An Algorithm for the Maximum Weight Independent Set Problem on Outerstring Graphs*

J. Mark Keil† Joseph S. B. Mitchell‡ D. Pradhan§ Martin Vatshelle¶

Abstract

Outerstring graphs are the intersection graphs of curves that lie inside a disk such that each curve intersects the boundary of the disk. Outerstring graphs are among the most general classes of intersection graphs studied. To date, no polynomial time algorithm is known for any of the classical graph optimization problems on outerstring graphs; in fact, most are NP-hard. It is known that there is an intersection model for any outerstring graph that consists of polygonal arcs attached to a circle. However, this representation may require an exponential number of segments relative to the size of the graph.

Given an outerstring graph and an intersection model consisting of polygonal arcs with a total of N segments, we develop an algorithm that solves the Maximum Weight Independent Set problem in $O(N^3)$ time. If the polygonal arcs are restricted to single segments, then outersegment graphs result. For outersegment graphs, we solve the Maximum Weight Independent Set problem in $O(n^3)$ time where n is the number of vertices in the graph.

1 Introduction

A graph G is a *geometric intersection graph* if the vertex set of G is a set of geometric objects and two such objects are adjacent in G if and only if they intersect. An *independent set* in a geometric intersection graph corresponds to a set of disjoint geometric objects in the intersection model. The Maximum (Weight) Independent Set problem in intersection graphs of geometric objects in the plane has many applications, including train dispatching [8], map labelling [1], and data mining [13]. In the railroad dispatching problem studied by Filer, Mihalák, Schöbel, Widmayer and Zych [8], we are given a set of paths (strings) in the plane and asked for a maximum set of non-conflicting train routes, i.e. a maximum independent set in a string graph. Due to the fact that most trains either leave or enter the station area, it is natural to consider outerstring graphs in this application. This is the problem we solve in this paper.

String graphs are the intersection graphs of curves in the plane and they are among the most general geometric intersection graphs that have been studied. String graphs are a superclass of planar graphs [25], chordal graphs, co-comparability graphs [15], subtree filament graphs [14] and circle graphs. Indeed the intersection graph of any collection of connected sets in the plane is a string graph. As early as 1959, Benzer [4] encountered string graphs in his study of genetic structures. Since then they have been extensively studied and have many applications. Kratochvíl et al. [19] showed that every string graph can be realized by a family of polygonal arcs with a finite number of intersections. However in 1991, Kratochvíl and Matoušek [21] constructed string graphs on n vertices that require at least 2^{cn} intersection points in any realization. This also implies that a representation of a string graph with a family of polygonal arcs may require an exponential number of bends in the polygonal arcs. In 1991, Kratochvíl [18] proved that the problem of recognizing string graphs is NP-hard, but more than a decade passed before Schaefer et al. [24] showed that recognizing string graphs is in NP.

In 1966 Sinden [25] showed that all planar graphs are string graphs, thus the Maximum Independent Set problem became known to be NP-hard on string graphs when it was proven to be NP-hard in planar graphs. Recently, Fox and Pach [10] provided approximation algorithms and exact sub-exponential algorithms for the Maximum Independent Set problem in string graphs. In 1976, the 3-Colorability problem for string graphs was proven NP-complete by Ehrlich et al. [7], even when a geometric representation is given as the input. The Maximum Clique problem has long been known to be NP-hard [22, 23] on string graphs. Indeed most of the classical NP-hard graph optimization problems remain NP-hard when restricted to string graphs, even when given a geometric representation.

It seems that one must somehow restrict string graphs to achieve polynomial time algorithms. The two most natural ways to restrict string graph are to either limit the shapes of the strings, or to limit the positions of the strings. The most commonly studied such restrictions are to limit the strings to be straight line segments or

*J. M. Keil supported by NSERC; J. Mitchell is partially supported by NSF (CCF-1018388) and the US-Israel Binational Science Foundation (Grant 2010074).

†Dept. of Computer Science, University of Saskatchewan
‡Dept. of Appl. Math. and Statistics, Stony Brook University
§Dept. of Applied Math, Indian School of Mines, Dhanbad
¶Dept. of Informatics, University of Bergen

to require that each string touches the infinite face of the plane. We first consider each restriction separately and then the case combining them.

Segment graphs are the intersection graphs of line segments in the plane. This restriction to line segments still allows the graphs to be useful in many applications, but unfortunately most of the classical NP-complete graph problems remain intractable on segment graphs. Since all planar graphs are segment graphs [6], the 3-Colorability problem and the Maximum Independent Set problem remain NP-complete on segment graphs. Kratochvíl and Nešetřil [22] proved that the Maximum Independent Set problem in segment graphs is NP-hard even if all the segments are restricted to lie in at most two directions in the plane. It has recently been shown that the Maximum Clique problem is NP-hard on segment graphs [5]. Thus even a severe limiting of the shapes of the strings in a string graph does not lead to polynomial time algorithms.

The restriction that each string touches the infinite face of the plane was explored in 1991 [17] by Kratochvíl who defined *outerstring graphs* to be the intersection graphs of curves that lie inside a disk such that each curve intersects the boundary of the disk in one of its endpoints. Although outerstring graphs have been studied for more than 20 years [11, 12, 17, 20], when we consider the classical NP-hard graph optimization problems on outerstring graphs, we again do not find any known polynomial time algorithms. For outerstring graphs the NP-completeness of Minimum Clique Cover, Colorability, Minimum Dominating Set, and Hamiltonian Cycle follow from the fact that they contain circle graphs. The Maximum Clique problem was recently shown to be NP-hard on ray graphs [5], a subclass of outerstring graphs. The Maximum Independent Set problem remains open on outerstring graphs.

In their study of train dispatching, Flier et al. [9] consider subclasses of outerstring graphs, in particular the intersection graphs of segments lying inside a disk having one endpoint attached to the boundary of the disk, called *outersegment graphs*. Applying the additional restriction that each segment is either horizontally or vertically aligned, they are able to obtain a polynomial time algorithm for the Maximum Independent Set problem given a geometric representation of the graph.

In the next section, we describe a dynamic programming algorithm for the Maximum Weight Independent Set problem in an outerstring graph which runs in time polynomial in the size of the geometric input representation of the graph. Finally, we show how our algorithm can be used to find a maximum weight set of disjoint boundary rectangles in $O(n^3)$ time. This problem has applications in PCB routing [16].

2 Outerstring graphs

Outerstring graphs are the intersection graphs of curves in the plane that lie inside a circle such that each curve intersects the boundary of the circle in one of its endpoints. Let $G = (V, E)$ be an outerstring graph with n weighted vertices. In order to find the Maximum Weight Independent Set of G, our algorithm assumes the input is a polygonal geometric representation of G. The circle is represented as a simple polygon P with $O(n)$ vertices. Lying completely inside P, each string s corresponding to a vertex of G is represented by a non-self-intersecting polygonal line with one endpoint, $start(s)$, coinciding with a unique vertex of P. We call the vertices of s that are different from $start(s)$ the interior vertices of s, as they lie in the interior of P. Let S be the set of polygonal lines corresponding to the vertices of V. Then $R(G) = (P, S)$ is a representation of G. See the top left of Figure 1. Let N be the total number of segments used to represent the strings in S, and the polygon P. In many applications of outerstring graphs this polygonal geometric representation is the natural input.

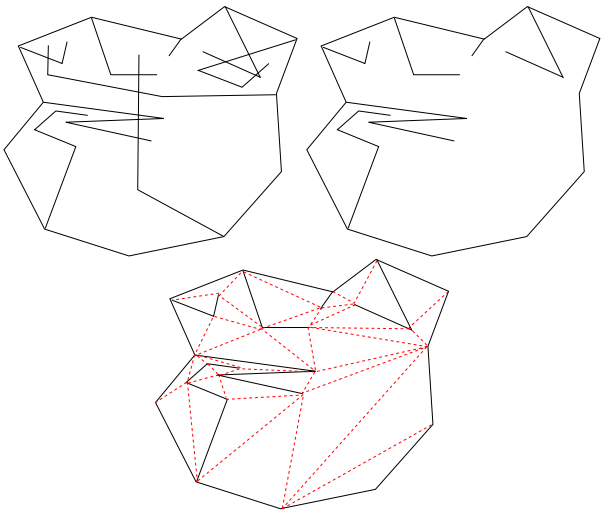

Figure 1: The representation of an outerstring graph, a maximum set S^* of disjoint strings, and a triangulation of P respecting S^*.

An optimal solution to the Maximum Weight Independent Set problem for an outerstring graph G with geometric representation $R(G) = (P, S)$ appears as a set of disjoint polygonal lines $S^* \subseteq S$ inside P. See the top right of Figure 1. Together, $P \cup S^*$ form a connected planar straight line graph that can be triangulated. The bottom drawing of Figure 1 show the triangles that are inside P in a triangulation of $P \cup S^*$. Let u and v be vertices of either P or of the polygonal lines of S. In order to define the subproblems used in

our dynamic program, we will use u and v to define a subpolygon $P(u, v)$ of P. The subproblem associated with $P(u, v)$ is that of finding a maximum weight independent set of strings of S that lie wholly inside $P(u, v)$, where $P(u, v)$ is treated as a closed set. It is sufficient to consider all u and v such that uv forms an edge in a triangulation of the planar straight line graph $P \cup S^*$, where $S^* \in S$ is the set of strings in an optimal solution. We do not know S^*, but if u is an interior vertex of string S_u and v is an interior vertex of string S_v, then if uv is to be a diagonal in a triangulation of $P \cup S^*$ strings S_u and S_v must be in the optimal solution, and thus be disjoint. Thus if u or v are interior vertices of strings of S, we also insist on including those strings in the solution to the subproblem for $P(u, v)$ whether or not they lie completely in $P(u, v)$. Also segment uv cannot intersect any segment in S_u or S_v.

If u and v are both vertices of P, we define $P(u, v)$ to be the part of P to the left of the directed edge $\vec{uv}$. See the top drawing in Figure 2. If v is an interior vertex of a string A in S, and u lies on polygon P such that string A does not intersect segment uv then we define $P(u, v)$ to be the polygonal region bounded by the portion of P clockwise from u to $start(A)$, the portion of string A from $start(A)$ to v, and the edge uv. See the top drawing in Figure 3 for an example. If u is an interior vertex of a string and v lies on P then $P(u, v)$ is defined analogously. If both u and v are interior vertices of distinct strings of P, $P(u, v)$ is defined to be the polygonal region bounded by the portion of P clockwise from $start(S_u)$ to $start(S_v)$, the portion of S_v from $start(S_v)$ to v, segment uv, and the portion of S_u from u to $start(S_u)$. See for example the top drawing in Figure 4. Let $f(u, v)$ be the optimal value of a solution to the subproblem associated with $P(u, v)$. The dynamic programming algorithm considers the subpolygons $P(u, v)$ in increasing order of area, and for each computes the optimal weight $f(u, v)$ for a solution to the subproblem.

The type of a subproblem depends on whether u or v or both are interior vertices of strings of S.

Type 0: In a type 0 subproblem neither u nor v is an interior vertex of a string of S; thus, both are vertices of P. See Figure 2.

In a triangulation of $P(u, v)$ where $P(u, v)$ contains the strings of an optimal solution, the third vertex w of the triangle containing u and v may be a vertex of P. In this case the optimal solution is the disjoint union of the solution to the subproblem associated with $P(u, w)$ and the solution to the subproblem associated with $P(w, v)$.

If w is an interior vertex of a string A then the solution to $P(u, v)$ will depend upon whether or not string A contains u or v as $start(A)$. If neither u nor v is $start(A)$, then the subproblem for $P(u, w)$ is of type 1 and the solution contains string A. Likewise the solution to the subproblem for $P(w, v)$ is of type 1 and contains string A. The solution to the subproblem for $P(u, v)$ is the union of the solution to the subproblem for $P(u, w)$ and the solution to subproblem for $P(w, v)$. In the union, the string A is only included once; thus, $f(u, v) = f(u, w) + f(w, v) - weight(A)$. If $start(A)$ is u, then the region cut off by $u\vec{w}$ is a "pocket" of string A that cannot contain any other string. See the fourth situation of Figure 2. The only subproblem that exists is that for subpolygon $P(w, v)$; thus, $f(u, v) = f(w, v)$.

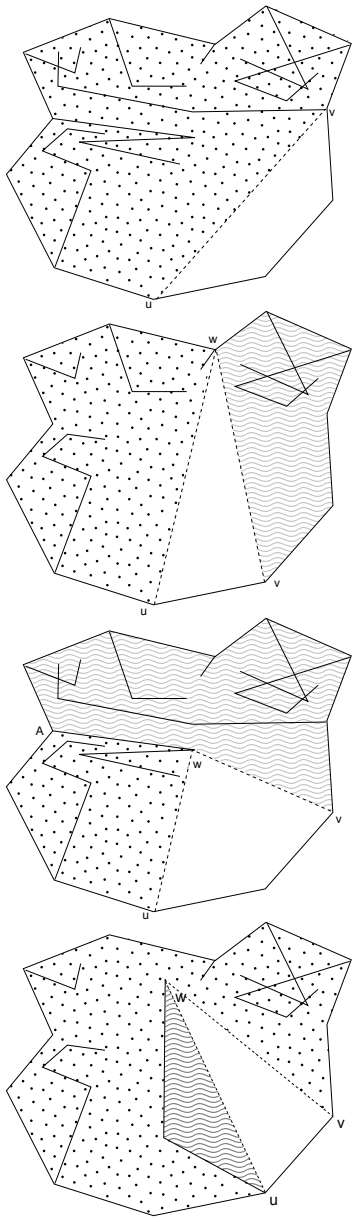

Figure 2: In a type 0 subproblem, the third vertex w of triangle uwv may lie on P or on a string A of S.

Type 1: In this case, exactly one of u and v lie on P. Without loss of generality, assume that u lies on P and v is an interior vertex of a string A. Hence, the subproblem for $P(u, v)$ includes string A plus the solution to the MAXIMUM WEIGHT INDEPENDENT SET problem for strings that lie inside P, left of $\vec{uv}$ that avoid string A. See Figure 3. The third vertex w of the triangle that contains u and v in a triangulation of $P(u, v)$ may lie on P, A or another string B. If w lies on P then we have reduced the problem to a smaller type 0 subproblem associated with $P(u, w)$ and a smaller type 1 subproblem associated with $P(w, v)$. If w lies on A then we have reduced the problem to a smaller type 1 subproblem associated with $P(u, w)$. Note that the region cut off by $\vec{wv}$ is a "pocket" of string A that cannot contain any other string. See the third situation in Figure 3. If w lies on another string B, then the problem reduces to a subproblem of type 1 associated with $P(u, w)$, plus a subproblem of type 2 associated with $P(w, v)$, as in the final drawing of Figure 3.

Type 2: In a type 2 subproblem u is an interior vertex of a string A and v is an interior vertex of a string B. The third vertex w of a triangle containing u and v may lie on P, one of A or B, or on another string C. See Figure 4. If w lies on P then the solution to the subproblem for $P(u, v)$ is the union of the solutions to two type 1 subproblems, for $P(u, w)$ and for $P(w, v)$. If w lies on B, the same string as v, then the region cut off by segment wv cannot contain any other strings and the solution to for $P(u, v)$ will consist of the solution to the smaller subproblem for $P(u, w)$. See the third drawing of Figure 4. The situation where w lies on A is analogous. If w lies on a new string C, then the solution to the subproblem for $P(u, v)$ is the union of the two type 2 subproblems associated with $P(u, w)$ and $P(w, v)$, as in the final drawing of Figure 4.

Theorem 1 *Given the geometric representation $R(G) = (P, S)$ of a weighted outerstring graph G, the dynamic programming algorithm described above computes the maximum weight of an independent set for G in $O(N^3)$ time, where N is the number of segments used to represent the strings of S and the polygon P.*

Proof. The correctness of the computation of each $f(u, v)$ can be verified by induction. If $P(u, v)$ is a triangle then $f(u, v)$ is either zero or equal to the weight of the input string(s) containing u and/or v. Otherwise $f(u, v)$ can be computed in constant time from $f(u, w)$ and $f(w, v)$, for one of the $O(N)$ possible w in $P(u, v)$, where $P(u, w)$ and $P(w, v)$ are smaller area subpolygons such that the triangle uwv only intersects the strings in the optimal solution of the subproblem associated with $P(u, v)$ in vertices of the representation.

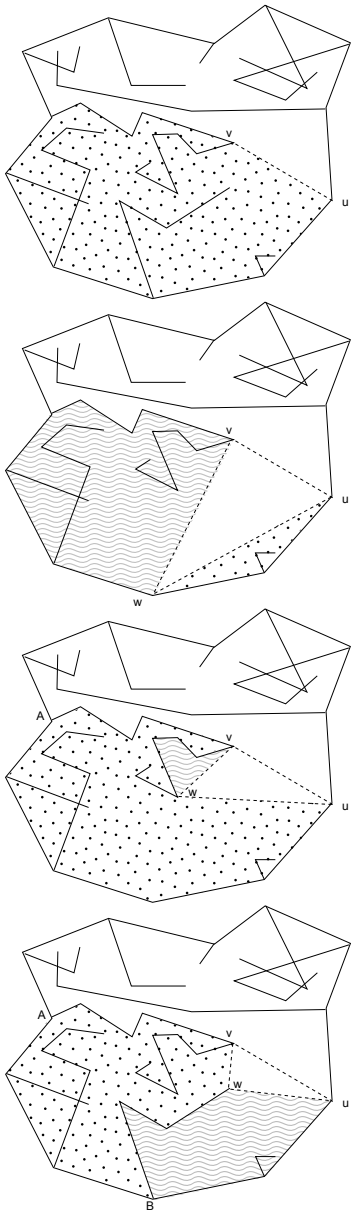

Figure 3: In a type 1 subproblem, the third vertex w of triangle uwv may lie on P, on the same string as v, or on a different string.

The $O(N^3)$ running time can be achieved by precomputing intersection information. There are $O(N^2)$ segments uv that may potentially define subpolygons $P(u, v)$. Each of these segments can be tested against the N segments in the representation to determine the strings of S that the segment intersects. In order for $P(u, v)$ to be a subpolygon, segment uv must not intersect with any segment in the string S_u containing u nor in the string S_v containing v. Further, there can be no intersection between strings S_u and S_v. For all S_i and S_j, it can be precomputed in $O(N^2)$ time whether

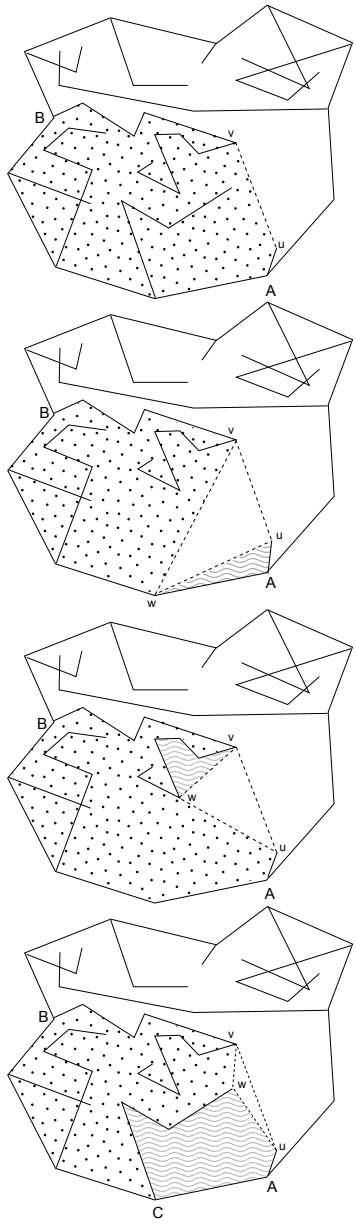

Figure 4: A type 2 subproblem

or not the strings S_i and S_j intersect, by first precomputing all the pairwise intersections of segments in the representation $R(G)$.

Given a subpolygon $P(u, v)$, where u and v are interior vertices of strings A and B respectively, a vertex w is potentially the third vertex of a triangle uwv in a triangulation of $P(u, v)$ containing the strings corresponding to an optimal solution for $P(u, v)$ if uw and wv do not intersect the strings A and B. These intersections have been precomputed and stored. It is also necessary to ensure that w is on the left side of $\vec{uv}$. □

If the strings in $R(G) = (P, S)$ are each single segments then G is an outersegment graph. The geometric representation of an outersegment graph with n vertices requires only n segments to represent S and P; thus:

Corollary 1 *Given the geometric representation of a weighted outersegment graph with n vertices, the dynamic programming algorithm computes the maximum weight of an independent set in $O(n^3)$ time.*

3 Application

We use our Maximum Weight Independent Set algorithm for outerstring graphs to find a maximum weight disjoint set of boundary rectangles. Given a rectangular region R, a rectangle r contained in R is a *boundary rectangle* with respect to R if at least one of the sides of r is a subset of a side of R. The problem of finding a maximum weight disjoint set of boundary rectangles has application in printed circuit board routing [16]. Kong et al. [16] provide the first polynomial time algorithm for the problem running in $O(n^6)$ time. This was improved to $O(n^4)$ in [3] and [2]. Using our Maximum Weight Independent Set algorithm for outerstring graphs, we can achieve $O(n^3)$ time.

Given a rectangle R and a set Q of n weighted boundary rectangles inside R, we create an instance of the Maximum Weight Independent Set for an outerstring graph G with geometric representation $R(G) = (P, S)$, where P is the boundary of R and each boundary rectangle r in Q maps to a four segment polygonal chain in S. Let s_{min} be the minimum side length of a rectangle in Q, and let δ be $\frac{s_{min}}{2}$.

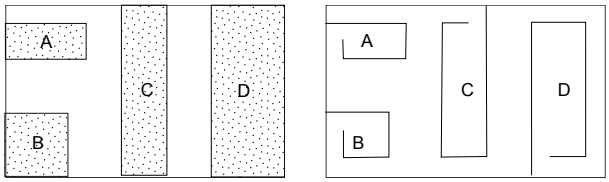

Figure 5: The boundary rectangles in Q are represented by strings in S

If a rectangle $r \in Q$ has one side contained in a side of R, then the corresponding string s has $start(s)$ at the clockwisemost intersection point of the boundary of r and R. See rectangle and string A in Figure 5. The first two segments in the polygonal chain of s coincide with the first two sides of r in a clockwise traversal of the boundary of r. The third segment of s is a subset of the third side of r stopping δ before the side of R. The fourth segment of s is parallel to the side of R containing $start(s)$ at distance δ from the boundary of R. There is a gap of δ between the end of the fourth segment of s and the first segment of s. When a rectangle $r \in Q$

shares two or three sides with R, the construction of the strings is illustrated in Figure 5.

The construction of the strings in S corresponding to the rectangles in R allows us to conclude that two rectangles r_1 and r_2 in Q intersect if and only if their corresponding strings s_1 and s_2 in S intersect.

Our $O(N^3)$ Maximum Weight Independent Set algorithm for outerstring graphs thus gives us the following theorem.

Theorem 2 *Given a rectangle R and a set Q of n weighted boundary rectangles inside R, a maximum weight disjoint set of rectangles in Q can be found in $O(n^3)$ time.*

References

[1] P. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(3):209–218, 1998.

[2] A. AhmadiNejad and H. Zarrabi-Zadeh. The maximum disjoint set of boundary rectangles. In *Proceedings of CCCG*, pages 302–307, 2014.

[3] S. Assadi, E. Emamjomeh-Zadeh, S. Yazdanbod, and H. Zarrabi-Zadeh. On the rectangle escape problem. In *Proceedings of CCCG*, pages 235–240, 2013.

[4] S. Benzer. On the topology of genetic fine structure. *Proceedings of the National Academy of Sciences*, 47:1607, 1959.

[5] S. Cabello, J. Cardinal, and S. Langerman. The clique problem in ray intersection graphs. *Discrete & Computational Geomtry*, 50(3):771–783, 2013.

[6] J. Chalopin and D. Gonçalves. Every planar graph is the intersection graph of segments in the plane. In *Proceedings of STOC*, pages 631–638, 2009.

[7] S. Ehrlich, S. Even, and R. Tarjan. Intersection graphs of curves in the plane. *J. Combin. Theory Ser. B.*, 21:8–20, 1976.

[8] H. Flier, M. Mihalák, A. Schöbel, P. Widmayer, and A. Zych. Vertex disjoint paths for dispatching in railways. In *Proceedings of ATMOS*, pages 61–73, 2010.

[9] H. Flier, M. Mihalák, P. Widmayer, and A. Zych. Maximum independent set in 2-direction outersegment graphs. In *Proceedings of WG*, pages 155–166, 2011.

[10] J. Fox and J. Pach. Computing the independence number of intersection graphs. In *Proceedings of SODA*, pages 1161–1165, 2011.

[11] J. Fox and J. Pach. Coloring K_k-free intersection graphs of geometric objects in the plane. *Eur. J. Comb.*, 33(5):853–866, 2012.

[12] J. Fox and J. Pach. String graphs and incomparability graphs. *Advances in Math.*, 230:1381–1401, 2012.

[13] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining with optimized two-dimensional association rules. *ACM Trans. on Database Systems*, 26(2):179–213, 2001.

[14] F. Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73:181–188, 2000.

[15] M. Golumbic, D. Rotem, and J. Urrutia. Comparability graphs and intersection graphs. *Discrete Math.*, 43:37–46, 1983.

[16] H. Kong, Q. Ma, T. Yan, and M. D. Wong. An optimal algorithm for finding disjoint rectangles and its applications to PCB routing. In *Proceedings of DAC*, pages 212–217, 2010.

[17] J. Kratochvíl. String graphs I. The number of critical nonstring graphs is infinite. *J. Comb. Theory, Ser. B*, 52(1):53–66, 1991.

[18] J. Kratochvíl. String graphs II. Recognizing string graphs is NP-hard. *J. Comb. Theory, Ser. B*, 52(1):67–78, 1991.

[19] J. Kratochvíl, M. Goljan, and P. Kučera. *String Graphs*. Academia, Prague, 1986.

[20] J. Kratochvíl, A. Lubiw, and J. Nešetřil. Noncrossing subgraphs in topological layouts. *SIAM J. Discrete Math.*, 4(2):223–244, 1991.

[21] J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *J. Comb. Theory, Ser. B*, 53(1):1–4, 1991.

[22] J. Kratochvíl and J. Nešetřil. Independent set and clique problems in intersection-defined classes of graphs. *Comment. Mat. Univ. Carolinae*, 31:85–93, 1990.

[23] M. Middendorf and F. Pfeiffer. The max clique problem in classes of string-graphs. *Discrete Math.*, 108:365–372, 1992.

[24] M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs is in NP. *J. Comput. Syst. Sci.*, 67(2):365–380, 2003.

[25] F. W. Sinden. Topology of thin film RC-circuits. *Bell System Tech. J.*, 45:1639–1662, 1966.