Minimum Length Embedding of Planar Graphs at Fixed Vertex Locations

Timothy M. Chan, Hella-Franziska Hoffmann, Stephen Kiazyk, and Anna Lubiw

David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
{tmchan,hrhoffmann,skiazyk,alubiw}@uwaterloo.ca

Abstract. We consider the problem of finding a planar embedding of a graph at fixed vertex locations that minimizes the total edge length. The problem is known to be NP-hard. We give polynomial time algorithms achieving an $O(\sqrt{n} \log n)$ approximation for paths and matchings, and an $O(n)$ approximation for general graphs.

1 Introduction

Suppose we want to draw a planar graph and the vertex locations are specified. Such a planar drawing always exists, although not necessarily with straight line edges. Pach and Wenger [1] showed how to construct a drawing using $O(n)$ bends on each edge, where n is the number of vertices. We consider an equally natural optimization criterion—to minimize the total edge length.

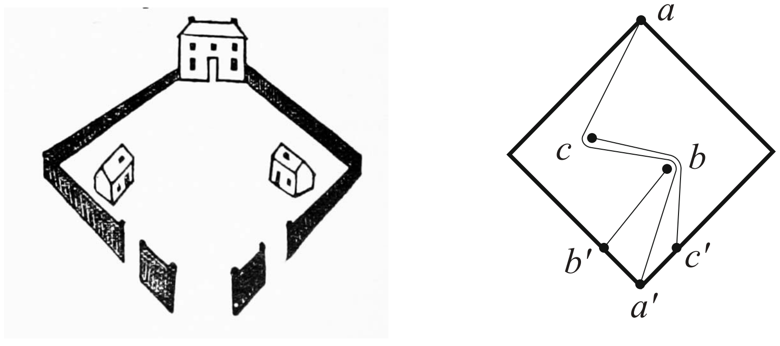

Fig. 1. A puzzle from Loyd [2]—connect each house to the opposite gate with non-crossing paths. On the left is the minimum length solution to an asymmetric version.

For example, Figure 1 shows a puzzle disseminated by Sam Loyd [2, p. 27]. The goal is to connect each house with the gate opposite its door via non-crossing paths. There are two distinct solutions but if the points are shifted as shown on the right in Figure 1, there is a unique shortest solution.

© Springer International Publishing Switzerland 2013

In this example the fixed outer wall plays a significant role, so the example demonstrates a more general problem—to extend a planar drawing of a subgraph to a planar drawing of the whole graph minimizing the total length. Fixed edges in the drawing act as *obstacles*. We call this *Minimum Length Planar Drawing of Partially Embedded Graphs*.

Angelini et al. [3] gave a linear time algorithm to decide if a planar drawing of a subgraph can be extended to a planar drawing of the whole graph. Our problem is the optimization version, to minimize the total edge length.

We will restrict attention to the case where all vertex positions are fixed. Furthermore, most of our results are for the case when none of the edges are fixed. We call this *Minimum Length Planar Drawing [or Embedding] at Fixed Vertex Locations*. This problem is very interesting even for special graphs such as matchings and paths.

When the edges to be added form a matching, the problem is to join specified pairs of points via non-crossing paths of minimum total length. The case when there are no obstacles was considered by Liebling et al. [4] in 1995. They gave some heuristics based on finding a short non-crossing tour of the points and then "wrapping" the matching edges around the tour. We use this same technique for our approximation results. They also proved that for points in the unit square a shortest non-crossing matching has length $O(n\sqrt{n})$ and there are examples realizing this bound. The lower bound (due to Peter Shor) relies on the existence of expander graphs with large crossing number. In 1996 Bastert and Fekete [5] proved that the problem is NP-hard, even with no obstacles. There is also substantial work on the case where there are obstacles (i.e. when some edges are fixed)—specifically when the points lie on the boundary of a polygon [6], or multiple polygons [7] (in which case the run time is exponential in the number of polygons). We give more details below in Section 1.1, and also discuss related work on finding "thick" paths that are separated from each other.

The problem of Minimum Length Planar Embedding at Fixed Vertex Locations is also interesting when the graph we want to embed is a path. This version of the problem was formulated by Polishchuk and Mitchell [8]. Their main goal was to find a minimum length tour that visits a given sequence of convex bodies in $\mathbb{R}^d$ (see [9] for the planar case), without regard to whether the path is self-intersecting, but in their conclusion section they ask about finding a minimum length non-crossing tour for a sequence of points.

Our Contributions. We give polynomial time approximation algorithms for Minimum Length Planar Embedding at Fixed Vertex Locations. In the case of general planar graphs we achieve an approximation ratio of $O(n)$. In the case of a matching or a path we achieve an approximation ratio of $O(\sqrt{n}\log n)$. Our main technique is to route graph edges around a carefully chosen path or tree defined on the input points in the plane.

1.1 Related Work

Bastert and Fekete [5] prove that Minimum Length Planar Drawing at Fixed Vertex Locations is NP-hard when the graph is a matching.

Patrignani [10] proved that it is NP-hard to decide if a planar drawing of a subgraph can be extended to a planar straight-line drawing of the whole graph.

Papadopoulou [6] gave an efficient algorithm for finding minimum length non-crossing paths joining pairs of points on the boundary of a polygon. In this case each path is a shortest path, but Papadopoulou finds them more efficiently than the obvious approach. Erickson and Nayyeri [7] extended this to points on the boundaries of h polygonal obstacles. Their algorithm has a running time that is linear for fixed h, but grows exponentially in h.

The difficulty with multiple polygons is deciding which homotopy of the paths gives minimum length. If the homotopy is specified the problem is easy [11,12].

In the aforementioned results on shortest non-crossing paths, one issue is that paths will overlap in general even though crossings are forbidden (see Figure 1 for an example). In practical applications we often need paths that are disjoint and maintain some minimum separation from each other. This issue is addressed in papers about drawing graphs with "thick" edges. Duncan et al. [13] show how to find thick shortest homotopic paths. Polishcuk and Mitchell [14] show how to find shortest thick disjoint paths joining endpoints on the boundaries of polygonal obstacles (with exponential dependence on the number of obstacles). They also show hardness results, including hardness of approximation.

In our problem the correspondence between the vertices and the fixed points in the plane is given. There is a substantial body of work where the correspondence of vertices to points is not fixed. Cabello [15] showed that it is NP-hard to decide if there is a correspondence that allows a straight-line planar drawing. Many special cases have been classified as polynomial time or NP-complete. A related problem is to find small *universal point sets* on which all planar graphs can be straight-line embedded (see [16]).

A problem related to minimum length planar embedding at fixed vertex locations is to draw planar graphs so that each edge is a monotone path of axis-parallel line segments. Any such path is a shortest path in the L_1 or Manhattan metric, and these drawings are called Manhattan-geodesic embeddings. This concept was introduced by Katz et al. [17]. They considered the case where the graph is a matching and showed that the problem is NP-hard when the drawing is restricted to a grid, but solvable in polynomial time otherwise.

We restricted the general problem of extending a partial planar embedding to the case where all vertex positions are fixed. The case where some vertices are free to move is also very interesting and is related to work on Steiner trees with fixed tree topology [18] and Steiner trees with obstacles [19]. Finally, one may consider drawing graphs at fixed vertex locations but allowing edges to cross. This is interesting and non-trivial when crossings must have large angles [20].

For other geometric graph augmentation problems see the survey by Hurtado and Tóth [21].

1.2 Definitions and Basic Observations

We consider the following problem called *Minimum Length Planar Embedding at Fixed Vertex Locations:* Given a planar graph $G = (\{v_1, \ldots, v_n\}, E)$ and a set of points $P = \{p_1, \ldots, p_n\}$, find a planar embedding of G in the plane that places vertex v_i at point p_i and minimizes the total edge length.

Edges of the embedding are allowed to overlap, but they must be non-crossing (i.e. infinitesimally deformable into disjoint paths). In the following we will refer to the vertices and their respective fixed locations interchangeably. The Euclidean distance between two points $p, q \in \mathbb{R}^2$ is denoted $d(p, q)$.

Observation 1. *$L = \sum_{(v_i,v_j)\in E} d(p_i, p_j)$ is a trivial lower bound for the total length of any planar embedding of graph G at fixed vertex locations P.*

One approach for finding short embeddings at fixed vertex locations is to draw each edge (v_i, v_j) as a curve whose length is within a constant factor of distance $d(p_i, p_j)$. Unfortunately such a planar drawing does not always exist; see Fig. 2.

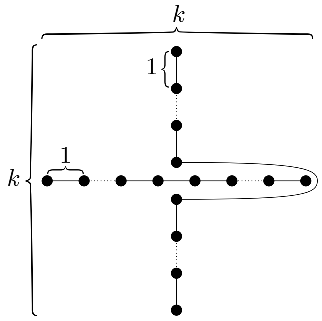

Fig. 2. Example for which any solution contains at least one edge of length greater than $k = n/2 - 1$

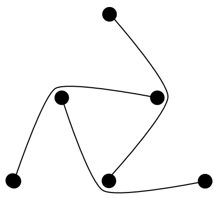

Fig. 3. Example for which any optimal solution contains no straight line edges

The example in Fig. 3 shows an instance where no straight line edge is included in any optimal solution, which means that obvious greedy algorithms for the problem fail. This was first observed by Liebling et al. [4].

Note that any edge of an optimal embedding bends only at vertex locations. Thus any optimal embedding lives in some underlying triangulation of the point set. Given the triangulation, the problem becomes that of finding short non-crossing paths in a planar graph. This problem was first proposed by Takahashi et al. [22] who considered the case of terminal points on two faces. Erickson and Nayyeri [7] say that the general problem is NP-hard, citing Bastert and Fekete [5]. Unfortunately, we cannot find this result in the version of the report that we have.

Instead of fixing the underlying triangulation, we use a carefully chosen path or tree as the layout for our embeddings.

2 Embedding a Path or Matching

In this section we give polynomial time approximation algorithms for the case where G is a path or a matching. Our starting point is a 1-dimensional version of the problem that will be the basis for all further results. We give a polynomial time (exact) algorithm for the case where G is a path, and the points lie on a line. We note that Liebling et al. [4] apparently knew the analogous result for the case when G is a matching, since they say that the edges of a matching can be "wrapped around" any non-self-intersecting tour of the points. They give no details of how to do the wrapping, and we consider the details worth explaining.

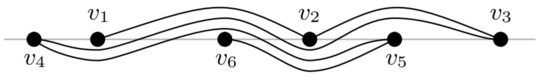

Fig. 4. A minimum length embedding of a path on fixed points that lie on a line. Edges are drawn with gaps between them for clarity only.

Lemma 1. *There is a polynomial time algorithm to find a minimum length embedding of a path on fixed vertex positions that lie on a line.*

Proof. Without loss of generality, assume that all the points lie on a horizontal line. See Fig. 4. This allows us to speak of "above" and "below". We draw the edges in order along the path. Draw edge (v_i, v_{i+1}) as a curve from point p_i to point p_{i+1} that stays below all edges drawn so far, but stays above all later points $p_j, j > i + 1$. This ensures that later edges of the path can reach their endpoints without crossing earlier edges. $\square$

As noted by Liebling et al., this idea of "weaving" the edges through the points can be extended to the the case where the points lie on a simple (i.e. non-self-intersecting) curve in the plane. (In fact, the idea even extends to a tree, as we shall see in Section 3.) If one can find a simple curve C passing through every point in P, then "weaving" the edges of the path along the curve creates a path of length at most

$$\sum_{i=1}^{n-1} d_C(p_i, p_{i+1}), \tag{1}$$

where $d_C(p_i, p_{i+1})$ denotes the length of the subcurve of C from p_i to p_{i+1}. This sum is trivially upper-bounded by n times the length of C.

We can choose the curve C to be an $O(1)$-approximation to the minimum-length Hamiltonian path (i.e., the traveling salesman path) for P (e.g., the simplest option would be the standard 2-approximate solution obtained from the minimum spanning tree). Since the length of the traveling salesman path is a lower bound for the problem in the path case, this would give an $O(n)$-approximate solution overall.

In Section 3 we will extend this idea to obtain an $O(n)$ approximation for general planar graphs. In the remainder of the current section we show how to improve the approximation factor for a path or matching by choosing a better curve C. The property we need is that points that are close in the plane are

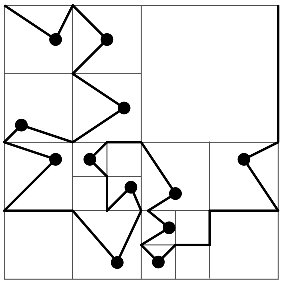

Fig. 5. The curve that is used as routing layout for embedding paths and matchings

close on the curve. The idea is to use a construction based on *shifted quadtrees*, similar to certain well known families of space-filling curves such as the *Z-order curve* or the *Hilbert curve.*

Let D_0 be the diameter of P. Without loss of generality, assume that $P \subset [0, D_0]^2$. We initially shift all the points in P by a random vector $v \in [0, D_0]^2$. Now, $P \subset [0, 2D_0]^2$.

Given a square S, the following procedure returns a simple polygonal curve, with the property that the curve starts at one corner of S, ends at another corner, and stays inside S while visiting all points of $P \cap S$.

```
Curve($S$):
1.  if $|P \cap S| \le 1$ then
2.      return a curve with the stated property, using at most 2 line segments
3.  divide $S$ into 4 subsquares $S_1, \ldots, S_4$
4.  for $i = 1, \ldots, 4$, compute $C_i = $ Curve($S_i$)
5.  return a curve with the stated property by joining $C_1, \ldots, C_4$,
    using $O(1)$ connecting line segments
```

Slight perturbation may be needed in line 5 to ensure that we obtain a simple curve. There is flexibility as to which corner we choose to start or end. (If we always start at the upper left corner and end at the lower right corner, the construction is similar to the Z-order curve. If we choose starting and ending corners to be adjacent in a manner similar to the Hilbert curve instead, the connecting line segments in line 5 may even be avoided; see Fig. 5. The main difference with standard space-filling curves is that we terminate the recursion as soon as we reach a square containing zero or one point.)

Lemma 2. *The length of the curve returned by* Curve(S) *is at most* $O(D_S\sqrt{n_S})$, *where* $n_S = |P \cap S|$ *and* D_S *is the side length of* S.

Proof. Let L_i be the sum of the lengths of all line segments generated in line 2 or 5 at the i-th level of the recursion. The squares at the i-th level have side length $D_S/2^i$, and the number of squares at the i-th level is upper-bounded by both 4^i and n_S. Thus,

$$L_i \;\le\; O(D_S/2^i) \cdot \min\{4^i, n_S\} \;=\; O(\min\{D_S 2^i,\; D_S n_S/2^i\}).$$

The total length of the curve is then at most

$$\sum_{i=0}^{\infty} L_i \;\le \sum_{i=0}^{\lceil (1/2)\log n_S\rceil -1} D_S 2^i \;+ \sum_{i=\lceil (1/2)\log n_S\rceil}^{\infty} D_S n_S/2^i \;=\; O(D_S\sqrt{n_S}). \qquad \square$$

We now let C be the curve returned by CURVE(S_0) with $S_0 = [0, 2D_0]^2$. All the squares generated by the recursive calls are *quadtree squares*.

Define $D(p,q)$ to be the side length of the smallest quadtree square enclosing p and q. Note that $D(p,q) \ge d(p,q)/\sqrt{2}$. It is known that after random shifting, $D(p,q)$ approximates $d(p,q)$ to within a logarithmic factor in expectation (e.g., see [23, Lemma 5.1]). We include a quick proof for the sake of completeness.

Lemma 3. *For a fixed pair of points $p, q \in P$,*

$$\mathbf{E}[D(p,q)] \le O(\log(D_0/d(p,q))) \cdot d(p,q).$$

Proof. $D(p,q) > D_0/2^i$ if and only if $\overline{pq}$ crosses a horizontal or vertical grid line in the grid formed by the quadtree squares of side length $D_0/2^i$. The probability that this happens is $O\left(\frac{d(p,q)}{D_0/2^i}\right)$. Thus,

$$\mathbf{E}[D(p,q)] \;\le\; O\left(\sum_{i=0}^{\lceil \log(D_0/d(p,q))\rceil} \frac{d(p,q)}{D_0/2^i} \cdot D_0/2^i\right) \;\le\; O(\log(D_0/d(p,q))) \cdot d(p,q).$$

$\square$

Lemma 4. *For a fixed pair of points $p, q \in P$,*

$$\mathbf{E}[d_C(p,q)] \le O(\sqrt{n}\log(D_0/d(p,q))) \cdot d(p,q).$$

Proof. The portion of the curve C from p to q lies inside a quadtree square with side length $D(p,q)$. Thus, by Lemma 2, $d_C(p,q)$ is at most $O(D(p,q)\sqrt{n})$. The conclusion then follows from Lemma 3. $\square$

Theorem 2. *For a path G with fixed vertex locations, there is a polynomial-time randomized algorithm which computes a planar embedding of expected length at most $O(\sqrt{n}\log n) \cdot L$ where $L = \sum_{(p,q)\in E} d(p,q)$.*

Proof. By (1) and linearity of expectation, we obtain an embedding of expected length at most $(1+\epsilon)\sum_{(p,q)\in E} \mathbf{E}[d_C(p,q)]$. By Lemma 4, this quantity is at most

$$\sum_{(p,q)\in E} O(\sqrt{n}\log(D_0/d(p,q))) \cdot d(p,q) \;\le\; O(\sqrt{n}\log(nD_0/L)) \cdot L$$

because the logarithm function is concave. The theorem follows since $L \ge \Omega(D_0)$ for the case of a path G. $\square$

Since L is a lower bound on the optimal cost, we obtain an $O(\sqrt{n}\log n)$-approximation algorithm for the case of a path G. For the case of a matching, we obtain the same result with just slightly more effort:

Theorem 3. *For a matching G with fixed vertex locations, there is a polynomial-time randomized algorithm which computes a planar embedding of expected length at most $O(\sqrt{n}\log n) \cdot L$ where $L = \sum_{(p,q)\in E} d(p,q)$.*

Proof. We can use essentially the same algorithm. The cost of the solution is still bounded by (1) and the same analysis goes through, except that $L \geq \Omega(D_0)$ may no longer be true.

Observe that if there is a vertical line that separates the line segments $\{\overline{pq} : (p,q) \in E\}$ into two nonempty parts, then we can just recursively compute a planar embedding on both sides, since each embedding can be shrunk to lie within the minimum (axis-aligned) bounding box of its points. We may thus assume that no such vertical separating line exists, which implies that L is at least the width of the bounding box of P. Similarly, we may assume that no horizontal separating line exists, which implies that L is at least the height of the bounding box of P. These two assumptions imply $L \geq \Omega(D_0)$ and we may proceed as before. □

Remarks. To obtain a time bound not sensitive to the bit complexity of the input, we can adopt a variant of the method where we compress long chains of degree-1 nodes in the tree (called the *compressed quadtree*), to ensure that the number of recursive calls is $O(n)$.

On the other hand, if input coordinate values are $O(\log n)$ bits long, we can derandomize the algorithm in polynomial time by trying all possible shifts.

The upper bound relative to L in Theorems 2 and 3 is tight up to a logarithmic factor: Liebling et al. [4] provide examples (due to Peter Shor) with points in the unit square for which any shortest non-crossing matching has length $\Omega(n\sqrt{n})$, proving a lower bound of $\Omega(\sqrt{n}) \cdot L$.

3 Embedding General Planar Graphs

In this section we give an $O(n)$-approximation algorithm for constructing a planar embedding of a planar graph G at fixed vertex locations P.

The construction is based on the algorithm by Pach and Wenger [1] for finding a planar polygonal embedding of a graph with fixed vertex locations and with $O(n)$ bends per edge. Pach and Wenger draw the edges of the graph by tracing around a tree of n edges drawn in the plane. Each edge of the graph is drawn as a curve that walks around the tree a constant number of times, which gives the bound of $O(n)$ bends per edge. For their tree Pach and Wenger use a star with a leaf at each vertex.

In our case we want to bound the length of each edge, which can be done by bounding the length of the tree. We cannot use a star; instead, we will use a tree that is a subset of the (non-planar) drawing of G where each edge is drawn as a straight line segment. This ensures that the tree has total length at most $L = \sum_{(p,q)\in E} d(p,q)$. Because of connectivity issues, we will actually use a set of disjoint trees:

Lemma 5. *Given a graph G, and fixed vertex locations P, we can construct in $O(n^2)$ time an embedded forest F with $O(n)$ vertices and total length at most L, such that for every edge (p,q) in G, p and q are in the same tree of F.*

Proof. We construct the forest F iteratively by adding edges of the graph one by one. For each edge $(p,q) \in E$ we will add some subsegments of the line segment pq to F. The forest will be a subset of an arrangement of $O(n)$ lines. The arrangement can be constructed in $O(n^2)$ time [24]. Consider edge (p,q). If p and q are already in the same tree of F, we are done. Otherwise consider the line segment pq. It crosses at most n segments of F, and these crossing points subdivide it into $p = p_1, p_2, \ldots, p_k = q$ with $k \leq n$. We treat these segments one by one in order. Consider segment p_ip_{i+1}. If p_i and p_{i+1} are already in the same tree of F, we are done. Otherwise we add segment p_ip_{i+1} to F. Fig. 6 illustrates this idea. We use a union-find data structure to test if points are in the same tree of F. By construction, the length of F is bounded by L. Furthermore, we only add a segment when we join two trees of F, and this can happen at most n times. Thus F has $O(n)$ vertices and is a subset of an arrangement of n lines. □

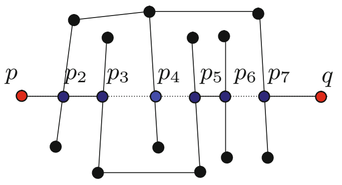

Fig. 6. Constructing the forest F in Lemma 5. Segment pq crosses multiple components of F. Segments p_3p_4, p_4p_5, and p_6p_7 are not added to F.

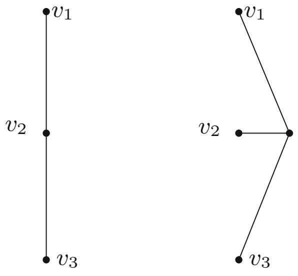

Fig. 7. Perturbing the tree to change v_2 from an internal vertex to a leaf.

Theorem 4. *Given a planar graph G with n vertices and fixed vertex locations P, there is an $O(n^2)$-time approximation algorithm to construct a planar embedding of G on P with total length $O(n) \cdot L$ where $L = \sum_{(p,q)\in E} d(p,q)$.*

Proof. Use Lemma 5 to construct a forest F, that will serve as the basis for our edge routing. Because we do not want paths to travel through intermediate vertices, we perturb the trees in F slightly so that each vertex of G is a leaf of the tree that contains it. See Fig. 7. This can be done while keeping the trees disjoint and of total length $O(L)$.

Consider a single tree T of the forest F, together with the induced graph G_T on the vertices of G that lie in T. We will follow the approach of Pach and Wenger and draw the edges of G_T as paths hugging the tree T. Because every edge of G lies in some G_T, and the trees are disjoint (as objects in the plane), it suffices to describe the solution for a single tree T. To simplify notation, we will assume for the remainder of the proof that we have a single tree T and $G = G_T$.

We now follow Pach and Wenger's solution, the main difference being that we have a more general tree than their star. We outline their solution and remark on the modifications required for our situation.

Pach and Wenger's solution is based on a Hamiltonian cycle that they construct by adding vertices and edges to the graph. Specifically, they subdivide each edge of the graph by at most two new vertices and add some edges between vertices to obtain a planar graph with a Hamiltonian cycle [1, Lemma 5]. (Note that the new edges do not appear in the final drawing.) The Hamiltonian cycle C partitions edges of the planar graph relative to some (arbitrary) planar embedding into the edges *inside* C and the edges *outside* C. They first draw the edges of the Hamiltonian cycle C and then draw the inside and outside edges.

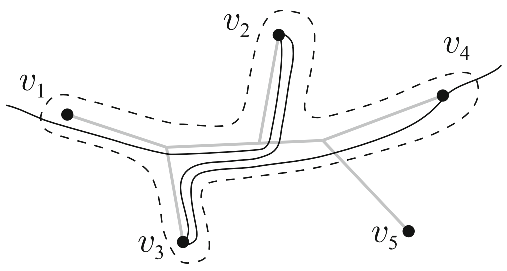

Fig. 8. Drawing the graph G around the tree T (drawn in gray) whose leaves are the graph vertices. The portion of the Hamiltonian cycle C from v_1 to v_4 is drawn as a solid curve. The dashed curve Λ_4 surrounds T_4 and is split by C into two paths between v_4 and v_1, one inside C and one outside C.

To draw the edges of C they use an approach similar to the weaving technique described in Lemma 1. Renumber vertices so they appear in the order $v_1, v_2, \ldots, v_n$ along the Hamiltonian cycle. Some of these are new vertices that were added to create the Hamiltonian cycle. Pach and Wenger assign arbitrary locations to the new vertices, but we locate them very close to the tree T, adding them as leaves of T and keeping the length of T in $O(L)$. We will use v_i to refer to the vertex of G, the corresponding point in the plane, and the corresponding leaf of T. Edge (v_{i-1}, v_i) of C will be drawn around a subtree T_i of T. We define T_i more carefully for our situation: T_i is the connected subtree of T induced on leaves $v_1, v_2, \ldots, v_i$. With these modifications, the rest of Pach and Wenger's solution applies unaltered.

As they draw C they add (multiple copies of) auxiliary paths Λ_i from v_i to v_1, one inside and one outside C. See Fig. 8. Then each edge (v_i, v_j) of G inside [outside] C is routed using the paths inside [outside] C from v_i to v_1 and from v_1 to v_j. For further details please refer to their paper [1]. The end result is a planar drawing of G on vertex locations P. Every original edge e of G has been subdivided by at most two new vertices, and each of the resulting three edges has been drawn as two paths in the tree. The total length of the drawing of e is therefore bounded by 6 times the length of T, and thus in $O(L)$.

Pach and Wenger's algorithm takes $O(n^2)$ time so our overall running time is $O(n^2)$ as well. □

4 Conclusion and Open Problems

The problem of drawing a planar graph at fixed vertex locations while minimizing the total edge length seems to be very difficult although we are not aware of any hardness of approximation results. In fact, for the case of a path, even an NP-hardness result is lacking. Our algorithms achieve approximation factors of $O(n)$ for general graphs and $O(\sqrt{n}\log n)$ for paths and matchings. Besides the obvious question of improving these approximation factors (or proving hardness), we suggest looking at: (1) the problem of drawing a graph at fixed vertex locations with thick edges; and (2) looking at the case where some vertex locations are not fixed, which is related to drawing Steiner trees with fixed topology [18].

Acknowledgments. This work was done in the Algorithms Problem Session at Waterloo and we thank the other participants for good discussions. We learned about Sam Loyd's disjoint paths puzzle (which is not original to him) from Marcus Schaefer who has studied the history of such planarity puzzles.

References

1. Pach, J., Wenger, R.: Embedding planar graphs at fixed vertex locations. Graphs and Combinatorics 17(4), 717–728 (2001)
2. Sam Loyd, J.: Sam Loyd's Cyclopedia of 5000 Puzzles Tricks and Conundrums. Lamb Publishing Company (1914)
3. Angelini, P., Di Battista, G., Frati, F., Jelínek, V., Kratochvíl, J., Patrignani, M., Rutter, I.: Testing planarity of partially embedded graphs. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 202–221 (2010)
4. Liebling, T.M., Margot, F., Müller, D., Prodon, A., Stauffer, L.: Disjoint paths in the plane. ORSA Journal on Computing 7(1), 84–88 (1995)
5. Bastert, O., Fekete, S.P.: Geometric wire routing. Technical Report 332, Angewandte Mathematik und Informatik, Universität zu Köln (1996) (in German)
6. Papadopoulou, E.: k-pairs non-crossing shortest paths in a simple polygon. In: Nagamochi, H., Suri, S., Igarashi, Y., Miyano, S., Asano, T. (eds.) ISAAC 1996. LNCS, vol. 1178, pp. 305–314. Springer, Heidelberg (1996)

7. Erickson, J., Nayyeri, A.: Shortest non-crossing walks in the plane. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 297–308 (2011)
8. Polishchuk, V., Mitchell, J.S.: Touring convex bodies – a conic programming solution. In: Proceedings of the 17th Canadian Conference on Computational Geometry, pp. 290–293 (2005)
9. Dror, M., Efrat, A., Lubiw, A., Mitchell, J.S.B.: Touring a sequence of polygons. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 473–482 (2003)
10. Patrignani, M.: On extending a partial straight-line drawing. International Journal of Foundations of Computer Science 17(5), 1061–1069 (2006)
11. Bespamyatnikh, S.: Computing homotopic shortest paths in the plane. Journal of Algorithms 49(2), 284–303 (2003)
12. Efrat, A., Kobourov, S.G., Lubiw, A.: Computing homotopic shortest paths efficiently. Computational Geometry 35(3), 162–172 (2006)
13. Duncan, C.A., Efrat, A., Kobourov, S.G., Wenk, C.: Drawing with fat edges. International Journal of Foundations of Computer Science 17(5), 1143–1164 (2006)
14. Mitchell, J.S., Polishchuk, V.: Thick non-crossing paths and minimum-cost flows in polygonal domains. In: Proceedings of the 23rd Annual Symposium on Computational Geometry (SoCG), pp. 56–65 (2007)
15. Cabello, S.: Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. Journal of Graph Algorithms and Applications 10(2), 353–363 (2006)
16. Dujmović, V., Evans, W.S., Lazard, S., Lenhart, W., Liotta, G., Rappaport, D., Wismath, S.K.: On point-sets that support planar graphs. Computational Geometry 46(1), 29–50 (2013)
17. Katz, B., Krug, M., Rutter, I., Wolff, A.: Manhattan-geodesic embedding of planar graphs. In: Eppstein, D., Gansner, E.R. (eds.) GD 2009. LNCS, vol. 5849, pp. 207–218. Springer, Heidelberg (2010)
18. Hwang, F., Weng, J.: The shortest network under a given topology. Journal of Algorithms 13(3), 468–488 (1992)
19. Winter, P.: Euclidean Steiner minimal trees with obstacles and Steiner visibility graphs. Discrete Applied Mathematics 47(2), 187–206 (1993)
20. Fink, M., Haunert, J.-H., Mchedlidze, T., Spoerhase, J., Wolff, A.: Drawing graphs with vertices at specified positions and crossings at large angles. In: Rahman, M. S., Nakano, S.-i. (eds.) WALCOM 2012. LNCS, vol. 7157, pp. 186–197. Springer, Heidelberg (2012)
21. Hurtado, F., Tóth, C.: Plane geometric graph augmentation: A generic perspective. In: Pach, J. (ed.) Thirty Essays on Geometric Graph Theory, pp. 327–354. Springer, New York (2013)
22. Takahashi, J., Suzuki, H., Nishizeki, T.: Shortest noncrossing paths in plane graphs. Algorithmica 16(3), 339–357 (1996)
23. Kamousi, P., Chan, T.M., Suri, S.: Stochastic minimum spanning trees in Euclidean spaces. In: Proceedings of the 27th Annual ACM Symposium on Computational Geometry (SoCG), pp. 65–74 (2011)
24. Edelsbrunner, H., O'Rourke, J., Seidel, R.: Constructing arrangements of lines and hyperplanes with applications. SIAM Journal on Computing 15(2), 341–363 (1986)