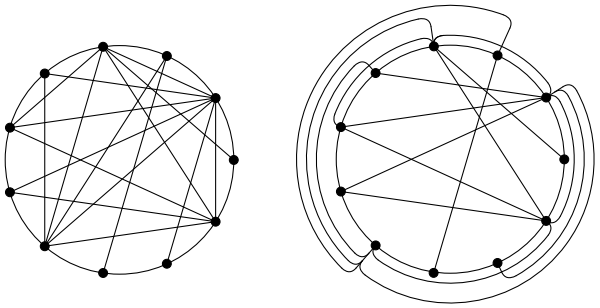
Minimizing crossings in constrained two-sided circular graph layouts

Fabian Klute* Martin Nöllenburg*

Abstract

Circular layouts are a popular graph drawing style, where vertices are placed on a circle and edges are drawn as straight chords. One way to reduce clutter caused by edge crossings is to use *two-sided circular layouts*, in which some edges are drawn as curves in the exterior of the circle. We study the problem of minimizing the crossings for a fixed cyclic vertex order by computing an optimal 1-plane set of exteriorly drawn edges. This relates to finding maximum-weight degree-constrained induced subgraphs in circle or overlap graphs.



(a) One-sided layout (b) Two-sided layout

Figure 1: Circular graph layouts

1 Introduction

Circular graph layouts are a popular drawing style to visualize graphs, which focuses on a clear positioning of the vertices on a circle, while the edges are drawn as straight-line chords of said circle. As it is often the case in graph drawing the crossings between the edges play a big role in optimizing the readability of the visualization. If the edges are drawn as chords all crossings are determined by the order of the vertices. Finding a vertex order that minimizes the crossings is NP-hard [4]. Heuristics and approximation algorithms have been studied in numerous papers, see e.g. [1].

Gansner and Koren [2] 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 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, and the third step introduces edge bundling to save ink and reduce clutter in the interior.

Inspired by their approach we take a closer look at the second step of the above process, which, in other words, determines an outerplane subgraph to be drawn outside the circle such that the remaining crossings of the chords are minimized. Gansner and Koren [2] 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 $G = (V, E)$ (see Section 2). This graph has a vertex for each edge of $\mathcal{G}$ and an edge between each pair of crossing chords in the circular layout of $\mathcal{G}$. The maximum independent set problem in a circle graph can be solved in $O(\ell)$ time [6], where $\ell \in \Omega(|\mathcal{E}|) \cap O(|\mathcal{E}|^2)$ is the total chord length in an interval representation of G.

We generalize the problem from outerplane graphs to outer k-plane graphs, 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. For $k = 0$ this is the same problem considered by Gansner and Koren [2]. In this paper we present an efficient algorithm based on dynamic programming for the case $k = 1$, where at most one crossing per exterior edge is permitted. Of course, this is only a first step towards solving the general case. Yet, due to non-local dependencies that occur for $k \geq 2$, we do not see an obvious way of extending our algorithm.

2 Optimizing interior crossings

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph and π a cyclic order of $\mathcal{V}$. We arrange the vertices in this order on a circle C and draw edges as straight chords to obtain a (one-sided) *circular drawing* Γ, see Fig. 1a. Note that the number of crossings of Γ is fully determined by π. Our goal in this paper is to draw a subset of edges on the exterior side of C in order to reduce the number of edge crossings.

In a *two-sided circular drawing* Δ of $\mathcal{G}$ and π we still draw all vertices on a circle C according to 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 curves in the exterior of C, see Fig. 1b. Rather than asking for a set $\mathcal{E}_2$ that globally minimizes the crossings in Δ, which is equivalent to the NP-hard fixed linear crossing

*Algorithms and Complexity Group, TU Wien, Austria

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

minimization problem in 2-page book embeddings [5], 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 exterior edges are harder to read and should not be further impaired by too many crossings.

Instead of working with $\mathcal{G}$ and π directly we consider the corresponding *circle graph* $G = G_{\mathcal{G},\pi} = (V, E)$ of $\mathcal{G}$ and π, where V 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 edges corresponding to u and v cross in the circular layout Γ. So the number of vertices $|V| = |\mathcal{E}|$ equals the number of edges of $\mathcal{G}$ and the number of edges $|E|$ is the number of crossings of Γ. We further assign to every vertex $v \in V$ a weight $w(v) \in \mathbb{R}^+$ and to every edge $(u, v) \in E$ a weight $w(u, v) \in \mathbb{R}^+$.

Finding the set $\mathcal{E}_2$ can now be modeled as a constrained maximum induced subgraph problem on the circle graph G. The general problem can be stated as follows.

Definition 1 *(Max-Weight Deg-k Induced Subgraph) Given a weighted graph $G = (V, E)$ and $k \in \mathbb{N}$ find a set $V' \subset V$ such that the induced subgraph $G[V'] = (V', E')$ has maximum degree k and maximizes the sum*

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

For general graphs it is NP-hard [7] to find such a subgraph, but restricting the graph class of G to circle graphs makes the problem significantly easier as we will show in Section 3.

It remains to model our constrained crossing minimization problem for two-sided circular layouts as an instance of Max-Weight Deg-k Induced Subgraph. We define the weights of G as $w(v) = \deg(v)$ for all $v \in V$ and as $w(u, v) = 1$ or, alternatively, as $w(u, v) = 2$ for all $(u, v) \in E$, depending on the type of crossings to minimize.

Lemma 1 *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 G induces an outer k-plane graph in Γ that minimizes the number of crossings in the corresponding two-sided layout Δ.*

Proof. Let $V^* \subset V$ be a vertex set that induces a maximum-weight degree-k subgraph in $G = G_{\mathcal{G},\pi}$. 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 circular layout Γ. Hence each edge in $G[V^*]$ corresponds to an exterior crossing in Δ. Since the maximum degree of $G[V^*]$ 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 E' in the exterior part of Δ. However, if two vertices in V^* are connected by an edge, their corresponding edges in Δ cross in the exterior part of Δ and we need to add a correction term. 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 $w(u, v) = 2$, the weight W equals the number of crossings that are removed from the interior, but excluding those that are simply shifted to the exterior of Δ. □

3 Efficient Algorithm based on Overlap Graphs

In this section we use the known connection between circle graphs and overlap graphs, which are subgraphs of interval graphs, to design an efficient algorithm for our crossing minimization problem.

The key concept is to distinguish proper and non-proper overlaps of intervals. Let $\mathcal{I}$ be a set of intervals with distinct endpoints. For two intervals $I = [a, b], J = [c, d] \in \mathcal{I}$ we say that they overlap *properly* if either $a < c < b < d$ or $c < a < d < b$. We say that I *nests* J if $a < c < d < b$. Obviously, if two intervals intersect, they either overlap properly or one nests the other. For an interval $I \in \mathcal{I}$ we define the set $\mathcal{P}(I, \mathcal{I}) = \{J \mid J \in \mathcal{I} \text{ and } I \text{ properly overlaps } J\}$. By $\mathcal{P} = \cup_{I \in \mathcal{I}} \{(I, J) \mid J \in \mathcal{P}(I, \mathcal{I})\}$ we denote the set of all properly overlapping interval pairs of $\mathcal{I}$. Likewise we define $\mathcal{N}(I, \mathcal{I}) = \{J \mid J \in \mathcal{I} \text{ and } I \text{ nests } J\}$.

Given a set of intervals $\mathcal{I}$ we define the *overlap graph* of $\mathcal{I}$ as the graph $G_{\mathcal{I}} = (V, E)$ that has a vertex for each interval in $\mathcal{I}$ and an edge for each pair of properly overlapping intervals. In contrast to interval graphs, two nested intervals do not define an edge in the overlap graph.

Let ρ be the maximum degree of the overlap graph of $\mathcal{I}$ and let $\delta = \max\{|\mathcal{N}(I, \mathcal{I})| \mid I \in \mathcal{I}\}$ be the maximum number of intervals nested by any interval in $\mathcal{I}$. We define the parameter $\gamma = \max\{\rho, \delta\}$ as an upper bound on the number of intersections per interval.

As shown by Gavril [3] circle graphs and overlap graphs are isomorphic. The idea is to cut the circle C between two arbitrary vertices and project the chords onto the real line below C. Each chord is then represented by an interval and two chords intersect if and only if their projected intervals overlap properly, see Figure 2.

We rephrase Definition 1 in terms of interval representations of circle graphs. The weights can be taken directly from the circle graph $G_{\mathcal{G},\pi} = (V, E)$. For each interval $I \in \mathcal{I}$ corresponding to $v \in V$ we set $w(I) = w(v)$ and for each pair of properly overlapping intervals $(I, J) \in \mathcal{P}(\mathcal{I})$ corresponding to edge $e \in E$ we set $w(I, J) = w(e)$.

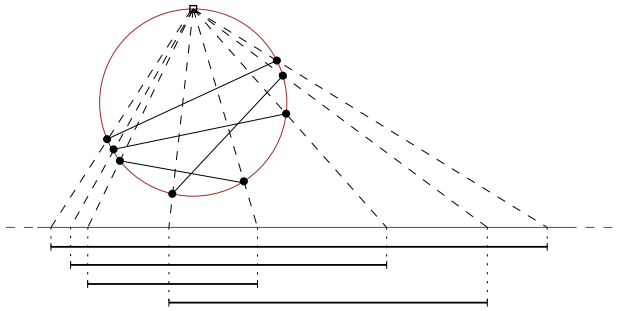


Figure 2: An example projection of a circle graph to a set of intervals with an isomorphic overlap graph.

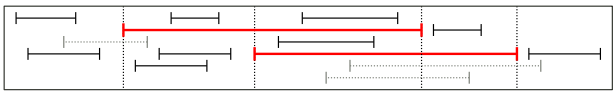


Figure 3: Split along the two red intervals. The dotted intervals are discarded and we recurse on the five sets with black intervals.

Definition 2 *(Max-Weight k-Intersection Set) Given an interval set $\mathcal{I}$ find a subset $\mathcal{I}' \subseteq \mathcal{I}$ such that no interval $I \in \mathcal{I}'$ has more than k proper intersections with other intervals in $\mathcal{I}'$ and the sum*

$$W = \sum_{I \in \mathcal{I}'} w(I) - \sum_{(I,J) \in P(\mathcal{I}')} w(I,J)$$

is maximized.

Since circle graphs and overlap graphs are isomorphic, we can also solve Max-Weight k-Intersection Set in order to solve our crossing minimization problem for two-sided circular layouts. In this paper, we restrict our attention to the case $k = 1$, i.e., finding an outer 1-plane edge set E' or, equivalently, finding an interval set with at most one proper intersection per interval.

3.1 Properties of max-weight 1-intersection sets

Before we describe our algorithm in Section 3.2 we introduce some notation and properties for splitting an interval set into subsets. Let $\mathcal{I}$ be a set of intervals. We say $\mathcal{I}$ has *common point* $x \in \mathbb{R}$ if $x \in I$ for all intervals $I \in \mathcal{I}$. For a general set of intervals $\mathcal{I}$ we define $\mathcal{I}|x = \{I \in \mathcal{I} \mid x \in I\}$ as the set of all intervals in $\mathcal{I}$ with common point x.

Further, for $x, y \in \mathbb{R} \cup \{\pm\infty\}$ with $x \leq y$ we define the set $\mathcal{I}[x,y] = \{I \in \mathcal{I} \mid I \subseteq [x,y]\}$. For any $x \leq y$ an interval set $\mathcal{I}[x,y]$ can be *split along* an interval $I = [a,b] \in \mathcal{I}$ into the three sets $\mathcal{I}[x,a]$, $\mathcal{I}[a,b]$, $\mathcal{I}[b,y]$. All intervals which are not contained in one of the three sets are discarded.

Finally we can split any $\mathcal{I}[x,y]$ along a pair of proper intersecting intervals $I = [a,b], J = [c,d] \in \mathcal{I}$. Without loss of generality let $a < c < b < d$. Then the split creates the five sets $\mathcal{I}[x,a]$, $\mathcal{I}[a,c]$, $\mathcal{I}[c,b]$, $\mathcal{I}[b,d]$, $\mathcal{I}[d,y]$. Again, all intervals which are not contained in one of the five sets are discarded. An example is shown in Figure 3.

Lemma 2 *Let $\mathcal{I}$ be a set of intervals. For any $x \in \mathbb{R}$ at most two properly intersecting intervals $I, J \in \mathcal{I}|x$ can be part of a max-weight 1-intersection set on $\mathcal{I}$.*

Proof. Assume there is a third interval $K \in \mathcal{I}|x$ in a max-weight 1-intersection set, which properly overlaps I or J or both. This K cannot be added to the solution set without creating at least one interval with more than one intersection, which is not allowed by definition. □

For an interval set $\mathcal{I}$ we call $I = [a,b] \in \mathcal{I}$ the *left-most* interval, if $a < a'$ for all $[a',b'] \in \mathcal{I} \setminus I$. We define the *left interval set* as $\mathscr{L}(\mathcal{I}) = P(I,\mathcal{I}) \cup N(I,\mathcal{I})$, the set of intervals intersecting the left-most interval I.

Lemma 3 *Let $\mathcal{I}$ be an interval set, $\mathcal{I}_o$ a max-weight 1-intersection set of $\mathcal{I}$ and $I \in \mathcal{I}$ the left-most interval. Then either $I \in \mathcal{I}_o$ or there exists at least one interval $J \in \mathcal{I}_o$ such that $J \in \mathscr{L}(\mathcal{I})$.*

Proof. Let $\mathcal{I}'_o$ be a max-weight 1-intersection set of $\mathcal{I}$ such that neither I nor an interval $J \in \mathscr{L}(\mathcal{I})$ is part of $\mathcal{I}'_o$. That is, there is no interval $K \in \mathcal{I}'_o$ that properly intersects I or is nested by I, but then $\mathcal{I}'_o \cup \{I\}$ is a solution to the max-weight 1-intersection set problem on $\mathcal{I}$ with larger weight which contradicts the optimality of $\mathcal{I}'_o$. □

3.2 Algorithm for the max-weight 1-intersection set problem

We use a dynamic programming algorithm to solve the max-weight 1-intersection set problem. The principal idea is to split a set of intervals $\mathcal{I}$ in each step along one interval or two properly intersecting intervals into smaller independent subsets. By Lemma 3 we do not have to consider splits along arbitrary intervals, but can choose either single intervals from $\mathscr{L}(\mathcal{I})$ or pairs of properly intersecting intervals, where at least one of them is in $\mathscr{L}(\mathcal{I})$.

We define a two-dimensional table T, in which we store the weight of an optimal local solution for each subinstance $\mathcal{I}[x,y]$ as the entry $T[x,y]$. Since for all relevant splits x and y are start- or end-points of intervals in $\mathcal{I}$ this table has size quadratic in the number of intervals. The best global solution corresponds to entry $T[-\infty,\infty]$, where $\pm\infty$ are symbolic dummy coordinates.

Picking one Interval A single interval $I = [a,b] \in \mathscr{L}(\mathcal{I}[x,y])$ is chosen as a candidate for the optimal solution. This gives us a split along one interval and three subinstances to consider. The optimal solution

$T_1[x,y]$ between x and y, when splitting along one interval I, is the maximum across these splits plus the weight of the interval I.

$$T_1[x,y] = \max_{I \in \mathscr{L}(\mathcal{I}[x,y])} \big\{T[x,a]+T[a,b]+T[b,y]+w(I)\big\}.$$

Since we consider every interval in $\mathscr{L}(\mathcal{I}[x,y])$ this step maximizes over $O(\gamma)$ sub-cases, one for each interval in $\mathscr{L}(\mathcal{I}[x,y])$ which has size at most $\gamma = \max\{\delta,\rho\}$.

Picking two Intervals Two properly intersecting intervals $I = [a,b] \in \mathscr{L}(\mathcal{I}[x,y])$ and $J = [c,d] \in \mathcal{P}(I,\mathcal{I}[x,y])$ are chosen as candidates for the optimal solution. This gives us a split along two intervals and five subinstances to consider. The optimal solution $T_2[x,y]$ for the set $\mathcal{I}[x,y]$ is the maximum across the possible splits generated by pairs of properly intersecting intervals. The weight of an individual split is the weight of the optimal solutions of the generated subinstances plus the weight of the two chosen intervals I, J minus the weight attributed to the pair (I,J).

$$T_2[x,y] = \max_{\substack{I \in \mathscr{L}(\mathcal{I}[x,y]) \\ J \in \mathcal{P}(I,\mathcal{I}[x,y])}} \big\{T[x,a] + T[a,c] + T[c,b] + T[b,d] + T[d,y] + w(I) + w(J) - w(I,J)\big\}.$$

Since we consider every pair of properly intersecting intervals, one of which in $\mathscr{L}(\mathcal{I}[x,y])$, this case maximizes over $O(\gamma^2)$ sub-cases.

Maximizing over both possibilities for the split we obtain the optimal local solution $T[x,y]$ as

$$T[x,y] = \max\{T_1[x,y], T_2[x,y]\}. \tag{1}$$

The set of intervals forming an optimal solution of the max-weight 1-intersection set problem can be recovered using the standard process of backtracking the decisions made by the maximization steps.

Theorem 4 *The* Max-Weight 1-Intersection Set *problem for a set of intervals $\mathcal{I}$ can be solved in $O(\gamma^2 n^2)$ time, where $n = |\mathcal{I}|$ and γ is an upper bound on the number of intersections per interval.*

Proof. The time to compute an entry $T[x,y]$ is dominated by the case of splitting along a pair of intervals, which requires $O(\gamma^2)$ time as argued above. Since T has size $O(n^2)$, the total computation time is $O(\gamma^2 n^2)$.

It remains to show the correctness. Let $\mathcal{I}_o$ be an optimal solution to the max-weight 1-intersection set problem on $\mathcal{I}$. By definition $\mathcal{I}_o$ can be decomposed into pairs $I, J \in \mathcal{I}_o$ such that I and J intersect properly, and single intervals $K \in \mathcal{I}_o$ such that no other interval in $\mathcal{I}_o$ overlaps K properly.

The proof is by induction over the number of intervals in $\mathcal{I}_o$. In case $\mathcal{I}_o$ consists of a single interval or two properly intersecting intervals these have to be in $\mathscr{L}(\mathcal{I})$ by Lemma 3. Our algorithm considers exactly all single intervals and pairs of properly intersecting intervals of $\mathscr{L}(\mathcal{I})$ in its first step, in particular it consider the intervals in $\mathcal{I}_o$.

So let $\mathcal{I}_o$ be an optimal solution with more than two intervals or two single intervals. By Lemma 3, $\mathcal{I}_o$ must contain a single interval from $\mathscr{L}(\mathcal{I})$ or a pair with one interval from $\mathscr{L}(\mathcal{I})$ along which we can split $\mathcal{I}_o$. With this split we create either three or five smaller and independent subinstances. For each subinstance we can compute an optimal solution by induction hypothesis. Since our algorithm also considers that particular split, our solution is at least as good as $\mathcal{I}_o$. In the end we return exactly $\mathcal{I}_o$ or a solution with equal weight. □

Combining the results from above we can conclude with the following result on two-sided layouts.

Corollary 1 *Given a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ and a cyclic vertex order π a crossing-minimal two-sided drawing Δ with outer 1-plane exterior edge set can be computed in $O(\gamma^2|\mathcal{E}|^2)$ time, where γ is the thickness of the overlap graph derived from the circle graph $G_{\mathcal{G},\pi}$.*

References

[1] M. Baur and U. Brandes. Crossing reduction in circular layouts. In *Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *LNCS*, pages 332–343. Springer Berlin Heidelberg, 2004.

[2] E. R. Gansner and Y. Koren. Improved circular layouts. In *Graph Drawing (GD'06)*, volume 4372 of *LNCS*, pages 386–398. Springer, 2007.

[3] F. Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973.

[4] S. Masuda, T. Kashiwabara, K. Nakajima, and T. Fujisawa. On the NP-completeness of a computer network layout problem. In *Circuits and Systems (ISCAS'87)*, pages 292–295. IEEE, 1987.

[5] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Computers*, 39(1):124–127, 1990.

[6] G. 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.

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