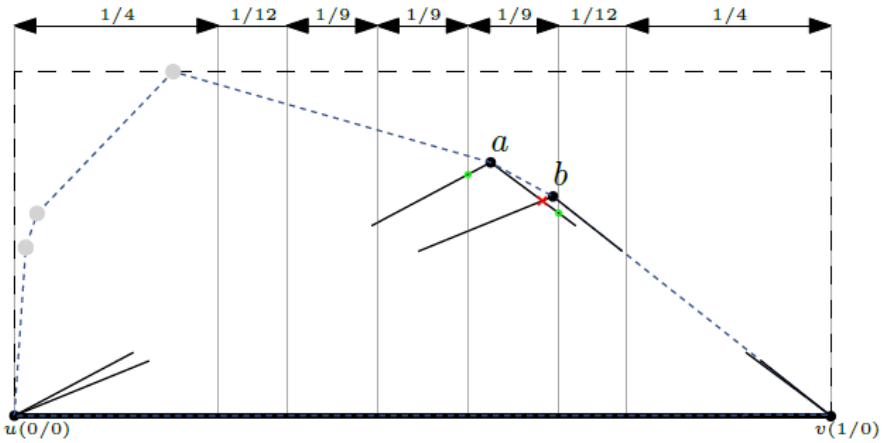


Figure 7: Construction of vertical strips

4 Bounds

One bound for δ-SHPED of a complete graph is exemplarily explained in more detail. This bound was presented by Bruckdorfer et al. in 2017 [8].

Notation
An edge connecting two vertices a and b is denoted with (a, b). The stub of the edge (a, b) which is incident to the vertex a is called ab and the sub incident to b is called ba.

4.1 Upper Bounds for Complete Graphs

In the first proof graphs which are mapped to one-sided convex point sets are discussed. A point set is one-sided convex if the convex hull of the point set contains one edge of the bounding box of the point set. This means one edge of the boundary of the convex hull must be also an edge of the bounding box.

Theorem *There is no set of* 16 *points in one-sided convex position on which the graph* K_{16} *can be embedded as* 1/4*-SHPED.*

The theorem is proofed by assuming the opposite, that it is possible to place 16 points in a one-sided convex position and draw a 1/4-SHPED for this complete graph. To make the proof easy manageable the point set is rotated so that the edge of the convex hull which witnesses the one-sidedness is horizontally and all other points are above this line. The point set is rescaled that the left endpoint u and the right endpoint v of this new baseline are 1 unit apart as shown in figure 7.

The bounding box is split into seven vertical strips of different width. The two outer strips have with 1/4 the next strips 1/12 and the three inner strips 1/9, as displayed in figure 7. We place a point a in an arbitrary inner strip S and draw it's stubs to the points u and v. Both stubs of a intersect the boundaries of the strip the point was placed in. This holds because the width of the strips is chosen in a way that the distance between the left boundary of a strip to the point u is at least three times the strip with. This means even if we place a point as right as possible within a strip the left stub au still intersects the left boundary of S. Analogous is true for the right stub av and the right boundary.

It is not possible to place a second point b in the same strip as a. Lets assume a is the leftmost point in it's strip. The point b can not lie above a or below the stub av otherwise the point set would not be in convex position. Also both subs incident to b intersect the strip boundaries and therefore the stubs of a and b would intersect. PED are defined as crossing free so it is not possible to have

more than one point in an inner strip. This is illustrated in figure 7. Since the construction contains 5 inner strips with this properties at least 11 points would need to be inside the union of the two outer strips.

Assume 6 points are placed in the left strip S_{left}. Let w be the rightmost point in S_{left}. The edge (u, w) is divided into 5 pieces. The two outer pieces have length $1/4 \cdot |(u, w)|$ the neighbouring parts $3/16 \cdot |(u, w)|$ and the middle slice $1/8 \cdot |(u, w)|$. The pieces are connected with the point v so that cones are generated like it is shown in figure 8.

If a point t is placed in S_{left} the stub tv must again intersect the boundary of S_left. Since w is the rightmost point in S_{left} the stub obviously also intefrsects the edge (u, w). Therefore no further points can be placed in the outer cones otherwise the stub tv would intersect with one of the stubs uw or wu. The width of the cones is chosen in a way that the stubs tw and tu intersect the boundaries of the cone. If a new point is placed inside the same cone as t the stubs would cross what is not allowed in a valid partial edge drawing, displayed in figure 8. This shows that there can not be more than 5points in S_{left}. The same is valid for S_{right}.

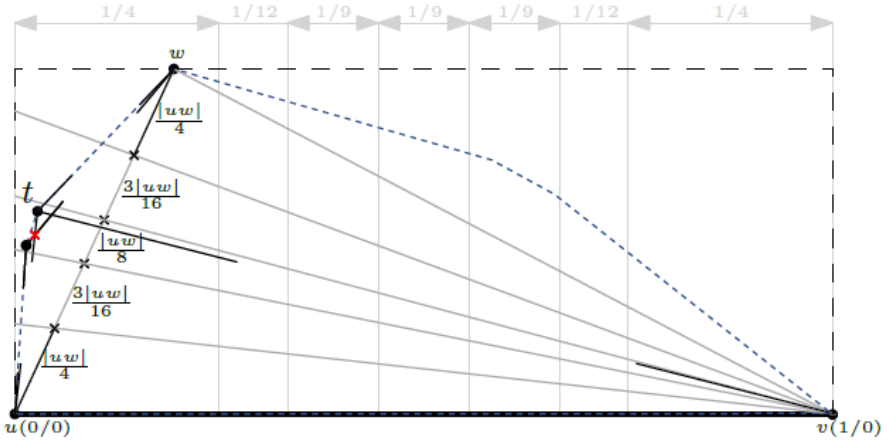


Figure 8: Construction of vertical strips

To sum it up it is not possible to have more than 5 points in the inner strips and at maximum 5 points in each of the outer strips, hence it is not possible to place 16 points in a one-sided convex position and draw a 1/4-SHPED of the complete graph.

Generalization

This construction can be extended to δ-SHPED for any $0 < \delta < 1/2$. From that a more general bound can be obtained. Therefore vertical strips with the same property, that the stubs of a point which lies within a strip have to cross the boundaries of the strip, have to be constructed. This can be achieved by constructing the two outer strips with width δ. Then strips are added in each step, the width of the i-th strip has to be $\delta^2/(1-\delta)^i$. From that a bound of the total number of strips that fit in the bounding box with a given delta can be obtained $2k$ with $k \geq \lceil \frac{\ln 2\delta}{\ln(1-\delta)} \rceil$.

Also cones with the given widths have to be construed. Outermost cones with δ next cones with $\delta(1-\delta)^i$. The total number of cones l can bounded by $l \geq \lceil \frac{\ln \frac{1}{2}}{\ln(1-\delta)} \rceil$.

Combining these two bounds gives: For $n > 10\frac{1}{\delta} \ln \frac{1}{\delta}$ with $0 < \delta < 1/2$ the graph K_n cannot be drawn as a δ-SHPED on a one-sided convex point set.

5 Algorithms

Two algorithms presented in the paper "Progress on Partial Edge Drawings" [8] are explained in more detail.

5.1 Maximizing Ink in 2-Planar Drawings

A maxSPED is a SPED where the total ink used in the straight line drawing is maximized. The problem of computing the maxSPED of a general drawing is NP-hard. This can be shown with a reduction from PLANAR3IN1SAT [**?**]. For two planar drawings this problem can be solved efficiently. Let's recall a two planar drawing allows at maximum 2 crossing per edge. The algorithm for this is presented in this section.

Theorem *Let G be a graph with n vertices, and let Γ be a 2-planar straight line drawing of G. A maxSPED of Γ can be computed in $O(n \log n)$ time.*

From the two planar drawing Γ a intersection graph C is generated. For each edge in Γ which admits at least one crossing a vertex is created in C. The vertices in C are connected iff the corresponding edges in Γ cross. In figure 9 a two planar drawing and it's intersection graph can be seen.

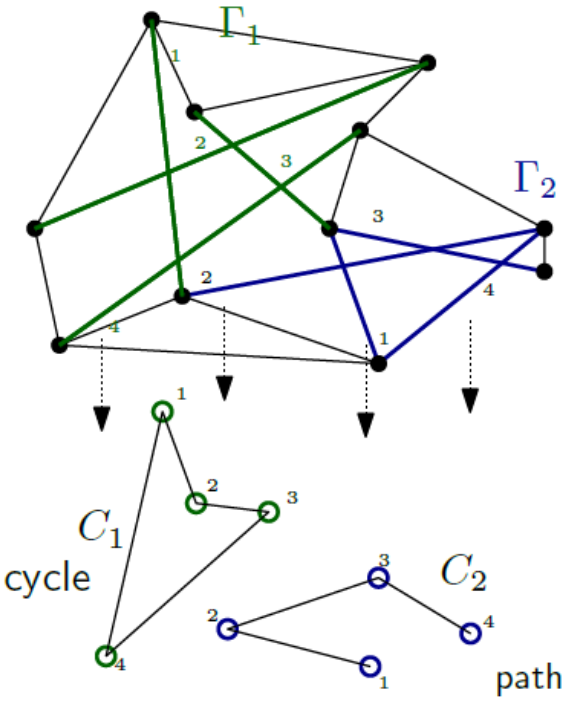


Figure 9: Two planar drawing Γ and corresponding intersection graph C

Since each edge has at most two crossings the maximum degree of each vertex in C is two, hence C can only consist of paths and cycles. Each connected part in C is a sub-graph C_i which is either a path or cycle. In each C_i an ordering of the vertices is defined. If the component is a path the ordering starts at an end node and goes along the path. If C_i is a circle an arbitrary edge in C_i is temporally removed and the ordering can be computed the same way as for a path.

This ordering in the intersection graph is applied to the edges in the given drawing Γ. Now an edge e_j in Γ is intersected by it's successor e_{j+1} for $j = 1, ..., n-1$ and it's predecessor e_{j-1} for $j = 2, ..., n$. This applied ordering can be seen in figure 9. In the case that C_i was a cycle additionally e_n and e_1

intersect.

Each crossing of an edge introduces a stub of this edge. Let l_j be the length of the edge e_j in Γ. The stub of the edge e_j which is caused by the crossing with the edge e_{j-1} is called backwardstub and is denoted with x'_j. The stub introduced by the successor edge e_{j+1} is called forwardstub and denoted with x''_j.

Because in a SPED the two stubs of an edge have to be of equal length, there are three possibilities for each edge. Either the entire edge e_j can be drawn or the two backwardstubs x'_j or the two forwardstubs x''_j.

The number of choices we have for the current edge depends on the decisions we made before. If for example we decided to draw one edge entirely we can only draw the backwardstubs of the next edge. Which choice is the best depends on the rest of the path. To solve this problem dynamic programming is used. Let's recall dynamic programming is a method for solving a problem by dividing it into simpler subproblems and solving each of the subproblems just once and storing their solution. Instead of recomputing the problem the next time it occurs one simple lookup is needed. In this case we always add one edge to our problem until we have a solution for all edges.

For each edge e_j with $j = 1, ..., n$ we compute and store three values:

- $O_{in}(e_j)$ maximum total ink of $e_j, ..., e_n$ under the condition to draw e_j entirely
- $O'_{out}(e_j)$ maximum total ink of $e_j, ..., e_n$ under the condition to draw the two backwardstubs x'_j of e_j
- $O''_{out}(e_j)$maximum total ink of $e_j, ..., e_n$ under the condition to draw the two forwardstubs x''_j of e_j

It has be distinguished between two cases, the sub-graph C_i is a path or a cycle. In the case of a path we compute the intermediate results in a bottom-up visit of the path starting at e_n. Since in a path e_1 does not have a backwardstub $l_1/2$ is used for the value of x'_1. Also e_n does not have a forwardstub, $l_n/2$ is used as value for x''_n.

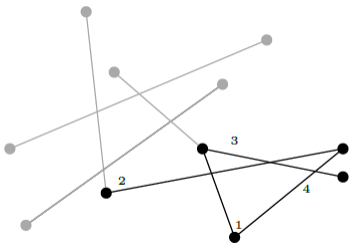
The following equations are used to calculate the intermediate results:

$$O_{in}(e_j) = \begin{cases} l_j + \max\{O'_{out}(e_{j+1}), O''_{out}(e_{j+1})\} & \text{if } x'_{j+1} \geq x''_{j+1} \\ l_j + O'_{out}(e_{j+1}) & \text{if } x'_{j+1} < x''_{j+1} \end{cases}$$

$$O'_{out}(e_j) = \begin{cases} 2x'_j + \max\{O'_{out}(e_{j+1}), O''_{out}(e_{j+1})\} & \text{if } x'_j > x''_j \wedge x'_{j+1} \geq x''_{j+1} \\ 2x'_j + O'_{out}(e_{j+1}) & \text{if } x'_j > x''_j \wedge x'_{j+1} < x''_{j+1} \\ 2x'_j + \max\{O_{in}(e_{j+1}), O'_{out}(e_{j+1}), O''_{out}(e_{j+1})\} & \text{if } x'_j \leq x''_j \end{cases}$$

$$O''_{out}(e_j) = 2x''_j + \max\{O_{in}(e_{j+1}), O'_{out}(e_{j+1}), O''_{out}(e_{j+1})\}$$

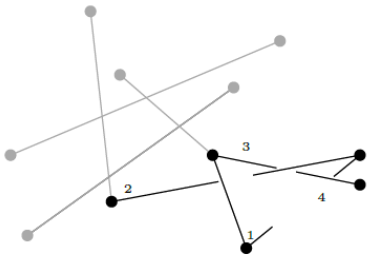
To get the optimal result we have to choose the maximum of O_{in}, O'_{out} and O''_{out} and backtrack which decisions lead to this value. An example for this calculation can be seen in figure 10. This can be applied to the drawing Γ.



(a) 2-planar drawing $\Gamma(G)$

j	l_j	x'_j	x''_j
1	2.5	1.25	0.5
2	6	2.5	1.5
3	3.5	1.5	0.5
4	3.5	1	1.75

(b) 2-planar drawing $\Gamma(G)$



(c) maxSPED

j	O_{in}	O'_{out}	O''_{out}
e_4	3.5	2	3.5
e_3	5.5	5	4.5
e_2	11	10	8.5
e_1	12.5	12.5	12

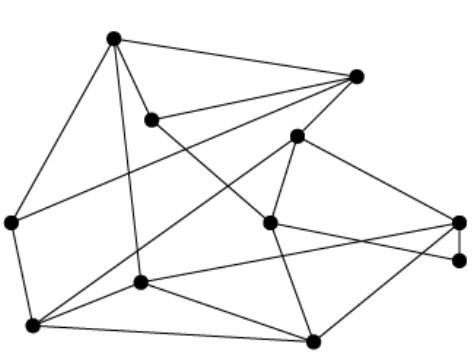
(d) maxSPED

Figure 10: Dynamic programming example for intersection graph C_2 which is path (figure 9)

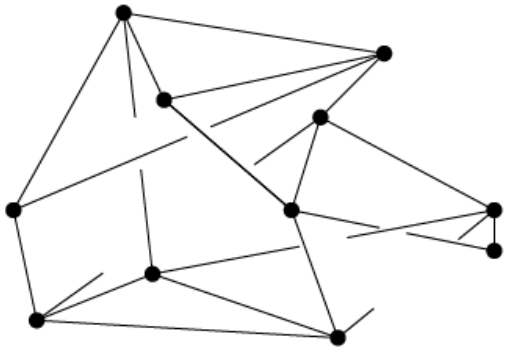
For the calculation of a cycle in the intersection graph C the crossing between the edges e_1 and e_n, which is not represented by the introduced ordering, has to be considered. Therefore the cycle is calculated three times like a path, each time with a different initial condition for the edge e_n. This allows to consider each possibility for the crossing between the first and last edge. From this calculation three tables are obtained from which the total maximum has to be chosen. With this the complete MaxSPED drawing can be computed, an example is shown in figure 11.

Runtime

The construction of the intersection graph C takes $O(m \log m)$ time using a standard sweep-line algorithm for computing the $O(m)$ line-segment intersections. Ordering the vertices in C takes $O(n_C) \in O(m)$ time, with n_C being the number of vertices in C. The bottom up visits and calculation takes $O(m)$ time. This gives us time complexity of $O(m \log m)$. For two-planar graphs $m \in O(n)$ holds [**?**] which results in a overall time complexity of $O(n \log n)$ with n edges in Γ.



(a) 2-planar drawing $\Gamma(G)$



(b) maxSPED

Figure 11: MaxSPED of a 2-planar drawing Γ of a graph G

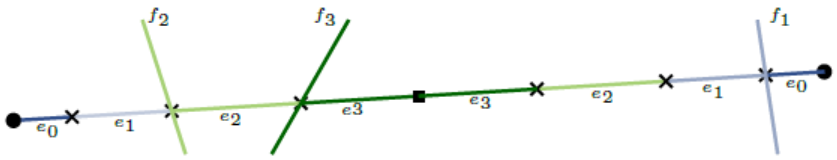


Figure 12: Segment pairs of an edge e and their ordering

5.2 Erasing Ink in Arbitrary Graph Drawings

In this section an approximation algorithm for the MinSPED problem is presented. The MinSPED problem is dual to the MaxSPED and aims to minimize the amount of erased ink in a graph to obtain a valid SPED. The MinSPED problem can be approximated by exploiting a connection to the NP-hard minimum-weight 2-SAT problem (=MinW2SAT). The MinW2SAT problem ask for satisfying variable assignment of a given 2-SAT formula. Additionally real weights are assigned to the variables and the total weight of the true variables should be minimized. There exists an 2-approximation algorithm for this problem. Therefore a instance Φ of MinW2SAT has to be constructed form an instance G of MinSPED.

Theorem *MinSPED can be 2-approximated in $O(I^2)$ time. With I being the number of crossings of the given drawing.*

Assume an edge e with k crossings, the edge is split into $k+1$ pairs of edge segments $e_0, ..., e_{k+1}$. The edges which cross e are ordered by their distance from the crossing point to the closest endpoint of e. Then the pairs of edge segments $e_1, ..., e_{k+1}$ are assigned to the edges which induce them as shown in figure 12. This means the segment pair e_0 of an edge can be drawn always, the other pairs can only be drawn if the assigned edge is not drawn entirely.

A valid PED can only have one continuous stub at each endpoint of an edge, therefore a valid PED is the union of all segment pairs up to some index $j \leq k$. Each segment pair e_i for $i = 1, ..., k$ is modulated as boolean variable $\hat{e_1}, ..., \hat{e_k}$, with the interpretation that a segment pair e_i is **not drawn** if the variable $\hat{e_i}$ is set to **true**. For $i = 1, ..., k$ the following clauses are introduced: $(\neg\hat{e}_{i+1} \Rightarrow \neg\hat{e}_i) \equiv (\hat{e}_{i+1} \vee \neg\hat{e}_i)$. This guarantees that e_{i+10} can be drawn only if e_i is drawn.

For every crossing between two edges e and f the clause $(\hat{e}_i \vee \hat{f}_i)$ is introduced, with e_i being the segment pair introduced by f and f_i the segment pair introduced by e. Because of this clause only one of the segment pairs will be drawn, otherwise the truth assignment would not be minimal according to weight. Hence no crossing between two edges can occur in the resulting drawing.

The weights assigned to the variable $\hat{e_i}$ is the euclidean length of the segment pair e_i. Then minimizing the weight of all valid variable assignments minimizes the amount of erased ink in the SPED. Which gives an 2-approximated MinSPED for the given drawing. This shows that the 2-approximation algorithm for MinW2SAT yields a 2-approximation for the problem MinSPED.

Runtime

MinW2SAT has a time complexity of $O(vc)$ with v being the number of variables and c the amount of clauses. If I is the number of intersections in our graph the algorithm has a time complexity of $O(I^2) \subseteq O(n^4)$ with n equals the number of edges in the given drawing.

6 Hardness

The problem maxSPED has been proven to be NP-hard [1]. In this section the idea for this proof is sketched.

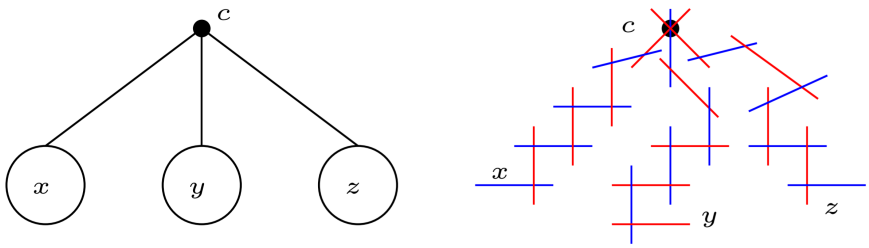


Figure 13: Left: clause with positve variables x, y, z Right: transformed clause contains 3 chains of segments, where red edges are drawn partially and blue ones completely

NP-hardness is shown by providing an reduction to positive planar 1-in-3SAT. Positive planar 1-in-3SAT is a 3 SAT formula, where only positive variables are allowed and for the truth assignment only one variable of each clause can be chosen. In addition it must be possible to connect the variables according to the clauses in a planar way.

For each variable and each clause a vertex is created. If a variable is part of a clause their must be connected by a path of lines and line segments, as displayed in figure 13. A variable is true if the corresponding segment in the clause is drawn entirely. In each clause-vertex three segments meet, since we have a 3SAT formula. In a valid PED no intersections are allowed and therefore only one edge can be drawn entirely, which corresponds to setting one variable in a clause to true. Then one edge is drawn for each variable. The paths which connect the variable with the clause are required to haven even length, if the variable was set to true in all clauses the line inserted for the variable can be drawn entirely since all the segments adjacent are only drawn partially. With the help of this construction from a valid truth assignment a valid maxSPED can be constructed and form a valid maxSPED a valid truth assignment can be obtained.

7 Conclusion

Partial edge drawings are a useful representation method for dense graphs to avoid visual clutter. As shown in this paper they can be used for network visualization and node-link diagrams to make complex situations better understandable for humans and allow them to evaluate the data.

There are several types of PEDs such as SPED, δ-SHPED, 1/4-nSHPED with different properties and divers advantages and disadvantages. The most common and intuitive one is the 1/4-SHPED, which is applicable for many classes of graphs.

To better understand PEDs one issue is to find out if a certain graph admits a δ-SHPED or which δ can be achieved if no embedding is given. As shown in this paper not every graph can be drawn as 1/4-SHPED. Especially Bruckdorfer et al. [8], [2] investigated the geometry of PEDs.

There are several approaches for algorithms like the force driven algorithm or the dynamic programming for 2-planar drawings to generate PEDs.

Some of the open questions and possible future work for PED are:

- Identify other classes of graphs that admit a δ-SHPED for some given δ
- Further investigation of the geometry behind 1/4-SHPED
- Complexity of MaxSPED when the input is a topological or abstract graph
- Experiments with curved or orthogonal links

Two further open problems where already discussed in the research sessions. The complexity of MaxSPED for k-planar drawings with $k = 3$, which we showed that is NP-hard. And an algorithm for MaxSPED for k-planar drawings with $k > 2$. We could come up with an algorithm that works for some classes of intersection graphs.

References

[1] Dipl-Inform Till Martin Bruckdorfer. Schematics of graphs and hypergraphs.

[2] Till Bruckdorfer and Michael Kaufmann. Mad at edge crossings? break the edges! *FUN 2012*, 7288:40–50, 2012.

[3] A. Rusu et al. Using the gestalt principle of closure to alleviate the edge crossing problem in graph drawings. *Proc. 15th Int. Conf. Inform. Visual.*, pages 488–493, 2011.

[4] C. Binucci et al. Partial edge drawing: Homogeneity is more important than crossings and ink. *Int. Conf. Inform. Intell. Syst. Appl.*, pages 1–6, 2016.

[5] M. Burch et al. Partial link drawings for nodes, links, and regions of interest. *18th Int. Conf. Inform. Vis*, pages 53–58, 2014.

[6] R. A. Becker et al. Visualizing network data. *IEEE Trans. Vis. Comp. Graph*, pages 16–28, 1995.

[7] T. Bruckdorfer et al. A practical approach for 1/4-shpeds. *6th Int. conf. Inform. Intell. Syst. Appl.*, pages 1–6, 2015.

[8] Till Bruckdorfer et al. Progress on partial edge drawings. *Journal of Graph Algorithms and Applications*, 21:757–786, 2017.