Fixed-Location Circular-Arc Drawing of Planar Graphs*

(Regular Paper)

Alon Efrat, Cesim Erten, and Stephen G. Kobourov

Department of Computer Science
University of Arizona
{alon,cesim,kobourov}@cs.arizona.edu

Abstract. In this paper we consider the problem of drawing a planar graph using circular-arcs as edges and given a one-to-one mapping between the vertices of the graph and a set of n points on the plane, where n is the number of vertices in the graph. If for every edge we have only two possible circular arcs, then there a simple reduction to 2SAT yields an $O(n^2)$ algorithm to find out if a drawing with no crossings can be realized. We present an improved $O(n^{7/4} polylog\, n)$ time algorithm. For the special case where the possible circular arcs for each edge are of the same length, we present an even more efficient algorithm that runs in $O(n^{3/2} polylog\, n)$ time. We also consider the problem if we have more than two possible circular arcs per edge and show that the problem becomes NP-Hard. Moreover, we show that two optimization versions of the problem are also NP-Hard.

1 Introduction

A natural question that arises in graph drawing is whether a graph with fixed vertices can be drawn without crossings, when several choices are given for each of the edges. From an information visualization point of view convex edges are preferable, i.e., straight line segments or circular arcs. In general, embedding a planar graph at fixed locations and drawing it with straight lines may result in many crossings. Using circular arcs instead can reduce or eliminate the crossings; see Fig. 1(a). Thus, a natural problem to consider is whether a given graph with fixed vertex locations can be drawn without crossings, using circular arcs.

We first consider the *2-Circle Drawing (2CD)* problem, in which each edge has to be drawn as one of the two circular arcs defined by a circle passing through the endpoints. This problem is reminiscent of the *Manhattan wiring problem*: Consider the axis-aligned rectangle with a diagonal defined by the line segment between two vertices connected by an edge. Then the two semi-rectangles separated by the diagonal are the two choices for drawing the edge. This formulation of the problem can be efficiently solved in $O(n \log n)$ time, using an efficient *find and delete* data structure (to find intersections between a pair of semi-rectangles and to delete a semi-rectangle from the data structure) [14].

* This work was partially supported by the NSF under grant ACR-02229290.

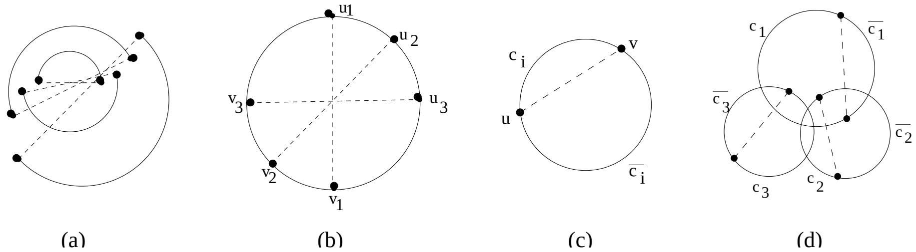

Fig. 1. (a) $\Omega(n^2)$ crossings with straight-line edges and none with half-circles; (b) A planar graph that cannot be drawn without crossings using *any* circular arc segments. (c) Given the circle C_i, edge $e_i = (u, v)$ is drawn either with the circular arc c_i or $\overline{c_i}$. (d) An example of a 2SAT reduction: $(\overline{c_1} \vee c_3) \wedge (\overline{c_1} \vee \overline{c_3}) \wedge (\overline{c_3} \vee \overline{c_2}) \wedge (\overline{c_1} \vee \overline{c_2}) \wedge (c_1 \vee c_2)$.

The same approach cannot be applied to the 2CD problem directly. Although efficient data structures exist for operations on full circles, no such data structures exist in the case of circular arcs (semi-circles). The novelty of our 2CD algorithm is that we provide a way to use an efficient data structure for full circles to solve the problem with circular arcs. For the sake of completeness, we first show that the 2CD problem can be reduced to 2SAT and thus solved in $O(n^2)$ time. Then we provide our novel approach to solve the 2CD problem in time $O(n^{7/4} polylog\, n)$ for the general case and in time $O(n^{3/2} polylog\, n)$ for the restricted case where the circular arcs are exactly half-circles. Although the practical gain in terms of running time is not significant, we believe that our approach for solving the 2CD problem might be of independent interest to solve similar problems.

Next, we consider the 3-Circle Drawing (3CD) problem, where for each edge there are three circular arcs to choose from. We show that the 3CD problem is NP-hard. Although using circular arcs to represent the edges allows a certain flexibility, not every planar graph can be drawn without crossings using circular arcs. Fig. 1(b) shows an example of a planar graph that cannot be drawn without crossings using *any* circular arc segments. This difficulty suggests two optimization problems: *Min2CD* is the problem of minimizing the number of crossings for a given 2CD instance by representing *every* edge with an appropriate circular arc. *Max2CD* is the problem of maximizing the number of edges that can be drawn without crossings using circular arcs. We show that both of these optimization problems are NP-hard.

2 Previous Work

Several variations of the problem of embedding a planar graph at fixed point locations have been studied. If we can choose the mapping between the vertices V and the points P, then Kaufmann and Wiese [15] show that the graph can be drawn without crossings using 2 bends per edge in polynomial time. However, if the mapping between V and P is given, Pach and Wenger [19] show that $O(n)$ bends per edge are necessary to guarantee planarity, where n is the number of vertices in the graph. Godau [12] shows that if each vertex is allowed to move slightly in the neighborhood of a fixed point then the problem becomes NP-hard.

Drawing graphs with circular arcs with no assigned vertex locations has been considered by Cheng *et al* [6] in the context of planarity, angular resolution and drawing area. The problems under consideration in this paper are also related to the *k-position point labeling problem*, extensively studied in *map labeling* [4, 21]. In the k-position point labeling problem we are given a set of points and a set of k possible label positions for each point and we would like to find a labeling of the points that optimizes specific criteria. Criteria such as maximizing the number of labeled points [3, 23] and maximizing the size of the labels [7, 8, 11] have been considered. A variant of the map labeling problem is reduced to 2SAT and NP-Completeness results are presented in [11]. Although these problems are related to drawing planar graphs with circular arcs, there is a significant difference: whereas map labeling problems are restricted by region intersections, drawing planar graphs with circular arcs is restricted by circular arc intersections.

3 Circular Arcs Drawing: 2CD

The input to the problem is a planar graph $G = (V, E)$, a point set P, and a one-to-one function $f : V \to P$ such that $|V| = |P| = n$ and $|E| = m$. Note that $m = O(n)$ as G is a planar graph. For any edge $e_i = (u, v) \in E$, we are given some circle C_i that passes through u and v. The two vertices, u and v, determine two circular arcs on C_i; let c_i and $\overline{c_i}$ be their labels, see Fig 1(c). We would like to find out whether G can be drawn without crossings using c_i or $\overline{c_i}$ for each e_i, and if so to provide such a drawing of G.

We first suggest a straightforward solution of the problem using a reduction from 2CD to 2SAT. The reduction to a 2SAT formula Φ requires that we identify all intersections between circular arcs. For each such intersection $c_i \cap c_j \neq \emptyset$, we add the formula $(\overline{c_i} \vee \overline{c_j})$ to Φ, see Fig 1(d). Since there are $O(n^2)$ crossings, the reduction takes $O(n^2)$ time and results in a 2SAT formula Φ with $O(n)$ variables and $O(n^2)$ clauses. It is easy to see that G can be drawn without intersections if and only if the corresponding formula Φ is satisfiable. Since 2SAT can be solved in time linear in the number of clauses and variables [5, 10], we have an $O(n^2)$ time algorithm for the 2CD problem. However we can do much better.

3.1 The 2CD Algorithm

Note that for a given edge $e_i = (u, v) \in E$, as possible drawings of e_i we consider the circular arcs defined by the circle C_i, and the position of u and v on C_i. Alternatively, we could consider the axis-aligned rectangle having its diagonal as the line segment (u, v), and then consider the 2 semi-rectangles separated by the diagonal as two choices for drawings of the edge e_i. This formulation of the problem is known as the *Manhattan wiring problem* which can be efficiently solved in $O(n \log n)$ time [14].

We describe a new algorithm that solves a more general problem for circular arcs. Our approach is different from the Manhattan wiring problem in that we perform operations only on complete circles. More formally, we *find/delete* complete circles as part of an intersection, as opposed to performing the operations on the circular arcs directly. Let $\mathcal{D}$ denote the data structure used to store all the circles. Let $find(\mathcal{D}, c_i)$ be a function that finds a circle C_j intersecting circular

Algorithm 2CD	**Traverse_possible(c_i)**
while $\mathcal{D}$ not empty	**while** $C_j = find(\mathcal{D}, c_i)$ not empty
let C_i be a circle in $\mathcal{D}$	$delete(\mathcal{D}, C_j)$
$delete(\mathcal{D}, C_i)$	let c_j be involved in the intersection
delete $\overline{c_i}$ from $\mathcal{P}$	*delete* c_j from $\mathcal{P}$
mark c_i with its *dfsnumber*	*mark* $\overline{c_j}$ with its *dfsnumber*
$traverse\_possible(c_i)$	$traverse\_possible(\overline{c_j})$
start with initial data structure $\mathcal{D}$	
while $\exists c_i, c_j \in P$ s.t. $c_i \cap c_j \neq \emptyset$	
let $dfsnumber(c_i) < dfsnumber(c_j)$	**Traverse_certain(c_i)**
$delete(\mathcal{D}, C_i)$	**while** $C_j = find(\mathcal{D}, c_i)$ not empty
delete c_i and $\overline{c_i}$ from $\mathcal{P}$	$delete(\mathcal{D}, C_j)$
add $\overline{c_i}$ into $\mathcal{C}$	let c_j be involved in the intersection
$traverse\_certain(\overline{c_i})$	*delete* c_j and $\overline{c_j}$ from $\mathcal{P}$
if $\exists$ intersecting half-circles in $\mathcal{C}$	*add* $\overline{c_j}$ into $\mathcal{C}$
output No	$traverse\_certain(\overline{c_j})$
else output $\mathcal{C} \cup \mathcal{P}$	

Fig. 2. Algorithm 2CD. The input to the algorithm is $\mathcal{D}$, the data structure used to store all the circles. If it is possible to draw the graph without crossings, the algorithm outputs the set of circular arcs used to draw the graph.

arc c_i, and $delete(\mathcal{D}, C_i)$ be a function that deletes the circle C_i from $\mathcal{D}$. Let $\alpha(n), \beta(n)$ denote the time required to perform the *find* and *delete* operations, respectively. We next describe how to construct the data structure $\mathcal{D}$ and how to perform the *find/delete* operations efficiently. The main algorithm is shown on Fig. 2.

Let a *possible* circular arc be one that is not yet chosen and not yet discarded and let $\mathcal{P}$ denote the set of *possible* circular arcs. $\mathcal{P}$ initially contains all the circular arcs. We traverse the circles in a depth-first manner starting with an arbitrary circular arc c_i and making all possible assignments. Each circle is found and deleted exactly once, resulting in a sequence of $O(n)$ *find/delete* operations and requires $O(n \times (\alpha(n) + \beta(n))$ time. At the end of the first *while* loop in the main algorithm, the set $\mathcal{P}$ contains exactly one circular arc for each edge. However, $\mathcal{P}$ might contain intersections. At this point we make the following observation:

Observation 1 *Let $c_i, c_j \in \mathcal{P}$. If $c_i \cap c_j \neq \emptyset$ and $dfsnumber(c_i) < dfsnumber(c_j)$, then the implication $c_i \Rightarrow c_j$ holds, and $\overline{c_i}$ must be included in the final solution.*

Let $\mathcal{C}$ denote the set of *certain* circular arcs. Initially $\mathcal{C}$ is empty. Once we find a circular arc, c_k, that is certainly in the final solution by the above observation, we perform a traversal from c_k, placing all certain arcs in the set $\mathcal{C}$. To find the intersections in $\mathcal{P}$ it suffices to perform a plane sweep over the whole set $\mathcal{P}$. Since, whenever we encounter an intersection in $\mathcal{P}$ we delete the whole circle, the plane sweep over $\mathcal{P}$ finds $O(n)$ intersections in total. Then the second *while* loop requires $O(n \log n + n \times (\alpha(n) + \beta(n)))$ time. Finally we end up with a set $\mathcal{C}$ of certain circular arcs, and a set $\mathcal{P}$ of possible circular arcs. At this point we make a second observation:

Observation 2 *Let $c_i, c_j \in \mathcal{C} \cup \mathcal{P}$. If $c_i \cap c_j \neq \emptyset$, then $c_i, c_j \in \mathcal{C}$.*

The observation holds for the following reasons: Assume $c_i, c_j \in \mathcal{P}$ were true. Then we must have encountered the intersection in the plane sweep step in which case one of them would have been deleted from $\mathcal{P}$. So, $c_i, c_j \in \mathcal{P}$ can not be true. On the other hand assume, $c_i \in \mathcal{C}$ and $c_j \in \mathcal{P}$ were true. Then we must have traversed through c_j before visiting c_i in the traversal step. But when we traverse through a circle, we delete the whole circle from $\mathcal{P}$. So this can not be the case either.

Then we need to concentrate on the intersections in $\mathcal{C}$. We perform a final plane sweep over the set $\mathcal{C}$. If we encounter an intersection, then there cannot be an assignment without intersections, otherwise $\mathcal{C} \cup \mathcal{P}$ gives us a feasible assignment. The running time of the algorithm is the time required for the two *while* loops in the main algorithm: $O(n \log n + n \times (\alpha(n) + \beta(n)))$. In the next section we describe the data structure that supports the needed operations.

3.2 The Data Structure

Given a circular arc query c_i, finding and deleting a circle C_j that intersects c_i is more efficient than performing the same operations on a circular arc c_j that intersects c_i. This observation led us to the 2CD algorithm which assumes the existence of a data structure $\mathcal{D}$ that stores all the circles and allows for efficient *find/delete* operations. Gupta *et al* [13] show how to reduce the problem of querying circles with a circular arc to one of half-space range searching in higher dimensions. The method requires at most a 4-dimensional half-space range searching. To report such intersections then, we make use of the ideas from *geometric range-searching* [1, 2, 18]. The main data structure we use is a partition tree, constructed using the partitioning theorem by Matoušek [17]: a point set P can be partitioned into $O(n^{1-1/d})$ classes in time $O(n \log n^{1-1/d})$ such that for any class P_i, $|P_i| < 2 \times n^{1/d}$ and any line l intersects at most $O(n^{(1-1/d)^2})$ classes, where d is the dimension of the search space. Using this partitioning theorem we can create a data structure $\mathcal{D}'$ that performs half-space range queries in time $O(n^{1-1/d} (\log n)^{O(1)})$. Moreover $\mathcal{D}'$ is dynamic, in the sense that we can delete a circle from $\mathcal{D}'$ in amortized time $O(\log n)$. Then using multiple levels of $\mathcal{D}'$ to satisfy the intersection conditions of [13], we create the data structure $\mathcal{D}$ that supports $find(\mathcal{D}, c_i)$ operations in $O(n^{3/4} \textit{polylog } n)$ time and that requires $O(n \log n)$ time for a sequence of $O(n)$ $delete(\mathcal{D}, C_i)$ operations. These results can be summarized with the following theorem for the 2CD problem.

Theorem 1. *The 2CD problem can be solved in time $O(n^{7/4} \textit{polylog } n)$.*

3.3 Allequal2CD

We can solve a restricted version of the 2CD problem even more efficiently. Let *Allequal2CD* be the version of 2CD where each edge e_i has the same length and the line segment between the endpoints of e_i is the diameter of circle C_i. In this case, for a given edge e_i, the circular arcs c_i and $\overline{c_i}$ are half-circles. We present an

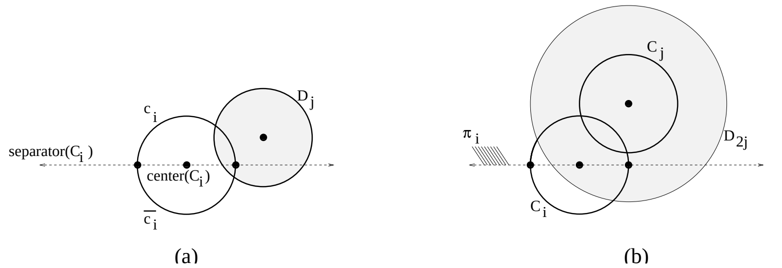

Fig. 3. (a) D_j contains an endpoint of c_i (b) center$(C_j) \in \pi_i$ and center$(C_i) \in D_{2j}$.

algorithm to solve Allequal2CD in $O(n^{3/2} polylog\, n)$ time using a data structure $\mathcal{D}$ that enables us to perform efficient *find/delete* operations. We provide the details for the construction of $\mathcal{D}$ here, since the general data structure described above can be constructed in a similar fashion.

Let $center(C_i)$ and $separator(C_i)$ denote, respectively, the center of C_i and the line separating the half-circles c_i and $\overline{c_i}$. Define π_i as the half-plane bounded by separator(C_i), and that contains c_i. Let D_i be the disk bounded by the circle C_i, and let D_{2i} be the disk concentric to D_i but with radius twice the radius of D_i, see Fig 3. The following lemma is easy to verify.

Lemma 1. *A circle C_j intersects a half-circle c_i if and only if* **(i)** *D_j contains an endpoint of c_i, or* **(ii)** *center$(C_j) \in \pi_i$ and center$(C_i) \in D_{2j}$.*

In order to report intersections of the first type we use the data structure described by Efrat *et al* in [9]: given n equal-sized disks in the plane, construct a data structure $\mathcal{DT}_1$ in time $O(n \log n)$ such that for a given query point p, finding a disk that contains p requires $O(\log n)$ time. Moreover, deleting a disk from $\mathcal{DT}_1$ requires amortized time $O(\log n)$. We preprocess the disks D_i for each i using this structure.

To deal with the intersections of the second type we make use of the partition tree $\mathcal{D}'$ described above. However, this time we perform half-space range searching in 2-dimensions using a two-level data structure. Let the data structure for the second type of intersections be $\mathcal{DT}_2$. The first level of $\mathcal{DT}_2$ is a partition tree, $\mathcal{DT}_2{}'$. Based on the partitioning theorem described above, we partition the centers of all the circles and recursively build the partition tree $\mathcal{DT}_2{}'$. The leaves of $\mathcal{DT}_2{}'$ partition the centers into constant-sized subsets. Each internal node v is associated with a subset P_v of the points contained in the leaves of the subtree rooted at v. We build the second level of the data structure based on these subsets. The second level data structure used in $\mathcal{DT}_2$ is the same as $\mathcal{DT}_1$, except we preprocess the disks D_{2i} for each i, rather than the disks D_i as is the case for constructing $\mathcal{DT}_1$. We call this second level data structure $\mathcal{DT}_2{}''$ to distinguish it from $\mathcal{DT}_1$ which we used to find the first type of intersections. Each internal node v in $\mathcal{DT}_2{}'$ contains a pointer to the corresponding $\mathcal{DT}_2{}''$, where $\mathcal{DT}_2{}''$ contains the data structure for all the disks D_{2i} centered at P_v. The preprocessing time for constructing the partition in a node of $\mathcal{DT}_2{}'$ with m points is $O(m \log m)$. Constructing $\mathcal{DT}_2{}''$ for the same node also takes time $O(m \log m)$. Since the number of points in nodes of $\mathcal{DT}_2{}'$ decreases as a double exponential with their depth in the tree, the total preprocessing time is $O(n \log n)$.

Theorem 2. *Allequal2CD problem can be solved in time $O(n^{3/2} polylog\ n)$.*

Proof Sketch: To find a circle C_j intersecting a given a half-circle c_i we first query $\mathcal{DT}_1$ with c_i's endpoints. This step requires $O(\log n)$ time. If we cannot find such a circle then we query $\mathcal{DT}_2$ with separator(C_i). Upon finding an internal node v such that P_v lies completely above separator(C_i), we query the associated $\mathcal{DT}_2''$ of v with center(C_i). Let $\alpha(n)$ be the time to find a circle intersecting a given half-circle c_i. Then $\alpha(n)$ is bounded by the query time of $\mathcal{DT}_2$ and we get:

$$\alpha(n) \leq O(\sqrt{n}) \times \log 2\sqrt{n} + O(\sqrt[4]{n}) \times \alpha(2\sqrt{n}) \tag{1}$$

Thus the time required to perform a *find* operation is $\alpha(n) = O(\sqrt{n} polylog\ n)$.

In order to delete a circle C_i, we first delete D_i from $\mathcal{DT}_1$ in $O(\log n)$ amortized time. We also need to delete the appropriate disks in $\mathcal{DT}_2$. To do this we simply find each internal node v of $\mathcal{DT}_2'$ such that center$(C_i) \in P_v$, and delete the corresponding disk from $\mathcal{DT}_2''$, the second level data structure pointed to by v. Since $\mathcal{DT}_2'$ has depth $O(\log \log n)$ and deleting a disk from $\mathcal{DT}_2''$ takes amortized time $O(\log n)$, the deletion of a circle takes $O(\log^2 n)$ amortized time. Since *find* and *delete* operations are defined for both $\mathcal{DT}_1$ and $\mathcal{DT}_2$, the two data structures form the complete data structure $\mathcal{D}$ and the theorem follows. $\square$

4 The 3CD Problem

The 3CD problem is similar to 2CD, except now we have three choices for the drawing of each edge $e_i = (u, v)$. We consider the problem in which in addition to the two half-circles we can also choose the line segment connecting u and v. We show that 3CD is NP-hard using a reduction from the NP-hard PLANAR-3SAT [16]. A 3SAT instance Φ is called a PLANAR-3SAT instance if the (bipartite) occurrence graph $G_\Phi = (V_\Phi, E_\Phi)$ is planar. In the occurrence graph, V_Φ contains a vertex for each variable and clause, and E_Φ contains an edge between two vertices $v, w \in V_\Phi$ if v represents a variable x that occurs in the clause represented by w. Let VR3SAT(Variable Restricted 3SAT) be the version of 3SAT with the restriction that each variable can appear at most three times, and VR1IN3SAT be the version of VR3SAT in which *exactly* one literal in each clause is required to be true. In the planar versions of these two problems, the occurrence graphs of the input instances must be planar. We will convert a PLANAR-3SAT instance Φ into a 3CD instance Φ_{3CD} through a series of modifications that preserve planarity.

Lemma 2. *PLANAR-VR3SAT is NP-hard.*

Proof Sketch: The reduction is from PLANAR 3SAT. Let Φ be an input formula for 3SAT. The reduction from 3SAT to VR3SAT converts Φ into Φ', such that if x is a variable appearing k times in Φ, then in Φ' the first occurrence of x is replaced by x_1, the second by x_2, and so on, where x_1, x_2, ..., x_k are k new variables. Φ' also includes the new clauses $(\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge \ldots \wedge (\overline{x_k} \vee x_1)$ [20]. We just need to show that if Φ is planar, then so is Φ'. Let G_Φ, $G_{\Phi'}$ be the occurrence graphs of Φ, Φ' respectively, and v_x be the vertex corresponding to the variable x

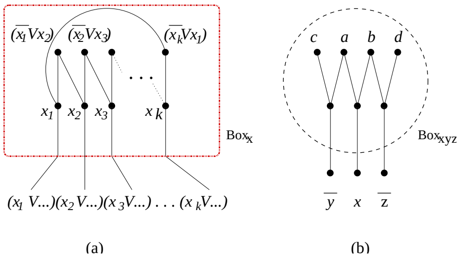

Fig. 4. (a) VR gadget; (b) 1-IN-3 gadget.

of Φ in G_Φ. For constructing $G_{\Phi'}$, we replace v_x with a box_x, where box_x contains the occurrence graph of the newly added clauses $(\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge \ldots \wedge (\overline{x_k} \vee x_1)$ and the newly added variables $x_1, x_2, \ldots, x_k$. It is easy to see that the occurrence graph inside box_x is planar. The vertices corresponding to the clauses containing x can still be incident to the vertices inside box_x without affecting planarity, as shown in Fig. 4(a). This implies that $G_{\Phi'}$ is also planar. □

Lemma 3. *PLANAR-VR1IN3SAT is NP-hard.*

Proof Sketch: The reduction from VR3SAT to VR1IN3SAT converts Φ', an instance of VR3SAT, to Φ'' such that each clause $(x \vee y \vee z)$ is replaced by the clauses $(x \vee a \vee b) \wedge (\overline{y} \vee a \vee c) \wedge (\overline{z} \vee b \vee d)$, and each clause $(x \vee y)$ is replaced by the clauses $(x \vee a) \wedge (\overline{y} \vee a \vee c)$. Let $G_{\Phi'}$ be the occurrence graph of Φ', and v_{xyz} be the vertex corresponding to the clause $(x \vee y \vee z)$ of Φ' in $G_{\Phi'}$. We replace v_{xyz} with a box_{xyz} in $G_{\Phi''}$, where box_{xyz} contains the occurrence graph of the newly added clauses $(x \vee a \vee b) \wedge (\overline{y} \vee a \vee c) \wedge (\overline{z} \vee b \vee d)$ and the newly added variables a, b, c, d. It follows that $G_{\Phi''}$ is also planar, see Fig. 4(b). □

Theorem 3. *The 3CD problem is NP-hard.*

Proof Sketch: We convert a PLANAR-VR1IN3SAT instance Φ'' into a 3CD instance Φ_{3CD}. Because of the VR reduction, the occurrence graph $G_{\Phi''}$ for Φ'' has maximum degree 3. Then there exists an orthogonal drawing for $G_{\Phi''}$ (a drawing such that each vertex is on the integer grid and each edge consists of horizontal and vertical edge segments) and the grid is of size quadratic in the size of $G_{\Phi''}$ [22]. Given the orthogonal drawing of $G_{\Phi''}$, we obtain Φ_{3CD} by the following method: We replace each vertex corresponding to a variable x_i with a *variable-circle*, with one half labeled x_i and the other $\overline{x_i}$, see Fig. 5(a). We replace each vertex corresponding to a clause, say, $(x_i \vee y_j \vee z_k)$ of Φ'', with a *clause-circle* having one half-circle labeled x_i, one labeled y_j, and the diameter of the circle labeled z_k, see Fig. 5(b). We represent each edge of $G_{\Phi''}$ with a *2-link-chain* which consists of two parallel links of *chain-circles*. Let e_i be the edge of $G_{\Phi''}$ between the vertex corresponding to the clause $(x_i \vee y_j \vee z_k)$ and the vertex corresponding to the variable x_i. Then we represent e_i by a 2-link-chain, where one of the links is connected to the half-circle of the clause-circle labeled with x_i on one end, and to the half-circle $\overline{x_i}$ of the variable-circle on the other. The other link intersects with both the half-circle y_j and the diameter

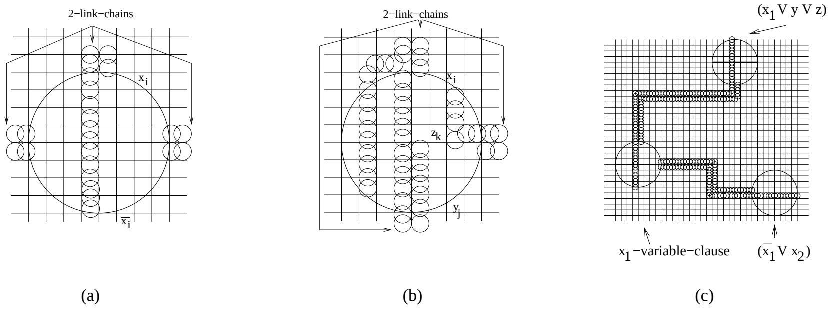

Fig. 5. (a) Variable-circle of x_i; (b) Clause-circle of $(x_i \vee y_j \vee z_k)$; (c) Circle drawing of $(x_1 \vee y \vee z) \wedge (\overline{x_1} \vee x_2) \ldots$. Each edge is represented by 2-link-chains (two parallel chains of small circles). Since the graph is planar there will be no crossings of the chains.

z_k of the clause-circle on one end, and is connected to the half-circle x_i of the variable-circle on the other, see Fig. 5(c).

We claim that Φ'' is satisfiable if and only if Φ_{3CD} has a feasible assignment without crossings. Assume that Φ'' is a satisfiable instance of PLANAR-VR1IN3SAT, and let α be a satisfying assignment. A feasible assignment of edges in Φ_{3CD} is as follows: For each variable-circle corresponding to variable x_i, assign the half-circle labeled with x_i or $\overline{x_i}$ depending on whether x_i is assigned to true or false in α respectively. For each clause-circle corresponding to a clause $(x_i \vee y_j \vee z_k)$, assign the half-circle (or diameter) corresponding to the (only) true literal in the clause, as determined by α. For each 2-link-chain connected to the variable-circle of x_i, if x_i is assigned to true in α, then for the link that is connected to x_i, assign the first chain-circle by choosing the half-circle that *does not* cross the x_i, and continue assigning the chain-circles through the link without creating any crossings. For the 2^{nd} link that is connected to the $\overline{x_i}$ half-circle, assign the first chain-circle by choosing the half-circle that *crosses* the half-circle $\overline{x_i}$, and continue assigning the chain-circles through the link without creating any crossings. This assignment does not contain any crossings. The only crossings that could occur would be between a chain-circle at the tip of a link and a clause-circle, but our method of assigning the chain-circles eliminates this possibility.

For the other direction, assume that Φ_{3CD} has a feasible assignment of edges without crossings. Then, finding a truth assignment α for Φ'' is straightforward: For each variable-circle corresponding to a variable x_i, if the half-circle labeled with x_i is chosen, then assign x_i to be true, otherwise assign it false. This yields a satisfying assignment, since the feasible assignment of edges in Φ_{3CD} chooses exactly one edge from each clause-circle such that there are no conflicts with the variable assignments and the true literal assignment for the other clauses. □

5 Drawing with Few Crossings

If G cannot be drawn without crossings using half-circles, there are two natural optimization problems. Define Min2CD as the following decision problem: Given $(G = (V, E), \kappa_{MIN})$, where G is a planar graph, and κ_{MIN} is a non-negative

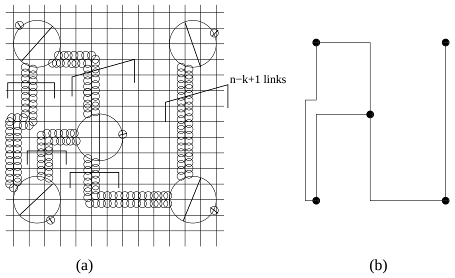

Fig. 6. (a) Min2CD reduction. *Vertex circles* correspond to vertices in H, *chain circles* correspond to edges in H, and *tail circles* are auxiliary circles. (b) The original graph H drawn on an integer grid.

integer, does there exist an assignment of half-circles (either c_i or $\overline{c_i}$) for *each* $e_i \in E$ such that the number of crossings is at most κ_{MIN}? The second optimization problem, Max2CD, is defined as follows: Given $(G = (V, E), \kappa_{MAX})$, where G is a planar graph, and κ_{MAX} is a non-negative integer, does there exist an assignment of half-circles (either c_i or $\overline{c_i}$) for *some* $e_i \in E$ such that there are no crossings and the number of assigned edges is at least κ_{MAX}? We prove that both problems are NP-hard by reductions from the Planar Degree-3 Independent Set problem (PD3IS). Let $H = (V_H, E_H)$ be an undirected graph. We say that a set $I \subseteq V_H$ is independent if for all pairs (i, j), where $i, j \in I$, $(i, j) \notin E_H$. The PD3IS problem is the following: Given $(H = (V_H, E_H), \kappa_{IND})$, where H is a planar graph with maximum degree 3 and κ_{IND} is a non-negative integer, does there exist an independent set I with |I|=κ_{IND}?

Lemma 4. *PD3IS is NP-hard.*

Proof Sketch: Due to space constraints we leave the proof of this lemma in the Appendix. □

Theorem 4. *Min2CD is NP-hard.*

Proof Sketch: Let $(H = (V_H, E_H), \kappa_{IND})$ be an instance of PD3IS. The reduction produces a Min2CD instance (G, κ_{MIN}), with $\kappa_{MIN} = n_H - \kappa_{IND}$. Since H has maximum degree 3, we can find an orthogonal drawing of H, such that each vertex is on the integer grid of size quadratic in the size of H [22], see Fig. 6(b). The reduction scales the grid of H by a factor of $n_H - \kappa_{IND} + 1$ and replaces the vertices of H with *vertex-circles*, circles of diameter $(n_H - \kappa_{IND} + 1)$ units, see the large circles in Fig. 6(a). Each edge of H is represented with $n_H - \kappa_{IND} + 1$ links of *chain-circles*, circles having half a unit diameter connected to a vertex-circle at its *head*. Each vertex-circle also has a *tail circle*, connected to it at its *tail* in such a way that the diameter of the tail-circle crosses the tail of the vertex-circle. Since the given graph H is planar, we can obtain such a grid drawing of circles without causing any intersection between the chain-circles.

We claim that H has an independent set of size κ_{IND} if and only if G can be drawn using half-circles with at most $n_H - \kappa_{IND}$ crossings. Assume H has an independent set I, where $\mid I \mid = \kappa_{IND}$. Then there are κ_{IND} vertices in H that are pairwise disconnected, which further implies that in G there are κ_{IND}

vertex-circles which are not connected by links. Then a feasible assignment of half-circles which allows a drawing of G with at most $n_H - \kappa_{IND}$ crossings follows easily: For each vertex-circle, if the vertex corresponding to it is in I, then assign the head of the vertex-circle, otherwise assign the tail as chosen. This results in an assignment that will have at least κ_{IND} heads, and at most $n_H - \kappa_{IND}$ tails. The chain-circles of the links connected to a vertex-circle which is already assigned to its head are assigned so that no crossing is created, i.e., starting from the chain-circle attached to the already assigned head, choose the half-circle that does not create any crossings. The chain-circles of the other links are assigned edges in a very similar way, but this time with no condition on the assignment of the first chain-circle. Finally the tail-circles are assigned randomly to the half-circles. Such an assignment assigns half-circles for every circle in the drawing, and creates no more than $n_H - \kappa_{IND}$ crossings. The only crossings created are those between the tail-circles and the vertex-circles assigned to their tails. We already know that there are at most $n_H - \kappa_{IND}$ such vertex tails, which implies the first direction of the claim.

For the other direction, assume G has an assignment of half-circles with at most $n_H - \kappa_{IND}$ crossings. Let C_H (C_T) be the sets of vertex-circles having their heads (respectively tails) chosen in this assignment We know that $\mid C_T \mid \leq n_H - \kappa_{IND}$, since otherwise the assignment would create more than $n_H - \kappa_{IND}$ crossings. This implies that $\mid C_H \mid \geq \kappa_{IND}$, since $\mid C_T \mid + \mid C_H \mid = n_H$. For any (c_i, c_j) pair, where c_i, $c_j \in C_H$, there cannot be any links between c_i and c_j because if c_i, c_j were linked together, each of the $n_H - \kappa_{IND} + 1$ links would have at least one crossing, creating more than $n_H - \kappa_{IND}$ crossings which would be a contradiction. Let I be the set of vertices corresponding to the vertex-circles in C_H; then I is an independent set with size at least κ_{IND}. □

Theorem 5. *Max2CD is NP-hard.*

Proof Sketch: The reduction is again from PD3IS. Let $(H = (V_H, E_H), \kappa_{IND})$ be an instance of PD3IS. The reduction produces a Max2CD instance (G, κ_{MAX}), with $\kappa_{MAX} = t_H - n_H + \kappa_{IND}$, where t_H is the total number of circles in G. The proof proceeds along the lines of the proof of Theorem 4 with a slight modification: in this case we add $n_H - \kappa_{IND} + 1$ tail-circles to each vertex-circle, rather than just one tail-circle. Then H has an independent set of size κ_{IND} if and only if G can be drawn without any crossings such that at least $\kappa_{MAX} = t_H - n_H + \kappa_{IND}$ edges have been assigned to some half-circle. □

6 Open Problems

We conclude with two open problems: (1) Can Min2CD and Max2CD be approximated within a constant factor? (2) Can we draw graphs with pre-specified vertex positions, using elliptic or other parabolic curve segments without creating too many crossings?

7 Acknowledgments

We would like to thank Roberto Tamassia, Sue Whitesides, and Helmut Alt for the stimulating discussions and suggestions.

References

1. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, CRC Press, 1997.* 1997.
2. P. Agarwal and J. Erickson. Geometric range searching and its relatives, 1999.
3. P. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. In *9th Canad. Conf. Comp. Geom.*, pages 233–238, 1997.
4. P. K. Agarwal and M. Sharir. Counting circular arc intersections. In *Proceedings of the 7th annual symposium on Computational geometry*, pages 10–20, 1991.
5. B. Apswall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Proc. Letters*, 8:121–123, 1979.
6. C. C. Cheng, C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Drawing planar graphs with circular arcs. *Discrete and Computational Geometry*, 25:405–418, 2001.
7. S. Doddi, M. Marathe, A. Mirzaian, B. Moret, and B. Zhu. Map labeling and its generalizations. In *8th Symposium on Discrete Algorithms*, pages 148–157, 1997.
8. S. Doddi, M. Marathe, and B. Moret. Point set labeling with specified positions. In *Proc. 16th ACM Sympos. Comput. Geom. (SoCG'00)*, pages 182–190, 2000.
9. A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–28, 2001.
10. S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5:691–703, 1976.
11. M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.
12. M. Godau. On the difficulty of embedding planar graphs with inaccuracies. In *Proceedings on Graph Drawing (GD'94)*, pages 254–261, 1994.
13. P. Gupta, R. Janardan, and M. Smid. Algorithms for some intersection searching problems involving circular objects. *Intl. J. of Math. Algorithms*, 1:35–52, 1999.
14. H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM J. Comput.*, 15:478–494, 1986.
15. M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. In *Graph Drawing*, pages 165–174, 1999.
16. D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11:329–343, 1982.
17. J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.
18. J. Matoušek. Geometric range searching, 1994.
19. J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. In *Graph Drawing*, pages 263–274, 1998.
20. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.
21. C. Poon, B. Zhu, and F. Chin. A polynomial time solution for labeling a rectilinear map. In *Proc. 13th Sympos. Comp. Geom. (SoCG'97)*, pages 451–453, 1997.
22. R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Trans. Circuits Syst.*, CAS-36(9):1230–1234, 1989.
23. M. van Kreveld, T. Strijk, and A. Wolff. Point set labeling with sliding labels. In *Proc. 14th Annu. ACM Sympos. Comput. Geom. (SoCG'98)*, pages 337–346, 1998.

Appendix A

Lemma 4. *PD3IS is NP-hard.*

Proof Sketch: The proof is based on a reduction from PLANAR-VR3SAT. Let ϕ be an instance of PLANAR-VR3SAT with p clauses $C_1, C_2, \ldots, C_p$, where $C_i = (l_{i1} \vee l_{i2} \vee l_{i3})$ or $C_i = (l_{i1} \vee l_{i2})$. Each l_{ij} is either x_k or $\overline{x_k}$, where x_k is one of the q variables. The reduction constructs an instance of PD3IS, $(H = (V_H, E_H), \kappa_{IND})$ as follows: $V_H = \{l_{ij} : i = 1, \ldots, p; j = 1, 2, or 3\} \cup \{x_k : k = 1, \ldots, q\} \cup \{\overline{x_k} : k = 1, \ldots, q\}$ and $E_H = \{(l_{ij}, l_{ik}) : i = 1, \ldots, p; j \neq k\} \cup \{(l_{ij}, x_k)$ if $l_{ij} = \overline{x_k}$ or $(l_{ij}, \overline{x_k})$ if $l_{ij} = x_k : i = 1, \ldots, p; j = 1, 2, or 3; k = 1, \ldots, q\} \cup \{(x_k, \overline{x_k}) : k = 1, \ldots, q\}$, see Fig. 7, and $\kappa_{IND} = p + q$. Note that since we start with an instance of planar 3SAT, graph H is also planar. Moreover, because the instance ϕ also has the property that each variable occurs at most three times, H will have degree 3.

There exists an independent set I of H with size $\kappa_{IND} = p + q$ if and only if ϕ is satisfiable. Assume there exists an independent set I of size $p + q$. Since I has size $p + q$ then it must contain a node from each box in Fig. 7. Since the nodes are labeled with literals, and I does not contain nodes corresponding to opposite literals, I is a truth assignment that satisfies ϕ. That is, the labels of the vertices in I are assigned true and the truth assignment is valid (i.e. exactly one of x_i or $\overline{x_i}$ is true). This is because for each variable x_i, either ($x_i \in I$ and $\overline{x_i} \notin I$) or ($x_i \notin I$ and $\overline{x_i} \in I$), so that no two contradictory literals can be in I. I has a node from each clause box and thus the truth assignment satisfies all clauses. On the other hand, if a satisfying assignment exists, then we can identify a true literal in each clause c_i. Let a true literal in c_i be l_i. Then we can pick the vertex corresponding to l_i from the clause box for c_i, and the vertex corresponding to the literal l_i from the variable box for x_i in H to be in I. This yields $p + q$ independent vertices in H. □

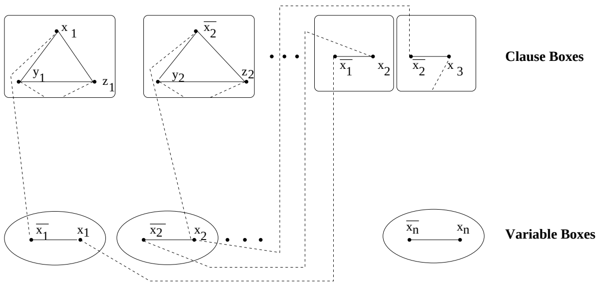

Fig. 7. A PLANAR-VR3SAT instance $(x_1 \vee y_1 \vee z_1) \wedge (\overline{x_2} \vee y_2 \vee z_2) \wedge \ldots \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_1) \wedge \ldots$. Since the 3SAT instance is planar, the graph with edges from clause to variable boxes is also planar.