Minimizing Crossings in Constrained Two-Sided Circular Graph Layouts

Fabian Klute
Algorithms and Complexity Group, TU Wien
Vienna, Austria
fklute@ac.tuwien.ac.at

Martin Nöllenburg
Algorithms and Complexity Group, TU Wien
Vienna, Austria
noellenburg@ac.tuwien.ac.at
https://orcid.org/0000-0003-0454-3937

Abstract

Circular layouts are a popular graph drawing style, where vertices are placed on a circle and edges are drawn as straight chords. Crossing minimization in circular layouts is NP-hard. One way to allow for fewer crossings in practice are two-sided layouts that draw some edges as curves in the exterior of the circle. In fact, one- and two-sided circular layouts are equivalent to one-page and two-page book drawings, i.e., graph layouts with all vertices placed on a line (the *spine*) and edges drawn in one or two distinct half-planes (the *pages*) bounded by the spine. In this paper we study the problem of minimizing the crossings for a fixed cyclic vertex order by computing an optimal k-plane set of exteriorly drawn edges for $k \geq 1$, extending the previously studied case $k = 0$. We show that this relates to finding bounded-degree maximum-weight induced subgraphs of circle graphs, which is a graph-theoretic problem of independent interest. We show NP-hardness for arbitrary k, present an efficient algorithm for $k = 1$, and generalize it to an explicit XP-time algorithm for any fixed k. For the practically interesting case $k = 1$ we implemented our algorithm and present experimental results that confirm the applicability of our algorithm.

2012 ACM Subject Classification Human-centered computing → Graph drawings, Mathematics of computing → Graph algorithms, Theory of computation → Computational geometry

Keywords and phrases Graph Drawing, Circular Layouts, Crossing Minimization, Circle Graphs, Bounded-Degree Maximum-Weight Induced Subgraphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.53

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.05705`.

1 Introduction

Circular graph layouts are a popular drawing style to visualize graphs, e.g., in biology [16], and circular layout algorithms [21] are included in standard graph layout software [11] such as yFiles, Graphviz, or OGDF. In a circular graph layout all vertices are placed on a circle, while the edges are drawn as straight-line chords of that circle, see Fig. 1a. Minimizing the number of crossings between the edges is the main algorithmic problem for optimizing the readability of a circular graph layout. If the edges are drawn as chords, then all crossings are determined solely by the order of the vertices. By cutting the circle between any two vertices and straightening it, circular layouts immediately correspond to one-page book drawings, in which all vertices are drawn on a line (the *spine*) and all edges are drawn in one half-plane (the *page*) bounded by the spine. Finding a vertex order that minimizes the crossings is NP-hard [18]. Heuristics and approximation algorithms have been studied in numerous papers, see, e.g., [2, 13, 20].

Gansner and Koren [8] presented an approach to compute improved circular layouts for a given input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in a three-step process. The first step computes a vertex order of $\mathcal{V}$ that aims to minimize the overall edge length of the drawing, the second step determines a crossing-free subset of edges that are drawn outside the circle to reduce edge crossings in the interior (see Fig. 1b), and the third step introduces edge bundling to save ink and reduce clutter in the interior. The layouts by Gansner and Koren draw edges inside and outside the circle and thus are called *two-sided circular layouts*. Again, it is easy to see that two-sided circular layouts are equivalent to two-page book drawings, where the interior of the circle with its edges corresponds to the first page and the exterior to the second page.

Inspired by their approach we take a closer look at the second step of the above process, which, in other words, determines for a given cyclic vertex order an outerplane subgraph to be drawn outside the circle such that the remaining crossings of the chords are minimized. Gansner and Koren [8] solve this problem in $O(|\mathcal{V}|^3)$ time.[1] In fact, the problem is equivalent to finding a maximum independent set in the corresponding circle graph $G^\circ = (V, E)$, which is the intersection graph of the chords (see Section 2 for details). The maximum independent set problem in a circle graph can be solved in $O(\ell)$ time [23], where ℓ is the total chord length of the circle graph (here $|\mathcal{E}| \le \ell \le |\mathcal{E}|^2$; see Section 4.2 for a precise definition of ℓ).

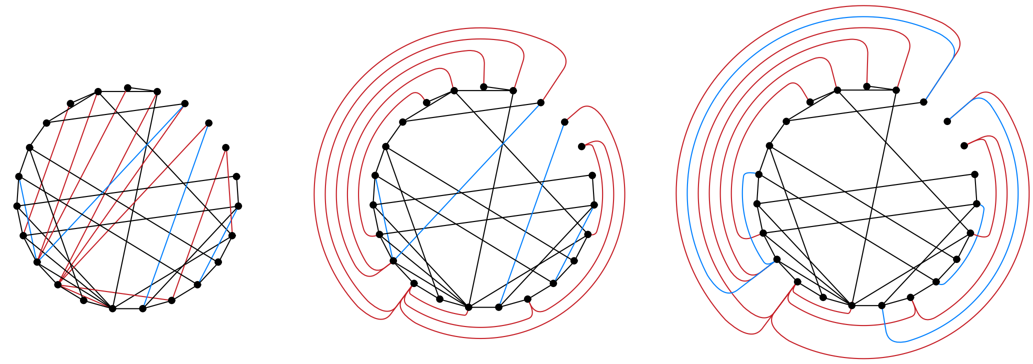
Contribution

We generalize the above crossing minimization problem from finding an outerplane graph to finding an outer k-plane graph, i.e., we ask for an edge set to be drawn outside the circle such that none of these edges has more than k crossings. Equivalently, we ask for a page assignment of the edges in a two-page book drawing, given a fixed vertex order, such that in one of the two pages each edge has at most k crossings. For $k = 0$ this is exactly the same problem considered by Gansner and Koren [8]. An example for $k = 1$ is shown in Fig. 1c. More generally, studying drawings of non-planar graphs with a bounded number of crossings per edge is a topic of great interest in graph drawing, see [15, 17].

We model the outer k-plane crossing minimization problem in two-sided circular layouts as a bounded-degree maximum-weight induced subgraph (BDMWIS) problem in the corresponding circle graph (Section 2). The BDMWIS problem is a natural generalization of the weighted independent set problem (setting the degree bound $k = 0$), which was the basis for Gansner and Koren's approach [8]. It is itself a weighted special case of the bounded-degree vertex deletion problem [3, 5, 7], a well-studied algorithmic graph problem of independent interest. For arbitrary k we show NP-hardness of the BDMWIS problem in Section 3. Our algorithms in Section 4 are based on dynamic programming using interval representations of circle graphs. For the case $k = 1$, where at most one crossing per exterior edge is permitted, we solve the BDMWIS problem for circle graphs in $O(|\mathcal{E}|^4)$ time. We then generalize our algorithm and obtain a problem-specific XP-time algorithm for circle graphs and any fixed k, whose running time is $O(|\mathcal{E}|^{2k+2})$. We note that the pure existence of an XP-time algorithm can also be derived from applying a metatheorem of Fomin et al. [6] using counting monadic second order (CMSO) logic, but the resulting running times are far worse. Finally, in Section 5, we present the results of a first experimental study comparing the crossing numbers of two-sided circular layouts for the cases $k = 0$ and $k = 1$.

---

[1] The paper claims $O(|\mathcal{V}|^2)$ time without a proof; the immediate running time of their algorithm is $O(|\mathcal{V}|^3)$.



(a) One-sided layout with 125 crossings.

(b) Two-sided layout for $k = 0$ with 48 crossings.

(c) Two-sided layout for $k = 1$ with 30 crossings.

Figure 1 Circular layouts of a graph $(\mathcal{G}, \pi)$ (23 vertices, 45 edges) computed by our algorithms.

2 Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph and π a cyclic order of $\mathcal{V}$. We arrange the vertices in order π on a circle C and draw edges as straight chords to obtain a (*one-sided*) *circular drawing* Γ, see Fig. 1a. Note that all crossings of Γ are fully determined by π: two edges cross iff their endpoints alternate in π. Our goal in this paper is to find a subset of edges to be drawn in the unbounded region outside C with no more than k crossings per edge in order to minimize the total number of edge crossings or the number of remaining edge crossings inside C.

More precisely, in a *two-sided circular drawing* Δ of $(\mathcal{G}, \pi)$ we still draw all vertices on a circle C in the order π, but we split the edges into two disjoint sets $\mathcal{E}_1$ and $\mathcal{E}_2$ with $\mathcal{E}_1 \cup \mathcal{E}_2 = \mathcal{E}$. The edges in $\mathcal{E}_1$ are drawn as straight chords, while the edges in $\mathcal{E}_2$ are drawn as simple curves in the exterior of C, see Fig. 1c. Asking for a set $\mathcal{E}_2$ that globally minimizes the crossings in Δ is equivalent to the NP-hard fixed linear crossing minimization problem in 2-page book drawings [19]. Hence we add the additional constraint that the exterior drawing induced by $\mathcal{E}_2$ is *outer k-plane*, i.e., each edge in $\mathcal{E}_2$ is crossed by at most k other edges in $\mathcal{E}_2$. This is motivated by the fact that, due to their detour, exterior edges are already harder to read and hence should not be further impaired by too many crossings. The parameter k, which can be assumed to be small, gives us control on the maximum number of crossings per exterior edge. Previous work [8] is limited to the case $k = 0$.

2.1 Problem transformation

Instead of working with a one-sided input layout Γ of $(\mathcal{G}, \pi)$ directly we consider the corresponding *circle graph* $G^\circ = (V, E)$ of $(\mathcal{G}, \pi)$. The vertex set V of G° has one vertex for each edge in $\mathcal{E}$ and two vertices $u, v \in V$ are connected by an edge (u, v) in E if and only if the chords corresponding to u and v cross in Γ, i.e., their endpoints alternate in π. The number of vertices $|V|$ of G° thus equals the number of edges $|\mathcal{E}|$ of $\mathcal{G}$ and the number of edges $|E|$ of G° equals the number of crossings in Γ. Moreover, the degree $\deg(v)$ of a vertex v in G° is the number of crossings of the corresponding edge in Γ.

Next we show that we can reduce our outer k-plane crossing minimization problem in two-sided circular layouts of $(\mathcal{G}, \pi)$ to an instance of the following bounded-degree maximum-weight induced subgraph problem for G°.

▶ **Problem 1** (Bounded-Degree k Maximum-Weight Induced Subgraph (k-BDMWIS)). *Let $G = (V, E)$ be a weighted graph with a vertex weight $w(v) \in \mathbb{R}^+$ for each $v \in V$ and an edge weight $w(u, v) \in \mathbb{R}^+$ for each $(u, v) \in E$ and let $k \in \mathbb{N}$. Find a set $V' \subset V$ such that the induced subgraph $G[V'] = (V', E')$ has maximum vertex degree k and maximizes the weight*

$$W = W(G[V']) = \sum_{v \in V'} w(v) - \sum_{(u,v) \in E'} w(u, v).$$

For general graphs it follows immediately from Yannakakis [24] that k-BDMWIS is NP-hard, but restricting the graph class to circle graphs makes the problem significantly easier, at least for constant k, as we show in this paper.

For our reduction it remains to assign suitable vertex and edge weights to G°. We define $w(v) = \deg(v)$ for all vertices $v \in V$ and $w(u, v) = 1$ or, alternatively, $w(u, v) = 2$ for all edges $(u, v) \in E$, depending on the type of crossings to minimize.

▶ **Lemma 2.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with cyclic vertex order π and $k \in \mathbb{N}$. Then a maximum-weight degree-k induced subgraph in the corresponding weighted circle graph $G^\circ = (V, E)$ induces an outer k-plane graph that minimizes the number of crossings in the corresponding two-sided layout Δ of $(\mathcal{G}, \pi)$.*

Proof. Let $V^* \subset V$ be a vertex set that induces a maximum-weight subgraph of degree at most k in G°. Since vertices in G° correspond to edges in $\mathcal{G}$, we can choose $\mathcal{E}^* = V^*$ as the set of exterior edges in Δ. Each edge in G° corresponds to a crossing in the one-sided circular layout Γ. Hence each edge in the induced graph $G^\circ[V^*]$ corresponds to an exterior crossing in Δ. Since the maximum degree of $G^\circ[V^*]$ is k, no exterior edge in Δ has more than k crossings.

The degree of a vertex $v \in V^*$ (and thus its weight $w(v)$) equals the number of crossings that are removed from Γ by drawing the corresponding edge in $\mathcal{E}^*$ in the exterior part of Δ. However, if two vertices in V^* are connected by an edge, their corresponding edges in $\mathcal{E}^*$ necessarily cross in the exterior part of Δ and we need to add a correction term, otherwise the crossing would be counted twice. So for edge weights $w(u, v) = 1$ the weight W maximized by V^* equals the number of crossings that are removed from the interior part of Δ. For $w(u, v) = 2$, though, the weight W equals the number of crossings that are removed from the interior, but not counting those that are simply shifted to the exterior of Δ. ◀

Lemma 2 tells us that instead of minimizing the crossings in two-sided circular layouts with an outer k-plane exterior graph, we can focus on solving the k-BDMWIS problem for circle graphs in the rest of the paper.

2.2 Interval representation of circle graphs

There are two alternative representations of circle graphs. The first one is the *chord representation* as a set of chords of a circle (i.e., a one-sided circular layout), whose intersection graph actually serves as the very definition of circle graphs. The second and less immediate representation is the *interval representation* as an *overlap graph*, which is more convenient for describing our algorithms. In an interval representation each vertex is represented as a closed interval $I \subset \mathbb{R}$. Two vertices are adjacent if and only if the two corresponding intervals partially overlap, i.e., they intersect but neither interval contains the other.

Gavril [10] showed that circle graphs and overlap graphs represent the same class of graphs. To obtain an interval representation from a chord representation Γ on a circle C the idea is to pick a point p on C, which is not the endpoint of a chord, rotate Γ such that

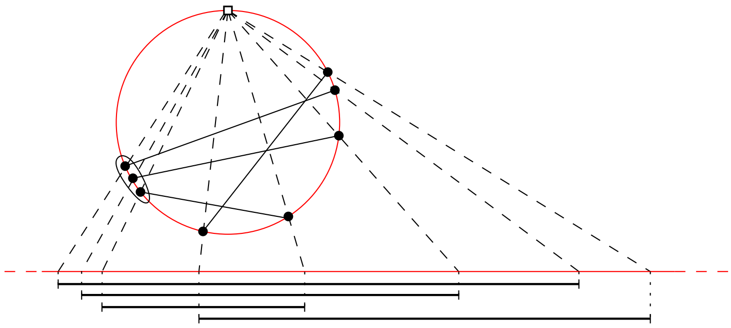


Figure 2 An example projection of the chord representation of a circle graph (here: $K_{1,3}$) to obtain an interval representation of the same graph as an overlap graph. Marked groups of endpoints indicate how chords incident to the same vertex are separated before the projection.

p is the topmost point of Γ and project the chords from p onto the real line below C, see Fig. 2. Each chord is then represented as a finite interval and two chords intersect if and only if their projected intervals partially overlap. We can further assume that all endpoints of the projected intervals are distinct by locally separating chords with a shared endpoint in Γ before the projection, such that the intersection graph of the chords does not change.

3 NP-hardness

For arbitrary, non-constant $k \in \mathbb{N}$ we show that k-BDMWIS is NP-hard, even on circle graphs. Our reduction is from the MINIMUM DOMINATING SET problem, which is NP-hard on circle graphs [12].

▶ **Problem 3** (MINIMUM DOMINATING SET). *Given a graph $G = (V, E)$, find a set $V' \subseteq V$ of minimum cardinality such that for each $u \in V \setminus V'$ there is a vertex $v \in V'$ with $(u, v) \in E$.*

▶ **Theorem 4.** *k-BDMWIS is NP-hard on circle graphs, even if all vertex weights are one and all edge weights are zero.*

Proof. Given an instance of MINIMUM DOMINATING SET on a circle graph $G = (V, E)$ we construct an instance of k-BDMWIS. First let $G' = (V', E')$ be a copy of G. We set the degree bound k equal to the maximum degree of G and attach new leaves to each vertex $v \in V'$ until every (non-leaf) vertex in G' has degree $k + 1$. Note that G' remains a circle graph when adding leaves. We set the weights to $w(v) = 1$ for $v \in V'$ and $w(u, v) = 0$ for $(u, v) \in E'$. This implies that the weight W to be maximized is just the number of vertices in the induced subgraph.

Now given a minimum dominating set $V_d \subseteq V$ of G, we know that for every vertex $v \in V$ either $v \in V_d$ or there exists a vertex $u \in V_d$ such that $(u, v) \in E$. This means if we set $V_s = V' \setminus V_d$ the graph $G'[V_s]$ has maximum degree k, since for every $v \in V_s$ at least one neighbor is in V_d and the maximum degree in G' is $k+1$. Since V_d is a minimum dominating set, V_s, for which we can assume that it contains all leaves, is the largest set of vertices such that $G'[V_s]$ has maximum degree k. Hence V_s is a solution to the k-BDMWIS problem on G'.

Conversely let $V_s \subseteq V'$ be a solution to the k-BDMWIS problem on G'. Again we can assume that V_s contains all leaves of G'. Otherwise let $u \in V' \setminus V_s$ be a leaf of G' with unique neighbor $v \in V'$. The only possible reason that $u \notin V_s$ is that $v \in V_s$ and the degree of v in $G'[V_s]$ is $\deg(v) = k$. If we replace in V_s a non-leaf neighbor w of v (which must exist) by the leaf u, the resulting set has the same cardinality and satisfies the degree constraint. Now let $V_d = V' \setminus V_s$. By our assumption V_d contains no leaves of G' and $V_d \subseteq V$. Since every vertex in $G'[V_s]$ has degree at most k we know that each $v \in V \setminus V_d$ must have a neighbor $u \in V_d$, otherwise it would have degree $k+1$ in $G'[V_s]$. Thus V_d is a dominating set. Further, V_d is a minimum dominating set. If there was a smaller dominating set V'_d in G then $V' \setminus V'_d$ would be a larger solution than V_s for the k-BDMWIS problem on G', which is a contradiction. ◀

4 Algorithms for k-BDMWIS on circle graphs

Before describing our dynamic programming algorithms for $k = 1$ and the generalization to $k \geq 2$ in this section, we introduce the necessary basic definitions and notation using the interval perspective on k-BDMWIS for circle graphs.

4.1 Notation and definitions

Let $G = (V, E)$ be a circle graph and $\mathcal{I} = \{I_1, \dots, I_n\}$ an interval representation of G with n intervals that have $2n$ distinct endpoints as defined in Section 2.2. Let $\sigma(\mathcal{I}) = \{\sigma_1, \dots, \sigma_{2n}\}$ be the set of all interval endpoints and assume that they are sorted in increasing order, i.e., $\sigma_i < \sigma_j$ for all $i < j$. We may in fact assume without loss of generality that $\sigma(\mathcal{I}) = \{1, \dots, 2n\}$ by mapping each interval $[\sigma_l, \sigma_r] \in \mathcal{I}$ to the interval $[l, r]$ defined by its index ranks. Clearly the order of the endpoints of two intervals $[\sigma_l, \sigma_r]$ and $[\sigma_{l'}, \sigma_{r'}]$ is exactly the same as the order of the endpoint of the intervals $[l, r]$ and $[l', r']$ and thus the overlap or circle graph defined by the new interval set is exactly the same as the one defined by $\mathcal{I}$.

For two distinct intervals $I = [a, b]$ and $J = [c, d] \in \mathcal{I}$ we say that I and J *overlap* if $a < c < b < d$ or $c < a < d < b$. Two overlapping intervals correspond to an edge in G. For an interval $I \in \mathcal{I}$ and a subset $\mathcal{I}' \subseteq \mathcal{I}$ we define the overlap set $\mathcal{P}(I, \mathcal{I}') = \{J \mid J \in \mathcal{I}' \text{ and } I, J \text{ overlap}\}$. Further, for $I = [a, b]$, we define the forward overlap set $\overrightarrow{\mathcal{P}}(I, \mathcal{I}') = \{J \mid J = [c, d] \in \mathcal{P}(I, \mathcal{I}') \text{ and } c < b < d\}$ of intervals overlapping on the right side of I and the set $\mathcal{P}(\mathcal{I}') = \{\{I, J\} \mid I, J \in \mathcal{I}' \text{ and } J \in \mathcal{P}(I, \mathcal{I}')\}$ of all overlapping pairs of intervals in $\mathcal{I}'$. If $J \subset I$, i.e., $a < c < d < b$, we say that I *nests* J (or J is *nested* in I). Nested intervals do not correspond to edges in G. For a subset $\mathcal{I}' \subseteq \mathcal{I}$ we define the set of all intervals nested in I as $\mathcal{N}(I, \mathcal{I}') = \{J \mid J \in \mathcal{I}' \text{ and } J \text{ is nested in } I\}$.

Let $\mathcal{I}' \subseteq \mathcal{I}$ be a set of n' intervals. We say $\mathcal{I}'$ is *connected* if its corresponding overlap or circle graph is connected. Further let $\sigma(\mathcal{I}') = \{i_1, \dots, i_{2n'}\}$ be the sorted interval endpoints of $\mathcal{I}'$. The *span* of $\mathcal{I}'$ is defined as $\operatorname{span}(\mathcal{I}') = i_{2n'} - i_1$ and the *fit* of the set $\mathcal{I}'$ is defined as $\operatorname{fit}(\mathcal{I}') = \max\{i_{j+1} - i_j \mid 1 \leq j < 2n'\}$.

For a weighted circle graph $G = (V, E)$ with interval representation $\mathcal{I}$ we can immediately assign each vertex weight $w(v)$ as an interval weight $w(I_v)$ to the interval $I_v \in \mathcal{I}$ representing v and each edge weight $w(u, v)$ to the overlapping pair of intervals $\{I_u, I_v\} \in \mathcal{P}(\mathcal{I})$ that represents the edge $(u, v) \in E$. We can now phrase the k-BDMWIS problem for a circle graph in terms of its interval representation, i.e., given an interval representation $\mathcal{I}$ of a circle graph G, find a subset $\mathcal{I}' \subseteq \mathcal{I}$ such that no $I \in \mathcal{I}'$ overlaps more than k other intervals in $\mathcal{I}'$ and such that the weight $W(\mathcal{I}') = \sum_{I \in \mathcal{I}'} w(I) - \sum_{\{I,J\} \in \mathcal{P}(\mathcal{I}')} w(I, J)$ is maximized. We call such an optimal subset $\mathcal{I}'$ of intervals a *max-weight k-overlap set*.

4.2 Properties of max-weight 1-overlap sets

The basic idea for our dynamic programming algorithm is to decompose any 1-*overlap set*, i.e., a set of intervals, in which no interval overlaps more than one other interval, into a sequence of independent single intervals and overlapping interval pairs. Consequently, we can

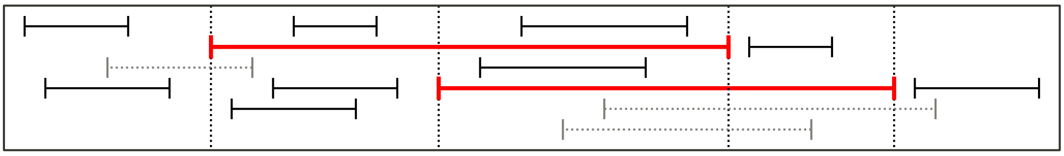


Figure 3 Split along the two thick red intervals. The dotted intervals are discarded and we recurse on the five sets with black intervals.

find a max-weight 1-overlap set by optimizing over all possible ways to select a single interval or an overlapping interval pair and recursively solving the induced independent subinstances that are obtained by splitting the instance according to the selected interval(s).

Let $\mathcal{I}$ be a set of intervals. For $x, y \in \mathbb{R} \cup \{\pm\infty\}$ with $x \leq y$ we define the set $\mathcal{I}[x, y] = \{I \in \mathcal{I} \mid I \subseteq [x, y]\}$ as the subset of $\mathcal{I}$ contained in $[x, y]$. Note that $\mathcal{I}[-\infty, \infty] = \mathcal{I}$. For any $I = [a, b] \in \mathcal{I}[x, y]$ we can *split* $\mathcal{I}[x, y]$ *along* I into the three sets $\mathcal{I}[x, a]$, $\mathcal{I}[a, b]$, $\mathcal{I}[b, y]$. This split corresponds to selecting I as an interval without overlaps in a candidate 1-overlap set. All intervals which are not contained in one of the three sets will be discarded after the split.

Similarly, we can split $\mathcal{I}[x, y]$ along a pair of overlapping intervals $I = [a, b]$, $J = [c, d] \in \mathcal{I}$ to be included in candidate solution. Without loss of generality let $a < c < b < d$. Then the split creates the five sets $\mathcal{I}[x, a]$, $\mathcal{I}[a, c]$, $\mathcal{I}[c, b]$, $\mathcal{I}[b, d]$, $\mathcal{I}[d, y]$, see Fig. 3. Again, all intervals which are not contained in one of the five sets are discarded. The next lemma shows that none of the discarded overlapping intervals can be included in a 1-overlap set together with I and J.

▶ **Lemma 5.** *For any $x \in \mathbb{R}$ at most two overlapping intervals $I = [a, b]$, $J = [c, d] \in \mathcal{I}$ with $a \leq x \leq b$ and $c \leq x \leq d$ can be part of a 1-overlap set of $\mathcal{I}$.*

Proof. Assume there is a third interval $K = [e, f] \in \mathcal{I}$ with $e \leq x \leq f$ in a 1-overlap set, which overlaps I or J or both. Interval K cannot be added to the 1-overlap set without creating at least one interval that overlaps two other intervals, which is not allowed in a 1-overlap set. ◀

Our algorithm for the max-weight 1-overlap set problem extends some of the ideas of the algorithm presented by Valiente for the independent set problem in circle graphs [23]. In our analysis we use Valiente's notion of *total chord length*, where the chord length is the same as the *length* $\ell(I) = j - i$ of the corresponding interval $I = [i, j] \in \mathcal{I}$. The *total interval length* can then be defined as $\ell = \ell(\mathcal{I}) = \sum_{I \in \mathcal{I}} \ell(I)$. We use the following bound in our analysis.

▶ **Lemma 6.** *Let $\mathcal{I}$ be a set of intervals and γ be the maximum degree of the corresponding overlap or circle graph, then $\sum_{I \in \mathcal{I}} \sum_{J \in \mathcal{P}(I, \mathcal{I})} (\ell(I) + \ell(J)) = O(\gamma\ell)$.*

Proof. We first observe that $J \in \mathcal{P}(I, \mathcal{I})$ if and only if $I \in \mathcal{P}(J, \mathcal{I})$. So in total each interval in $\mathcal{I}$ appears at most γ times as I and at most γ times as J in the double sum, i.e., no interval in $\mathcal{I}$ appears more than 2γ times and the bound follows. ◀

4.3 An algorithm for max-weight 1-overlap sets

Our algorithm to compute max-weight 1-overlap sets runs in two phases. In the first phase, we compute the weights of optimal solutions on subinstances of increasing size by recursively re-using solutions of smaller subinstances. In the second phase we optimize over all ways of combining optimal subsolutions to obtain a max-weight 1-overlap set.

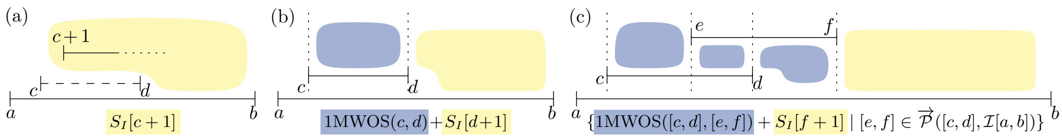


Figure 4 Illustration of Recurrence (3). The dashed intervals are discarded, while solid ones are considered in the solution.

The subinstances of interest are defined as follows. Let $\mathcal{I}' \subseteq \mathcal{I}$ be a connected set of intervals and let $l = l(\mathcal{I}')$ and $r = r(\mathcal{I}')$ be the leftmost and rightmost endpoints of all intervals in $\mathcal{I}'$. We define the value $1\text{MWOS}(\mathcal{I}')$ as the maximum weight of a 1-overlap set on $\mathcal{I}[l, r]$ that includes $\mathcal{I}'$ in the 1-overlap set (if one exists). Lemma 5 implies that it is sufficient to compute the 1MWOS values for single intervals $I \in \mathcal{I}$ and overlapping pairs $I, J \in \mathcal{I}$ since any connected set of three or more intervals cannot be a 1-overlap set any more.

We start with the computation of $1\text{MWOS}(I)$ for a single interval $I = [a, b] \in \mathcal{I}$. Using a recursive computation scheme of 1MWOS that uses increasing interval lengths we may assume by induction that for any interval $J \in \mathcal{I}$ with $\ell(J) < \ell(I)$ and any overlapping pair of intervals $J, K \in \mathcal{I}$ with $\text{span}(J, K) < \ell(I)$ the sets $1\text{MWOS}(J)$ and $1\text{MWOS}(J, K)$ are already computed. If we select I for the 1-overlap set as a single interval without overlaps, we need to consider for $1\text{MWOS}(I)$ only those intervals nested in I. Refer to Fig. 4 for an illustration. The value of $1\text{MWOS}(I)$ is determined using an auxiliary recurrence $S_I[x]$ for $a \leq x \leq b$ and the weight $w(I)$ resulting from the choice of I:

$$1\text{MWOS}([a, b]) = S_I[a + 1] + w(I). \tag{1}$$

For a fixed interval $I = [a, b]$ the value $S_I[x]$ represents the weight of an optimal solution of $\mathcal{I}[x, b]$. To simplify the definition of recurrence $S_I[x]$ we define the set $D_S([c, d], \mathcal{I}[a, b])$ with $[c, d] \in \mathcal{I}[a, b]$, in which we collect all 1MWOS values for pairs composed of $[c, d]$ and an interval in $\overrightarrow{\mathcal{P}}([c, d], \mathcal{I}[a, b])$ (see Fig. 4(c)) as

$$D_S([c, d], \mathcal{I}[a, b]) = \{1\text{MWOS}([c, d], [e, f]) + S_I[f + 1] \mid [e, f] \in \overrightarrow{\mathcal{P}}([c, d], \mathcal{I}[a, b])\}. \tag{2}$$

The main idea of the definition of $S_I[x]$ is a maximization step over the already computed sub-solutions that may be composed to an optimal solution for $\mathcal{I}[x, b]$. To stop the recurrence we set $S_I[b] = 0$ and for every end-point d of an interval $[c, d] \in \mathcal{I}[a, b]$ we set $S_I[d] = S_I[d+1]$. It remains to define the recurrence for the start-point c of each interval $[c, d] \in \mathcal{I}[a, b]$:

$$S_I[c] = \max\{\{S_I[c + 1], 1\text{MWOS}([c, d]) + S_I[d + 1]\} \cup D_S([c, d], \mathcal{I}[a, b])\}. \tag{3}$$

Figure 4 depicts which of the possible configurations of selected intervals is represented by which values in the maximization step of eq. (3). The first option (Fig. 4(a)) is to discard the interval $[c, d]$, the second option (Fig. 4(b)) is to select $[c, d]$ as a single interval, and the third option (Fig. 4(c)) is to select $[c, d]$ and an interval in its forward overlap set.

▶ **Lemma 7.** *Let $\mathcal{I}$ be a set of intervals and $I \in \mathcal{I}$, then the value $1MWOS(I)$ can be computed in $O(\gamma\, \ell(I))$ time assuming all $1MWOS(J)$ and $1MWOS(J, K)$ values are computed for $J, K \in \mathcal{I}$, $\ell(J) < \ell(I)$ and* $\text{span}(J, K) < \ell(I)$.

Proof. Recurrence (1) is correct if $S[a+1]$ is exactly the weight of a max-weight 1-overlap set on the set $\mathcal{N}(I, \mathcal{I})$, the set of nested intervals of I. The proof is by induction over the number of intervals in $\mathcal{N}(I, \mathcal{I})$. In case $\mathcal{N}(I, \mathcal{I})$ is empty $b = a + 1$ and with $S[b] = 0$ Recurrence (1) is correct.

Now let $\mathcal{N}(I,\mathcal{I})$ consist of one or more intervals. By Lemma 5 there can only be three cases of how an interval $J \in \mathcal{N}(I,\mathcal{I})$ contributes. We can decide to discard J, to add it as a singleton interval which allows us to split $\mathcal{N}(I,\mathcal{I})$ along J or to add an overlapping pair $J, K \in \mathcal{N}(I,\mathcal{I})$ such that $K \in \overrightarrow{\mathcal{P}}(J,\mathcal{I}[a,b])$ and split $\mathcal{N}(I,\mathcal{I})$ along J, K.

For the start-point c of an interval $J = [c,d]$ the maximization in the definition of S in Recurrence (3) exactly considers these three possibilities (recall Fig. 4). For all end-points aside from b we simply use the value of the next start-point or $S[b] = 0$ which ends the recurrence. Since all 1MWOS(J) and 1MWOS(J,K) are computed for $J, K \in \mathcal{I}$, $\ell(J) < \ell(I)$ and $\operatorname{span}(J,K) < \ell(I)$ the auxiliary table S is computed in one iteration across $\sigma(\mathcal{I}[a,b])$.

The overall running time is dominated by traversing the D_S sets, which contain at most γ values. This has to be done for every start-point of an interval in $\mathcal{N}(I,\mathcal{I})$ which leads to an overall computation time of $O(\gamma\,\ell(I))$ for 1MWOS(I). ◀

Until now we only considered computing the 1MWOS value of a single interval, but we still need to compute 1MWOS for pairs of overlapping intervals. Let $I = [c,d], J = [e,f] \in \mathcal{I}$ be two intervals such that $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$. If we split $\mathcal{I}$ along these two intervals we find three independent regions (recall Fig. 4(c)) and obtain

$$\text{1MWOS}(I,J) = L_{I,J}[c+1] + M_{I,J}[e+1] + R_{I,J}[d+1] + w(I) + w(J) - w(I,J). \tag{4}$$

The auxiliary recurrences $L_{I,J}, M_{I,J}, R_{I,J}$ are defined for the three independent regions in the very same way as S_I above with the exception that $L_{I,J}[e] = 0, M_{I,J}[d] = 0$ and $R_{I,J}[f] = 0$. Hence, following essentially the same proof as in Lemma 7 we obtain

▶ **Lemma 8.** *Let $\mathcal{I}$ be a set of intervals and $I, J \in \mathcal{I}$ with $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$, then 1MWOS($I,J$) can be computed in $O(\gamma \operatorname{span}(I,J))$ time assuming all 1MWOS(K) and 1MWOS(K,L) values are computed for $K, L \in \mathcal{I}$, $\ell(K) < \operatorname{fit}(I,J)$ and $\operatorname{span}(K,L) < \operatorname{fit}(I,J)$.*

▶ **Lemma 9.** *Let $\mathcal{I}$ be a set of intervals. The 1MWOS values for all $I \in \mathcal{I}$ and all pairs $I, J \in \mathcal{I}$ with $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$ can be computed in $O(\gamma^2\ell)$ time.*

Proof. For an interval $I \in \mathcal{I}$ the value 1MWOS(I) is computed in $O(\gamma\,\ell(I))$ time by Lemma 7. With $\ell = \sum_{I\in\mathcal{I}} \ell(I)$ the claim follows for all $I \in \mathcal{I}$.

By Lemma 8 the value 1MWOS(I,J) can be computed in $O(\gamma \operatorname{span}(I,J))$ time for each overlapping pair I, J with $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$. Since $\operatorname{span}(I,J) \le \ell(I) + \ell(J)$ the time bound of $O(\gamma^2\ell)$ follows by applying Lemma 6. ◀

In the second phase of our algorithm we compute the maximum weight of a 1-overlap set for $\mathcal{I}$ by defining another recurrence $T[x]$ for $x \in \sigma(\mathcal{I})$ and re-using the 1MWOS values. The recurrence for T is defined similarly to the recurrence of S_I above. We set $T[2n] = 0$. Let $I = [a,b] \in \mathcal{I}$ be an interval and $b \neq 2n$, then

$$T[b] = T[b+1] \qquad\qquad T[a] = \max \begin{Bmatrix} \{T[a+1]\} \cup \\ \{\text{1MWOS}([a,b]) + T[b+1]\} \cup \\ D_T(I,\mathcal{I}) \end{Bmatrix}, \tag{5}$$

where D_T is defined analogously to D_S by replacing the recurrence S_I with T in eq. (2). The maximum weight of a 1-overlap set for $\mathcal{I}$ is found in $T[1]$.

▶ **Theorem 10.** *A max-weight 1-overlap set for a set of intervals $\mathcal{I}$ can be computed in $O(\gamma^2\ell) \subseteq O(|\mathcal{I}|^4)$ time, where ℓ is the total interval length and γ is the maximum degree of the corresponding overlap graph.*

Proof. The time to compute all 1MWOS values is $O(\gamma^2\ell)$ with Lemma 9. As argued the optimal solution is found by computing $T[1]$. The time to compute T in Recurrence (5) is again dominated by the maximization, which itself is dominated by the evaluation of the D_T sets. The size of these sets is exactly γ times the sum we bounded in Lemma 6. So the total time to compute T is $O(\gamma^2\ell)$. Hence the total running time is $O(\gamma^2\ell)$. From $\gamma \leq |\mathcal{I}|$ and $\ell \leq |\mathcal{I}|^2$ we obtain the coarser bound $O(|\mathcal{I}|^4)$.

It remains to show the correctness of Recurrence (5). Again we can treat it with the same induction used in the proof of Lemma 7. To see this we introduce an interval $[0, 2n+1]$ with weight zero. Now the computation of the maximum weight of a 1-overlap set for $\mathcal{I}$ is the same as computing all 1MWOS values for the instance $\mathcal{I} \cup \{[0, 2n+1]\}$. Using standard backtracking, the same algorithm can be used to compute the max-weight 1-overlap set instead of only its weight. ◀

4.4 An XP-algorithm for max-weight k-overlap sets

In this section we generalize our algorithm to $k \geq 2$. While it is not possible to directly generalize Recurrences (3) and (5) we do use similar concepts. The difficulty for $k > 1$ is that the solution can have arbitrarily large connected parts, e.g., a 2-overlap set can include arbitrarily long paths and cycles. So we can no longer partition an instance along connected components into a constant number of independent sub-instances as we did for the case $k = 1$. Due to space constraints we only sketch the main ideas here and refer the reader to the full version for all omitted proofs.

We first generalize the definition of 1MWOS. Let $\mathcal{I}$ be a set of n intervals as before and $I = [a, b] \in \mathcal{I}$. We define the value kMWOS(I) as the maximum weight of a k-overlap set on $\mathcal{I}[a, b]$ that includes I in the k-overlap set (if one exists). When computing such a value kMWOS(I), we consider all subsets $\mathcal{J} \subseteq \mathcal{P}(I, \mathcal{I})$ of cardinality $|\mathcal{J}| \leq k$ of at most k neighbors of I to be included in a k-overlap set, while $\mathcal{P}(I, \mathcal{I}) \setminus \mathcal{J}$ is excluded.

For keeping track of how many intervals are still allowed to overlap each interval we introduce the *capacity* of each interval boundary $i \in \sigma(\mathcal{I}) = \{1, 2, \ldots, 2n\}$. These capacities are stored in a vector $\lambda = (\lambda_1, \ldots, \lambda_{2n})$, where each λ_i is the capacity of the interval boundary $i \in \sigma(\mathcal{I})$. Each λ_i is basically a value in the set $\{0, 1, \ldots, k\}$ that indicates how many additional intervals may still overlap the interval corresponding to i, see Fig. 5. We actually define kMWOS$_\lambda([a, b])$ as the maximum weight of a k-overlap set in $\mathcal{I}[a, b]$ with pre-defined capacities λ. In the full version of the paper we prove that the number of relevant vectors λ to consider for each interval can be bounded by $O(\gamma^k)$, where γ is the maximum degree of the overlap graph corresponding to $\mathcal{I}$.

For our recursive definition we assume that when computing kMWOS$_\lambda(I)$ all values kMWOS$_\lambda(J)$ with $J \in \mathcal{I}$ and $\ell(J) < \ell(I)$ are already computed. The following recurrence computes one kMWOS$_\lambda(I)$ value given a valid capacity vector λ and an interval $I = [a, b] \in \mathcal{I}$

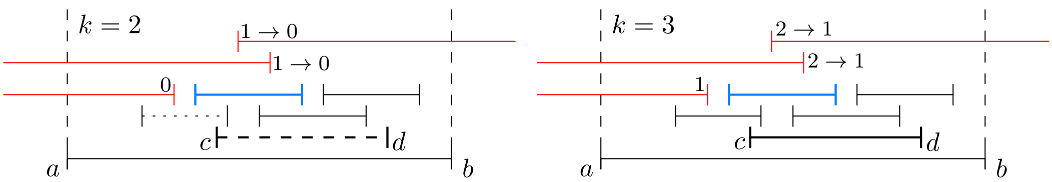


Figure 5 Examples for $k = 2$ and $k = 3$. The red intervals are in a solution set. The arrows indicate how the capacities change if the blue interval is included in a solution. For $k = 2$ we cannot use the interval $[c, d]$ since some capacities are zero, but for $k = 3$ it remains possible.

$$k\text{MWOS}_\lambda([a,b]) = S_{I,\lambda}[a+1] + w([a,b]). \tag{6}$$

This means that we select I for the k-overlap set, add its weight $w(I)$, and recursively solve the subinstance of intervals nested in I subject to the capacities λ. As in the approach for the 1MWOS values the main work is done in recurrence $S_{I,\lambda}[x]$ where $x \in \sigma(\mathcal{I}[a,b])$. In the full version of the paper we prove the correctness of this computation using a similar induction-based proof as Lemma 7 for $k = 1$, but being more careful with the computation of the correct weights. Lemma 11 is a simplified version of this.

▶ **Lemma 11.** *Let $\mathcal{I}$ be a set of intervals, $I \in \mathcal{I}$, λ a valid capacity vector for I, and γ the maximum degree of the corresponding overlap graph. Then kMWOS$_\lambda(I)$ can be computed in $O(\gamma^k\,\ell(I))$ time once the kMWOS$_\lambda(J)$ values are computed for all $J \in \mathcal{I}$ with $\ell(J) < \ell(I)$.*

Applying Lemma 11 to all $I \in \mathcal{I}$ and all valid capacity vectors λ results in a running time of $O(\gamma^{2k}\ell)$ to compute all $k\text{MWOS}_\lambda(I)$ values, where ℓ is the total interval length.

Now that we know how to compute all values $k\text{MWOS}_\lambda(I)$ for all $I \in \mathcal{I}$ and all relevant capacity vectors λ, we can obtain the optimal solution by introducing a dummy interval $\hat{I}$ with weight $w(\hat{I}) = 0$ that nests the entire set $\mathcal{I}$. We compute the value $k\text{MWOS}_{\hat{\lambda}}(\hat{I})$ for a capacity vector $\hat{\lambda}$ that puts no prior restrictions on the intervals in $\mathcal{I}$. This solution obviously contains the max-weight k-overlap set for $\mathcal{I}$. We summarize:

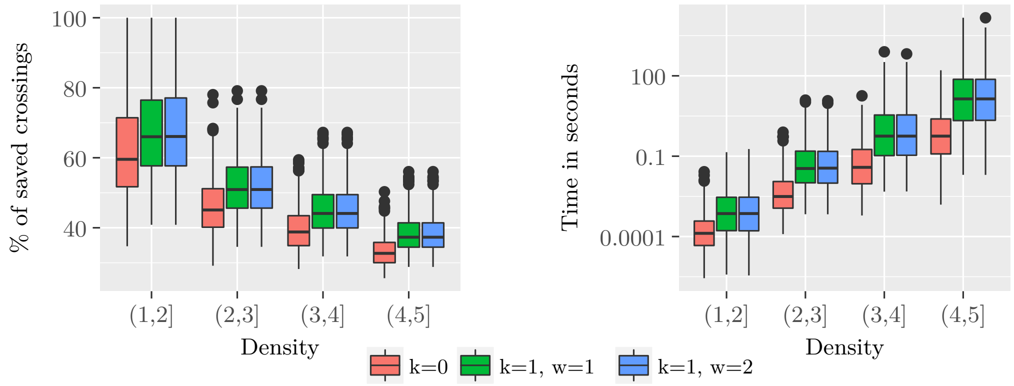
▶ **Theorem 12.** *A max-weight k-overlap set for a set of intervals $\mathcal{I}$ can be computed in $O(\gamma^{2k}\ell) \subseteq O(|\mathcal{I}|^{2k+2})$ time, where ℓ is the total interval length and γ is the maximum degree of the corresponding overlap graph.*

The running time in Theorem 12 implies that both the max-weight k-overlap set problem and the equivalent k-BDMWIS problem for circle (overlap) graphs are in XP.[2] This fact alone can alternatively be derived from a metatheorem of Fomin et al. [6] as follows.[3] The number of minimal separators of circle graphs can be polynomially bounded by $O(n^2)$ as shown by Kloks [14]. Further, since we are interested in a bounded-degree induced subgraph $G[V']$ of a circle graph G, we know from Gaspers et al. [9] that $G[V']$ has treewidth at most four times the maximum degree k. With these two pre-conditions the metatheorem of Fomin et al. [6] yields the existence of an XP-time algorithm for k-BDMWIS on circle graphs. However, the running time obtained from Fomin et al. [6] is $O(|\Pi_G| \cdot n^{t+4} \cdot f(t,\phi))$ where $|\Pi_G|$ is the number of potential cliques in G, t is the treewidth of $G[V']$ with $V' \subseteq V$ being the solution set, and f is a tower function depending only on t and the CMSO (Counting Monadic Second Order Logic) formula ϕ (compare Thomas [22] proving this already for MSO formulas). Let k be the desired degree of a k-BDMWIS instance, then the treewidth of $G[V']$ is at most $4k$. Further by Kloks [14] we know $|\Pi_G| = O(n^2)$. Hence the running-time of the algorithm would be in $O(n^{4k+6} \cdot f(4k,\phi))$, whereas our problem-specific algorithm has running time $O(n^{2k+2})$.

5 Experiments

We implemented the algorithm for 1-BDMWIS from Section 4.3 and the independent set algorithm for 0-BDMWIS in C++. The compiler was g++, version 7.2.0 with set -O3 flag.

[2] The class XP contains problems that can be solved in time $O(n^{f(k)})$, where n is the input size, k is a parameter, and f is a computable function.
[3] We thank an anonymous reviewer of an earlier version for pointing us to this fact.



(a) Density vs the percentage of saved crossings. **(b)** Density vs the computation time.

Figure 6 Plots for the test set with 4881 graphs. w is the weight given to the edges of the circle graph, see Section 2.1. This test set has between 20 and 60 vertices and an average density of 2.6.

Further we used the OGDF library [4] in its most current snapshot version. Experiments were run on a standard desktop computer with an eight core Intel i7-6700 CPU clocked at 3.4 GHz and 16 GB RAM, running Archlinux and kernel version 4.13.12. The implementation is available under `https://www.ac.tuwien.ac.at/two-sided-layouts/`.

We generated two sets of random biconnected graphs using OGDF. The first set has 5,156 and the second 4,822 different non-planar graphs. We varied the edge to vertex ratio between 1.0 and 5.0 and the number of vertices between 20 and 60. In addition, we used the Rome graph library (`http://www.graphdrawing.org/data.html`) consisting of 8,504 non-planar graphs with a density of 0.5 to 2.1 and 10 to 100 vertices. The relative sparsity of the test instances is not a drawback, since it is impractical to use circular layouts for visualizing very dense graphs. For dense graphs one would have to apply some form of bundling strategy to reduce edge clutter. Given such a bundled layout one could consider minimizing bundled crossings [1] and adapt our algorithms to the resulting “bundled” circle graph.

For the random test graphs Fig. 6a displays the percentage of crossings saved by the layouts with exterior edges versus the one-sided circular layout implemented in OGDF. Compared to the approach without exterior crossings we find that allowing up to one crossing per edge in the exterior one can save around 11% more crossings on average for the random instances and 7.5% for the Rome graphs. Setting the edge weight in 1-BDMWIS to one (i.e., counting exterior crossings in the optimization) or two (i.e., not counting exterior crossings in the optimization), has (almost) no noticeable effect.

Figure 6b depicts the times needed to compute the layouts for the respective densities. We observe the expected behaviour. The case of $k = 0$ with $O(|\mathcal{E}|^2)$ time is a lot faster as the graphs get more dense. Still for our sparse test instances our algorithm for 1-BDMWIS with $O(|\mathcal{E}|^4)$ time runs sufficiently fast to be used on graphs with up to 60 vertices (82 seconds on average). For additional plots we refer to the full version of the paper.

Our tests show a clear improvement in crossing reduction when going from $k = 0$ to $k = 1$. Of course this comes with a non-negligible runtime increase. For the practically interesting sparse instances, though, 1-BDMWIS can be solved fast enough to be useful in practice.

6 Open questions

The overall hardness of the k-BDMWIS problem on circle graphs, parametrized by just the desired degree k remains open. While we could show NP-hardness, we do not know whether an FPT-algorithm exists or whether the problem is W[1]-hard. In terms of the motivating graph layout problem crossing minimization is known as a major factor for readability. Yet, practical two-sided layout algorithms must also apply suitable vertex-ordering heuristics and they should further take into account the length and actual routing of exterior edges. Edge bundling approaches for both interior and exterior edges promise to further reduce visual clutter, but then bundling and bundled crossing minimization should be considered simultaneously. It would also be interesting to generalize the problem from circular layouts to other layout types, where many but not necessarily all vertices can be fixed on a boundary curve.

References

1. Md Jawaherul Alam, Martin Fink, and Sergey Pupyrev. The bundled crossing number. In Y. Hu and M. Nöllenburg, editors, *Graph Drawing and Network Visualization (GD'16)*, volume 9801 of *LNCS*, pages 399–412. Springer, 2016.
2. Michael Baur and Ulrik Brandes. Crossing reduction in circular layouts. In *Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *LNCS*, pages 332–343. Springer, 2004.
3. Nadja Betzler, Robert Bredereck, Rolf Niedermeier, and Johannes Uhlmann. On bounded-degree vertex deletion parameterized by treewidth. *Discrete Applied Mathematics*, 160(1–2):53–60, 2012.
4. Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (OGDF). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 17, pages 543–569. CRC Press, 2013.
5. Anders Dessmark, Klaus Jansen, and Andrzej Lingas. The maximum k-dependent and f-dependent set problem. In K. W. Ng, P. Raghavan, N. V. Balasubramanian, and F. Y. L. Chin, editors, *Algorithms and Computation (ISAAC'93)*, volume 762 of *LNCS*, pages 88–97. Springer, 1993.
6. Fedor V Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM J. Computing*, 44(1):54–87, 2015.
7. Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On Structural Parameterizations of the Bounded-Degree Vertex Deletion Problem. In *Theoretical Aspects of Computer Science (STACS'18)*, volume 96 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:14, 2018.
8. Emden R Gansner and Yehuda Koren. Improved circular layouts. In *Graph Drawing (GD'06)*, volume 4372 of *LNCS*, pages 386–398. Springer, 2007.
9. Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Ioan Todinca. Exponential time algorithms for the minimum dominating set problem on some graph classes. *ACM Trans. Algorithms*, 6(1):9:1–9:21, 2009.
10. Fanika Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973.
11. Michael Jünger and Petra Mutzel, editors. *Graph Drawing Software*. Mathematics and Visualization. Springer, 2004.
12. J Mark Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51–63, 1993.

13 Jonathan Klawitter, Tamara Mchedlidze, and Martin Nöllenburg. Experimental evaluation of book drawing algorithms. In F. Frati and K.-L. Ma, editors, *Graph Drawing and Network Visualization (GD'17)*, volume 10692 of *LNCS*, pages 224–238. Springer, 2018. URL: `1708.09221`.

14 Ton Kloks. Treewidth of circle graphs. *International J. Foundations of Computer Science*, 7(02):111–120, 1996.

15 Stephen G. Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *CoRR*, abs/1703.02261, 2017. `arXiv:1703.02261`.

16 Martin I. Krzywinski, Jacqueline E. Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J. Jones, and Marco A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19:1639–1645, 2009.

17 Giuseppe Liotta. Graph drawing beyond planarity: some results and open problems. In *Italian Conference on Theoretical Computer Science (ICTCS'14)*, pages 3–8, 2014.

18 Sumio Masuda, Toshinobu Kashiwabara, Kazuo Nakajima, and Toshio Fujisawa. On the NP-completeness of a computer network layout problem. In *Circuits and Systems (ISCAS'87)*, pages 292–295. IEEE, 1987.

19 Sumio Masuda, Kazuo Nakajima, Toshinobu Kashiwabara, and Toshio Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Computers*, 39(1):124–127, 1990.

20 Farhad Shahrokhi, László A. Székely, Ondrej Sýkora, and Imrich Vrt'o. The book crossing number of a graph. *J. Graph Theory*, 21(4):413–424, 1996.

21 Janet M. Six and Ioannis G. Tollis. Circular drawing algorithms. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 9, pages 285–315. CRC Press, 2013.

22 Wolfgang Thomas. Languages, automata, and logic. *Handbook of formal languages*, 3:389–455, 1996.

23 Gabriel Valiente. A new simple algorithm for the maximum-weight independent set problem on circle graphs. In *Algorithms and Computation (ISAAC'03)*, volume 2906 of *LNCS*, pages 129–137. Springer, 2003.

24 Mihalis Yannakakis. Node- and edge-deletion NP-complete problems. In *Theory of Computing (STOC'78)*, pages 253–264. ACM, 1978.