Lines: 225

Splitting Vertices in 2-Layer Graph Drawings

Reyan Ahmed[1], Patrizio Angelini[2], Giuseppe Di Battista[3], Philipp Kindermann[4], Karsten Klein[5], Stephen Kobourov[1], Martin Nöllenburg[6], Antonios Symvonis[7], and Markus Wallinger[6]

[1] University of Arizona, Tucson, AZ, USA
[2] John Cabot University, Rome, Italy
[3] Roma Tre University, Rome, Italy
[4] Universität Trier, Trier, Germany
[5] University of Konstanz, Germany
[6] TU Wien, Vienna, Austria
[7] NTUA, Athens, Greece

Abstract. Crossing minimization in 2-layer graph drawings is a prominent problem, but obtaining a drawing with no or just a few crossings is often an illusive goal. We propose to use vertex splitting as an operation to reduce the number of crossings by replacing selected vertices on one layer by two (or more) copies and suitably distributing their incident edges among the copies. We study the problems of minimizing the number of splits and minimizing the number of split vertices to obtain a crossing-free drawing. We prove NP-hardness results when both layers are permutable, and give linear-time algorithms when one layer is fixed in the input.

1 Introduction

In a *2-layer graph drawing* of a bipartite graph $G = (V_t \cup V_b, E)$, with $V_t \cap V_b = \emptyset$ and $E \subseteq V_t \times V_b$, vertices are drawn as points on two distinct parallel lines ℓ_t and ℓ_b, and edges are drawn as straight-line segments [16]. The vertices in V_t (*top vertices*) lie on ℓ_t (*top layer*) and those in V_b (*bottom vertices*) lie on ℓ_b (*bottom layer*). Such drawings occur as components in layered drawings of directed graphs [34] or between consecutive axes in hive plots [27], and also as final drawings, e.g., in tanglegram layouts for comparing phylogenetic trees [6, 7, 19, 33] or in layouts of networks highlighting the relationships between two communities [13, 29, 32].

The primary optimization goal for 2-layer graph drawings is to find permutations of one or both vertex sets V_t, V_b to minimize the number of edge crossings in the resulting layout. This problem is NP-complete [20], even if the permutation of one layer is given [16] or the degree is at most 4 [30], but both fixed-parameter algorithms [12, 26] and approximation algorithms [8] have been published. From a practical point of view, minimizing the number of crossings in 2-layer drawings may still result in visually complex drawings [24]. The existence of a planar 2-layer drawing can be tested in linear time [14, 21]. Layouts on two layers have been widely studied also in the area of graph drawing beyond planarity [2–5, 9–11].

In this paper, as an alternative approach to construct readable 2-layer drawings of bipartite graphs, we study vertex splitting [15, 17, 25, 28]. The *vertex-split* operation (or *split*, for simplicity) for a vertex v deletes v from G, adds two new copies v_1 and v_2 (in the original vertex subset of G), and distributes the edges originally incident to v among the two new vertices v_1 and v_2. Placing v_1 and v_2 independently in the 2-layer drawing can reduce the number of crossings. Vertex splitting has been studied in the context of the *splitting number*, which is the smallest number of vertex-splits needed to obtain a planar graph. The splitting number problem is NP-complete, even for cubic graphs [18], but the splitting numbers of complete and complete bipartite graphs are known [22, 23].

We study variations of the algorithmic problem of constructing planar 2-layer drawings with vertex splitting. In visualizing graphs defined on anatomical structures and cell types in the human body [1], the two vertex sets of G play different roles and vertex splitting is permitted only on one side of the layout. This motivates our interest in splitting only the bottom vertices. The top vertices may either be specified with a given context-dependent input ordering, e.g., alphabetically or according to some importance measure, or we may be allowed to arbitrarily permute them to perform fewer vertex splits.

Contributions. We prove that for a given integer k it is NP-complete to decide whether G admits a planar 2-layer drawing with an arbitrary permutation on the top layer and at most k vertex splits on the bottom layer. NP-completeness also holds if at most k vertices can be split, but each an arbitrary number of times. If, however, the vertex order of V_t is given, then we can compute a planar 2-layer drawing in linear time, both for minimizing the total number of splits and for minimizing the number of split vertices.

Preliminaries. We denote the order of the vertices in V_t and V_b in a 2-layer drawing by π_t and π_b, resp. If a vertex u precedes a vertex v, then we denote it by $u \prec v$. Although 2-layer drawings are defined geometrically, their crossings are fully described by π_t and π_b, as summarized in the following folklore lemma.

Lemma 1. *Let Γ be a 2-layer drawing of a bipartite graph $G = (V_t \cup V_b, E)$. Let (v_1, u_1) and (v_2, u_2) be two edges of E such that $v_1 \prec v_2$ in π_t. Then, edges (v_1, u_1) and (v_2, u_2) cross each other in Γ if and only if $u_2 \prec u_1$ in π_b.*

In the following we formally define the problems we study. For both of them, the input contains a bipartite graph $G = (V_t \cup V_b, E)$ and a split parameter k.

Crossing Removal with k Splits – CRS(k): Decide if there is a planar 2-layer drawing of G after applying at most k vertex-splits to the vertices in V_b.

Crossing Removal with k Split Vertices – CRSV(k): Decide if there is a planar 2-layer drawing of G after splitting at most k original vertices of V_b.

Note that in CRSV(k), once we decide to split an original vertex, then we can split its copies without incurring any additional cost. The example in Fig. 1 demonstrates the difference between the two problems. For both CRS(k) and

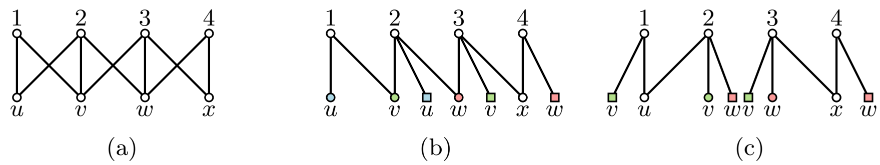


Fig. 1. (a) Instance G. (b) Optimal CRS solution with three splits, involving three different vertices. (c) Optimal CRSV solution with two split vertices.

CRSV(k), we refer to the variant where the order π_t of the vertices in V_t is given as part of the input by adding the suffix *"with Fixed Order"*

The following lemma implies conditions under which a vertex split must occur.

Lemma 2. *Let $G = (V_t \cup V_b, E)$ be a bipartite graph and let $u \in V_b$ be a bottom vertex adjacent to two top vertices $v_1, v_2 \in V_t$, with $v_1 \prec v_2$ in π_t. In any planar 2-layer drawing of G in which u is not split, we have that:*

C.1 A top vertex that appears between v_1 and v_2 in π_t can only be adjacent to u;
C.2 In π_b, u is the last neighbor of v_1 and the first neighbor of v_2.

Proof. If there is a top vertex v' between v_1 and v_2 adjacent to a bottom vertex $u' \neq u$, then (v', u') crosses (v_1, u) or (v_2, u). Also, if there is a neighbor u'' of v_1 after u in π_b, then the edges (v_1, u'') and (v_2, u) cross. A symmetric argument holds when there is a neighbor of v_2 before u in π_b. □

2 Crossing Removal with k Splits

In this section, we prove that the CRS(k) problem is NP-complete in general and linear-time solvable when the order π_t of the top vertices is part of the input.

Theorem 1. *The CRS(k) problem is* NP*-complete.*

Proof. The problem belongs to NP since, given a set of at most k splits for the vertices in V_b, we can check whether the resulting graph is planar 2-layer [14,21].

We use a reduction from the *Hamiltonian Path* problem to show the NP-hardness; see Fig. 2. Given an instance $G = (V, E)$ of the Hamiltonian Path problem, we denote by G' the bipartite graph obtained by subdividing every edge of G once. We construct an instance of the CRS(k) problem by setting the top vertex set V_t to consist of the original vertices of G, the bottom vertex set V_b to consist of the subdivision vertices of G', and the split parameter to $k = |E| - |V| + 1$. The reduction can be easily performed in linear time. We prove the equivalence.

Suppose that G has a Hamiltonian path $v_1, \ldots, v_n$. Set $\pi_t = v_1, \ldots, v_n$, and split all the vertices of V_b, except for the subdivision vertex of the edge (v_i, v_{i+1}), for each $i = 1, \ldots, n-1$. This results in $|V_b| - (n-1)$ splits, which is equal to k, since $|V_b| = |E|$ and $n = |V|$. We then construct π_b such that, for each $i = 1, \ldots, n-1$, all the neighbors of v_i appear before all the neighbors of v_{i+1},

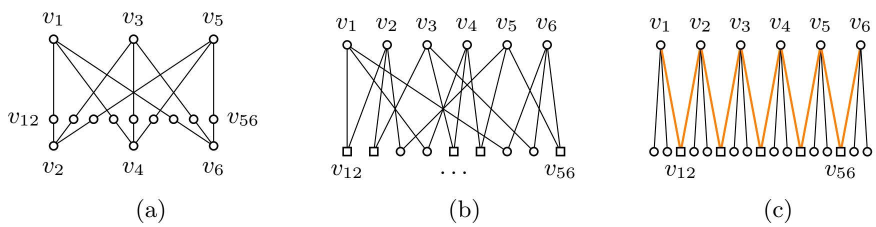


Fig. 2. (a) The subdivided graph G'. (b) The instance with V_b and V_t. (c) The splits are minimized if and only if G has a Hamiltonian path.

with their common neighbor being the last neighbor of v_i and the first of v_{i+1}. This guarantees that both conditions of Lemma 2 are satisfied for every vertex of V_b. Together with Lemma 1, this guarantees that the 2-layer drawing is planar.

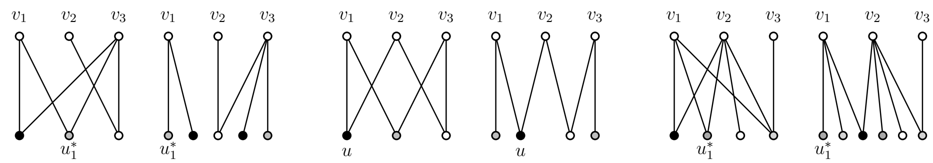
Suppose now that G' admits a planar 2-layer drawing with at most $|E|-|V|+1$ splits. Since $|E| = |V_b|$ and every vertex of V_b has degree exactly 2 (subdivision vertices), there exist at least $|V|-1$ vertices in V_b that are not split. Consider any such vertex $u \in V_b$. By C.1 of Lemma 2, the two neighbors of u are consecutive in π_t. Also, these vertices are connected in G by the edge whose subdivision vertex is u. Since this holds for each of the at least $|V| - 1$ non-split vertices, we have that each of the $|V| - 1$ distinct pairs of consecutive vertices in V_t (recall that $V_t = V$) is connected by an edge in G. Thus, G has a Hamiltonian path. □

Next, we focus on the optimization version of the CRS(k) with Fixed Order problem. Our recursive algorithm considers a constrained version of this problem, in which the first neighbor in π_b of the first vertex in π_t may be prescribed. At the outset of the recursion, there exists no prescribed first neighbor. The algorithm returns the split vertices in V_b and the corresponding order π_b.

In the base case, there is only one top vertex v, i.e., $|V_t| = 1$. Since all vertices in V_b have degree 1, no split takes place. We set π_b to be any order of the vertices in V_b where the first vertex is the prescribed first neighbor of v, if any.

In the recursive case when $|V_t| > 1$, we label the vertices in V_t as $v_1, \ldots, v_{|V_t|}$, according to π_t. If the first neighbor of v_1 is prescribed, we denote it by u_1^*. Also, we denote by N^1 the set of degree-1 neighbors of v_1, and by N^+ the other neighbors of v_1. Note that only the vertices in N^+ are candidate to be split for v_1. In particular, by C.1 of Lemma 2, a vertex in N^+ can avoid being split in this step only if it is also incident to v_2. Further, since any vertex in N^+ that is not split must be the last neighbor of v_1 in π_b, by C.2 of Lemma 2, at most one of the common neighbors of v_1 and v_2 will not be split. For the same reason, if vertex u_1^* is prescribed, then it must be split, unless v_1 has degree 1.

In view of these properties, we distinguish three cases based on the common neighborhood of v_1 and v_2. In all cases, we will recursively compute a solution for the instance composed of the graph $G' = (V'_t \cup V'_b, E')$ obtained by removing v_1 and the vertices in N^1 from G, and of the order $\pi'_t = v_2, \ldots, v_{|V_t|}$. We denote by π'_b and s' the computed order and the number of splits for the vertices in V'_b. In



(a) (b) (c)

Fig. 3. Algorithm for CRS(k) with Fixed Order optimization. Vertices in N^+ are colored in shades of grey. (a) Case 1, (b) Case 2, and (c) Case 3.

the following we specify for each case whether the first neighbor of v_2 in the new instance is prescribed or not, and how to incorporate the neighbors of v_1 into π'_b.

Case 1: v_1 and v_2 have no common neighbor; see Figure 3a. In this case, we do not prescribe the first neighbor of v_2 in the instance composed of G' and π'_t. To compute a solution for the original instance, we split each vertex in N^+ so that one copy becomes incident only to v_1. We construct π_b by selecting the prescribed vertex u_1^*, if any, followed by the remaining neighbors of v_1 in any order and, finally, by appending π'_b. This results in $s = |N^+| + s'$ splits.

Case 2: v_1 and v_2 have exactly one common neighbor u. If $u = u_1^*$ and v_1 has degree larger than 1, then u cannot be the last neighbor of v_1 and must be split. Thus, we perform the same procedure as in Case 1. Otherwise, in the instance composed of G' and π'_t, we set u as the prescribed first neighbor of v_2; see Figure 3b. To compute a solution for the original instance, we split each vertex in N^+, except u, so that one copy becomes incident only to v_1. We construct π_b by selecting the prescribed vertex u_1^*, if any, followed by the remaining neighbors of v_1 different from u in any order and, finally, by appending π'_b. This results in $s = |N^+| - 1 + s'$ splits.

Case 3: v_1 and v_2 have more than one common neighbor. If v_1 and v_2 have exactly two common neighbors u, u' and one of them is u_1^*, say $u = u_1^*$, then u cannot be the last neighbor of v_1, as v_1 has degree larger than 1. Thus, we proceed exactly as in Case 2, using u' as the only common neighbor of v_1 and v_2.

Otherwise, there are at least two neighbors of v_1 different from u_1^*; see Figure 3c. We want to choose one of these vertices as the last neighbor of v_1, so that it is not split. However, the choice is not arbitrary as this may affect the possibility for v_2 to save the split for a neighbor it shares with v_3. In the instance composed of G' and π'_t, we do not prescribe the first vertex of v_2. To compute a solution for the original instance, we simply choose as the last neighbor of v_1 any of its common neighbors with v_2 that has not been set as the last neighbor of v_2 in π'_b. Such a vertex, say u, always exists since v_1 and v_2 have at least two common neighbors different from u_1^*, and can be moved to become the first vertex in π'_b. Specifically, we split all the vertices in N^+, except for u, so that one copy becomes incident only to v_1. We construct π_b by selecting the prescribed vertex u_1^*, if any, followed by the remaining neighbors of v_1 different from u in any order. We then modify π'_b by moving u to be the first vertex. Note that this operation does not affect planarity, as it only involves reordering the set of consecutive degree-1 vertices incident to v_2. Finally, we append the modified π'_b. This results in $s = |N^+| - 1 + s'$ splits.

Theorem 2. *For a bipartite graph $G = (V_t \cup V_b, E)$ and an order π_t of V_t, the optimization version of CRS(k) with Fixed Order can be solved in $O(|E|)$ time.*

Proof. By construction, for each $i = 1, \dots, |V_t| - 1$, all neighbors of v_i precede all neighbors of v_{i+1} in π_b. Thus, by Lemma 1, the drawing is planar. The minimality of the number of splits follows from Lemma 2, as discussed before the case distinction. In particular, any minimum-splits solution can be shown to be equivalent to the one produced by our algorithm. The time complexity follows as each vertex only needs to check its neighbors a constant number of times. □

3 Crossing Removal with k Split Vertices

In this section, we prove that the CRSV(k) problem is NP-complete in general and linear-time solvable when the order π_t of the top vertices is part of the input.

To prove the NP-completeness we can use the reduction of Theorem 1. In fact, in the graphs produced by that reduction all vertices in V_b have degree 2. Hence, the number of vertices that are split coincides with the total number of splits.

Theorem 3. *The CRSV(k) problem is* NP*-complete.*

For the version with Fixed Order, we first apply C.1 of Lemma 2 to identify vertices that need to be split at least once, and repeatedly split them until each has degree 1. For a vertex $u \in V_b$, we can decide if it needs to be split by checking whether its neighbors are consecutive in π_t and, if u has degree at least 3, whether all its neighbors have degree exactly 1, except possibly for the first and the last.

We first perform all necessary splits. For each $i = 1, \dots, |V_t| - 1$, consider the two consecutive top vertices v_i and v_{i+1}. If they have no common neighbor, no split is needed. If they have exactly one common neighbor u, then we set u as the last neighbor of v_i and the first of v_{i+1}, which allows us not to split u, according to C.2. Since u did not participate in any necessary split, if u is also adjacent to other vertices, then all its neighbors have degree 1, except possibly the first and last. Hence, C.2 can be guaranteed for all pairs of consecutive neighbors of u.

Otherwise, v_i and v_{i+1} have at least two common neighbors and therefore have degree at least 2. Then each of their common neighbors has degree exactly 2, as otherwise it would have been split by C.1. Hence, all common neighbors (except for at most one) of v_i and v_{i+1} have to be split. Since all these vertices are incident only to v_i and v_{i+1}, we can choose any of them to be not split by setting it as the last neighbor of v_i and as the first of v_{i+1}.

At the end we construct the order π_b so that, for each $i = 1, \dots, |V_t|-1$, all the neighbors of v_i precede all the neighbors of v_{i+1}, and the unique common neighbor of v_i and v_{i+1}, if any, is the last neighbor of v_i and the first of v_{i+1}. By Lemma 1, this guarantees planarity. Identifying and performing all unavoidable splits and computing π_b can be easily done in $O(|E|)$ time. Since we only performed unavoidable splits, as dictated by Lemma 2, we have the following.

Theorem 4. *For a bipartite graph $G = (V_t \cup V_b, E)$ and an order π_t of V_t, the optimization version of CRSV(k) with Fixed Order can be solved in $O(|E|)$ time.*

4 Open Problems

Minimizing the total number of splits, or the number of split vertices are natural problems. Other variants include minimizing the maximum number of splits per vertex and considering the case where splits are allowed in both layers. Vertex splits can also be used to improve other quality measures of a 2-layer layout (besides crossings). When visualizing large bipartite graphs, a natural goal is to arrange the vertices so that a small window can capture all the neighbors of a given node, i.e., minimize the maximum distance between the first and last neighbors of a top vertex in the order of the bottom vertices. We define this problem in the appendix, where we also show NP-completeness for one variant.

Acknowledgments. This work started at Dagstuhl Seminar 21152 "Multi-Level Graph Representation for Big Data Arising in Science Mapping". We thank the organizers and participants for the discussions, particularly M. Bekos, M. Kaufmann, and C. Raftopoulou.

References

1. https://hubmapconsortium.github.io/ccf-asct-reporter/
2. Angelini, P., Da Lozzo, G., Förster, H., Schneck, T.: 2-layer k-planar graphs: Density, crossing lemma, relationships, and pathwidth. In: GD '20. vol. 12590, pp. 403–419. Springer (2020)
3. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M.: 2-level quasi-planarity or how caterpillars climb (SPQR-)trees. In: SODA. pp. 2779–2798. SIAM (2021)
4. Biedl, T.C., Chaplick, S., Kaufmann, M., Montecchiani, F., Nöllenburg, M., Raftopoulou, C.N.: Layered fan-planar graph drawings. In: MFCS. vol. 170, pp. 14:1–14:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
5. Binucci, C., Chimani, M., Didimo, W., Gronemann, M., Klein, K., Kratochvíl, J., Montecchiani, F., Tollis, I.G.: Algorithms and characterizations for 2-layer fan-planarity: From caterpillar to stegosaurus. JGAA **21**(1), 81–102 (2017)
6. Buchin, K., Buchin, M., Byrka, J., Nöllenburg, M., Okamoto, Y., Silveira, R., Wolff, A.: Drawing (complete) binary tanglegrams. Algorithmica **62**(1–2), 309–332 (2012)
7. Czabarka, É., Székely, L.A., Wagner, S.G.: A tanglegram Kuratowski theorem. JGT **90**(2), 111–122 (2019)
8. Demetrescu, C., Finocchi, I.: Removing cycles for minimizing crossings. JEA **6**, 2 (2001)
9. Di Giacomo, E., Didimo, W., Eades, P., Liotta, G.: 2-layer right angle crossing drawings. Algorithmica **68**(4), 954–997 (2014)
10. Didimo, W.: Density of straight-line 1-planar graph drawings. IPL **113**(7), 236–240 (2013)
11. Didimo, W., Liotta, G., Montecchiani, F.: A survey on graph drawing beyond planarity. ACM Computing Surveys **52**(1), 4:1–4:37 (2019)

12. Dujmovic, V., Whitesides, S.: An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. Algorithmica **40**(1), 15–31 (2004)
13. Dumas, M., McGuffin, M.J., Robert, J.M., Willig, M.C.: Optimizing a radial layout of bipartite graphs for a tool visualizing security alerts. In: GD '12. vol. 7034, pp. 203–214. Springer-Verlag (2012)
14. Eades, P., McKay, B., Wormald, N.: On an edge crossing problem. In: ACSC '86. pp. 327–334 (1986)
15. Eades, P., de Mendonça N., C.F.X.: Vertex-splitting and tension-free layouts. In: GD '95. vol. 1027, pp. 202–211. Springer (1996)
16. Eades, P., Wormald, N.C.: Edge crossings in drawings of bipartite graphs. Algorithmica **11**(4), 379–403 (1994)
17. Eppstein, D., Kindermann, P., Kobourov, S., Liotta, G., Lubiw, A., Maignan, A., Mondal, D., Vosoughpour, H., Whitesides, S., Wismath, S.: On the planar split thickness of graphs. Algorithmica **80**, 977–994 (2018)
18. Faria, L., de Figueiredo, C.M.H., Mendonça, C.F.X.: Splitting number is NP-complete. DAM **108**(1), 65–83 (2001)
19. Fernau, H., Kaufmann, M., Poths, M.: Comparing trees via crossing minimization. JCSS **76**(7), 593–608 (2010)
20. Garey, M.R., Johnson, D.S.: Crossing number is NP-complete. SIADM **4**(3), 312–316 (1983)
21. Harary, F., Schwenk, A.: A new crossing number for bipartite graphs. Utilitas Mathematica **1**, 203–209 (1972)
22. Hartsfield, N., Jackson, B., Ringel, G.: The splitting number of the complete graph. Graphs and Combinatorics **1**(1), 311–329 (1985)
23. Jackson, B., Ringel, G.: The splitting number of complete bipartite graphs. Archiv der Mathematik **42**(2), 178–184 (1984)
24. Jünger, M., Mutzel, P.: 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. JGAA **1**(1), 1–25 (1997)
25. Knauer, K., Ueckerdt, T.: Three ways to cover a graph. DM **339**(2), 745–758 (2016)
26. Kobayashi, Y., Tamaki, H.: A fast and simple subexponential fixed parameter algorithm for one-sided crossing minimization. Algorithmica **72**(3), 778–790 (2015)
27. Krzywinski, M., Birol, I., Jones, S.J., Marra, M.A.: Hive plots—rational approach to visualizing networks. Briefings in Bioinformatics **13**(5), 627–644 (2012)
28. Liebers, A.: Planarizing graphs – a survey and annotated bibliography. JGAA **5**(1), 1–74 (2001)
29. Meulemans, W., Schulz, A.: A tale of two communities: Assessing homophily in node-link diagrams. In: GD '15. vol. 9411, pp. 489–501. Springer (2015)
30. Muñoz, X., Unger, W., Vrto, I.: One sided crossing minimization is NP-hard for sparse graphs. In: GD '01. vol. 2265, pp. 115–123. Springer (2001)
31. Papadimitriou, C.H.: The NP-completeness of the bandwidth minimization problem. Computing **16**(3), 263–270 (1976)
32. Pezzotti, N., Fekete, J.D., Höllt, T., Lelieveldt, B.P.F., Eisemann, E., Vilanova, A.: Multiscale visualization and exploration of large bipartite graphs. CGF **37**(3), 549–560 (2018)
33. Scornavacca, C., Zickmann, F., Huson, D.H.: Tanglegrams for rooted phylogenetic trees and networks. Bioinformatics **27**(13), i248–i256 (2011)
34. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. TSMC **11**(2), 109–125 (1981)

Appendix

A Maximum Span h with k Splits

In this appendix, we consider a variant of 2-layer drawing where, instead of minimizing the number of crossings, we want to minimize the maximum *span* of the vertex set V_t, i.e., the distance between the first and the last neighbor of a top vertex in the order of the bottom vertices.

This problem is motivated from the visualization of anatomical structures and cell types, in which crossings are allowed but we aim for compact neighborhood ranges. This is specifically important for very large bipartite graphs, as it allows to capture all the neighborhood of the same vertex in a small window, which can then be fully displayed even on a screen with limited size.

The problem is formally defined as follows. The input consists of a bipartite graph $G = (V_t \cup V_b, E)$, a split parameter k, and a span parameter h.

Minimizing Maximum Span h with k Splits – MMSS(h, k): Decide if there is a 2-layer drawing of G where the maximum span of the vertices in V_t is at most h after applying at most k vertex splits to the vertices in V_b.

We observe that variations of the MMSS(h, k) problem can be formulated as for the CRS(k) problem, by limiting to k the number of split vertices (or splits per vertex), rather than the total number of splits. On the other hand, the variation "with Fixed Order" may not be relevant, as the order of the top vertices does not play a specific role in the given definition.

To demonstrate the difficulty underlying the problem we propose to study, we prove that the version in which $k = 0$ (that is, no split is allowed and the goal is to minimize the span by permuting the vertices in V_b) is NP-complete. Note that this result does not immediately imply the NP-completeness of the problem when some splits are permitted. In fact, if the number of allowed splits is large enough to reduce each bottom vertex to degree 1, then it is immediate to construct an ordering of the bottom vertices minimizing the span.

To prove the NP-hardness, we use a reduction from the NP-complete *graph bandwidth* problem [31]. Recall that in the bandwidth problem, given an input graph $G = (V, E)$ and an integer h, the goal is to label each vertex $u \in V$ with a distinct integer $f(u)$ such that $\max\{|f(u_i) - f(u_j)| : (u_i, u_j) \in E\} \leq h$.

Theorem 5. *The MMSS(h, 0) problem is* NP*-complete.*

Proof. The membership in NP can be proved by observing that the span of each top vertex can be easily computed for a given order of the bottom vertices.

For the NP-hardness, we reduce from the bandwidth problem. Let $G = (V, E)$ be an instance of the bandwidth problem. Let G' be the graph obtained from G by subdividing each edge once. To create the MMSS(h, 0) instance, we add all the subdivision vertices to V_t, and all the original vertices of G to V_b[8]. The reduction can be performed in linear time.

To prove the equivalence, we exploit the following observation: For each edge $(u_1, u_2) \in E$ in the bandwidth instance, there exists a vertex v in V_t in the MMSS(h, 0) instance having $u_1, u_2 \in V_b$ as its only two neighbors, and vice versa.

Suppose that the MMSS(h, 0) instance admits an ordering π_b of the vertices in V_b so that the maximum span of the top vertices is $c_m \leq h$. To compute a solution of the bandwidth instance $G = (V, E)$, we assign to each vertex $u \in V$ a label $f(u)$ that is equal to the position of u in π_b. By the above observation, for each edge $(u_i, u_j) \in E$, we have that $|f(u_i) - f(u_j)|$ is equal to the span of the vertex in V_t that is the subdivision vertex of (u_i, u_j), and thus $|f(u_i) - f(u_j)| \leq c_m \leq h$.

Analogously, given a labeling $f(u)$ of each vertex $u \in V$ that results in a bandwidth $c_b \leq h$ for G, we can construct the order π_b of the vertices in V_b according to this labeling. Exploiting the same observation, we can prove that this results in a span smaller or equal to $c_b \leq h$ for each top vertex. The reduction is illustrated in Fig. 4 with a small graph. □

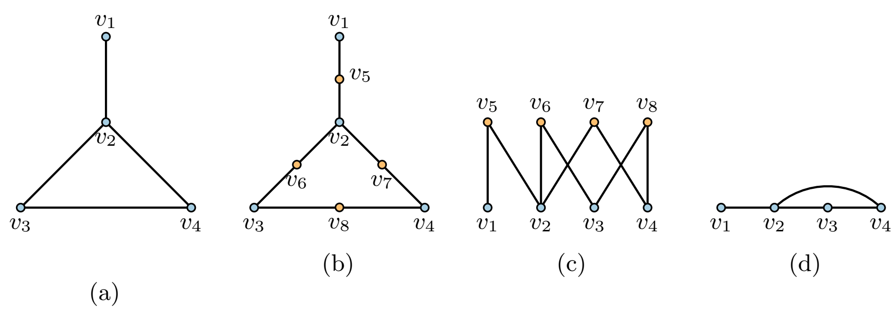


Fig. 4. Reduction from a bandwidth instance to a MMSS(0) instance.

[8] Observe that this is the opposite of the reduction in Theorem 1, where the subdivision vertices are in V_t and the original in V_b. This makes a significant difference, given the different role of the two layers.