Covering Points with Pairs of Concentric Disks*

Anil Maheshwari† Saeed Mehrabi† Sasanka Roy‡ Michiel Smid†

Abstract

In this paper, we study the following problem motivated by applications in wireless local area networks. We are given a set of m pairs of concentric disks in d-dimensional space, $d \in \{1, 2\}$, where each pair consists of one disk with radius one and the other with radius two. We are also given a set of n points such that the union of the m pairs of disks covers all the n points. The goal is to select *exactly* one disk from each pair such that every point is covered by at least one disk and the number of points covered by at least one disk with radius one is maximized; we refer to this as the sDiskCover problem.

When $d = 1$ (i.e., we have m pairs of intervals on the real line), we give an exact algorithm that solves the sDiskCover problem in $O(m^2n)$ time. We also consider a special case of the problem for $d = 1$, and show that it can be solved in $O(mn)$ time. For $d = 2$, we prove that the sDiskCover problem is NP-hard.

1 Introduction

In this paper, we study a problem that is motivated by applications in wireless local area networks (WLANs) [1]. In a WLAN, all the *users* (also called *stations*) receive data from *access points*. An access point can operate exactly one frequency, which can be chosen from many different frequencies at the beginning. When an access point is activated by a single frequency, it covers a circular area inside a disk. Higher frequency has higher speed, but lower coverage in disk area (i.e., covers a disk with smaller radius); see Figure 1 for an example. One can view different frequencies at an access point as concentric disks that are centered at the access point with different radii. Disks with smaller radius have higher frequency, which means the corresponding access point can provide data with higher speed. If a user is within a higher-frequency region of an access point, then they can be supplied data with higher speed; this will correspond to the profit of the service provider who installs the frequency at the access point. The service provider has to provide services to all the users, which might force the service provider to allocate lower frequency at an access point to get a higher coverage area.[1] The objective of the service provider is to increase sum of the total speed provided to the users, which in turn will maximize the profit made by the service provider. In this paper, we formalize this problem with two types of frequencies.

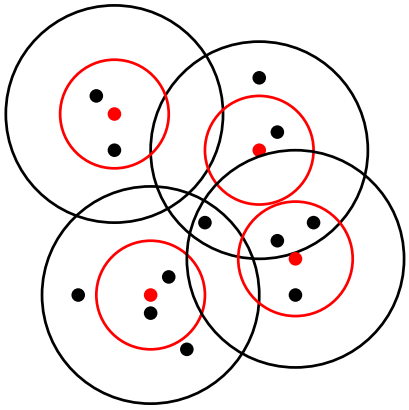

Figure 1: The points in red (resp., black) are the locations of access points (resp., users). The areas within red and black disks centered at each access point denote higher-to-lower frequency disks of the access point. A user can be served with the maximum speed if they are within a red disk and the corresponding access point is activated by that frequency.

Problem statement. Let $d \in \{1, 2\}$. Then, an *object* in d-dimensional space is a pair of disks, a disk with radius one and a disk with radius two, such that the disk with radius one is entirely contained in the one with radius two. For an object i, we call the disk of i with radius one (resp., two) the *small disk* (resp., *big disk*) of i and denote it by sDisk(i) (resp., bDisk(i)).

Consider a set of $n > 0$ points $p_1, \ldots, p_n$ and a set of $m > 0$ objects in d-dimensional space for some $d \in \{1, 2\}$. Then, the objective of the sDiskCover problem is to select *exactly* one disk from each object such that

*This work is supported in part by NSERC.

†School of Computer Science, Carleton University, Ottawa, Canada. anil@scs.carleton.ca, saeed.mehrabi@carleton.ca, michiel@scs.carleton.ca.

‡ACMU, Indian Statistical Institute (ISI), Kolkata, India. sasanka@isical.ac.in

[1] Here, we assume that the union of lowest-frequency disks covers all the users.

every point is contained in at least one disk and

$$\sum_{i=1}^{n} \min\{1, |\mathsf{small}(p_i)|\}$$

is maximized, where $\mathsf{small}(p_i)$ is the set of selected small disks that contain the point p_i. In other words, we want to select exactly one disk from each object such that all the points are covered and the number of points covered by at least one small disk is maximized.

Notation. For a point p in the plane, we denote the x- and y-coordinates of p by $x(p)$ and $y(p)$, respectively. Moreover, we denote the Euclidean distance between two points p_i and p_j by $\mathsf{dist}(p_i, p_j)$. For an object i, we denote the centres of $\mathsf{sDisk}(i)$ and $\mathsf{bDisk}(i)$ by $\mathsf{sCentre}(i)$ and $\mathsf{bCentre}(i)$, respectively.

Consider an instance of the sDiskCover problem. Let p be an input point that is contained in exactly one big disk $\mathsf{bDisk}(i)$ (for some object i) and not contained in any small disk. Then, any feasible solution must select $\mathsf{bDisk}(i)$. Moreover, let M be the set of all input points $q(\neq p)$ such that (i) q is covered by $\mathsf{bDisk}(i)$ and (ii) no small disk covers q (i.e., q is only covered by big disks). Then, we can include $\mathsf{bDisk}(i)$ into the solution, and then remove the object i and the set $M \cup \{p\}$ from the instance. Therefore, we assume the following throughout the paper.

Assumption 1 *Given an instance of the* sDiskCover *problem, if an input point is not contained in any small disk, then it is contained in at least two big disks.*

2 One-dimensional Objects

In this section, we consider the sDiskCover problem for n points and m one-dimensional objects: each object is a pair of intervals on the real line (i.e., an interval with length one and an interval with length two). For an interval i, we denote its left and right endpoints by $\mathsf{left}(i)$ and $\mathsf{right}(i)$, respectively. Moreover, we write $p(i)$ to denote the set of input points covered by i. For the rest of this section, we refer to the small and big disks of an object i as the small and big *intervals* of i and denote them by $\mathsf{slnt}(i)$ and $\mathsf{blnt}(i)$, respectively (we still use the term "object" whenever we are not referring to a specific interval). Moreover, we assume that the input points have distinct x-coordinates and $x(p_i)$ is distinct from that of the endpoints of any interval in the input objects, for all $1 \leq i \leq n$.

Here, we first consider the sDiskCover problem in a special case in which the objects are *left-aligned*: we have $x(\mathsf{left}(\mathsf{slnt}(i))) = x(\mathsf{left}(\mathsf{blnt}(i)))$ for all objects $1 \leq i \leq m$. In Section 2.2, we will solve the problem without this restriction. In this subsection, we assume that the points are ordered from left to right as $p_1, p_2, \ldots, p_n$, and the objects are sorted from left to right by the x-coordinate of the right endpoint of their *big* interval.

2.1 Left-aligned Intervals

An object i on the real line is called *left-aligned* if $x(\mathsf{left}(\mathsf{slnt}(i))) = x(\mathsf{left}(\mathsf{blnt}(i)))$; a set of one-dimensional objects is called left-aligned if every object in the set is left-aligned. Given a set of n points $p_1, \ldots, p_n$ and m one-dimensional left-aligned objects on the real line, we give an exact $O(mn)$-time algorithm for the sDiskCover problem.

For $1 \leq i \leq n$ and $1 \leq j \leq m$, define $A[i, j]$ to be the objective value of an exact solution for the problem on the points $p_1, p_2, \ldots, p_i$ and the objects $o_1, o_2, \ldots, o_j$. Similarly, define $B[i, j]$ to be the objective value of an exact solution for the problem on the points $p_1, p_2, \ldots, p_i$ and the objects $o_1, o_2, \ldots, o_j$, assuming that $\mathsf{blnt}(o_j)$ is in the solution. Our goal is to compute $A[n, m]$; the actual solution that gives us $A[n, m]$ can be computed in the standard manner. We next show how to compute $A[i, j]$ and $B[i, j]$. First, we need the following lemma.

Lemma 1 *Consider an instance of the* sDiskCover *problem, and let ℓ be the vertical line through $\mathsf{right}(\mathsf{slnt}(o_m))$. Moreover, assume that p_n lies to the right of ℓ, and let T denote the set of all big intervals that intersect ℓ (including $\mathsf{blnt}(o_m)$). Then, there exists an exact solution S for the problem such that $S \cap T \subseteq \{\mathsf{blnt}(o_m), \mathsf{blnt}(o_{m-1})\}$ and $S \cap T \neq \emptyset$.*

Proof. Since p_n lies to the right of ℓ, we have $|T| \geq 2$ by Assumption 1. If $|T| = 2$, then $T = \{o_{m-1}, o_m\}$ and so any feasible solution S must contain at least one of $\mathsf{blnt}(o_{m-1})$ and $\mathsf{blnt}(o_m)$. Hence, $S \cap T \subseteq \{\mathsf{blnt}(o_m), \mathsf{blnt}(o_{m-1})\}$ and $S \cap T \neq \emptyset$.

Now, assume that $|T| > 2$. Consider an exact solution that contains neither $\mathsf{blnt}(o_{m-1})$ nor $\mathsf{blnt}(o_m)$. Then, p_n must be covered by $\mathsf{blnt}(o_i)$ in this solution, for some $i < m - 1$, and so $\mathsf{blnt}(o_i) \in T$. We now replace $\mathsf{blnt}(o_i)$ and $\mathsf{slnt}(o_{m-1})$ with, respectively, $\mathsf{slnt}(o_i)$ and $\mathsf{blnt}(o_{m-1})$ in this solution; let S be the resulting set of intervals. Since the objects are left-aligned, these replacements do not leave any point uncovered: any point that was covered by $\mathsf{blnt}(o_i) \cup \mathsf{slnt}(o_{m-1})$ is still covered by $\mathsf{slnt}(o_i) \cup \mathsf{blnt}(o_{m-1})$. Moreover, since $i < m - 1$ and $o_i \in T$ (i.e., $\mathsf{blnt}(o_i)$ intersects ℓ), any point that was covered by $\mathsf{slnt}(o_{m-1})$ is still covered by $\mathsf{slnt}(o_i) \cup \mathsf{slnt}(o_m)$. Hence, S is a feasible solution for the problem and its objective value is at least as big as that of the initial solution. Observe that $S \cap T \subseteq \{\mathsf{blnt}(o_m), \mathsf{blnt}(o_{m-1})\}$ and $S \cap T \neq \emptyset$. □

Computing $A[i, j]$**.** Let ℓ denote the vertical line through $\mathsf{right}(\mathsf{slnt}(o_j))$. We consider two cases depending on whether p_i lies to the right or to the left of ℓ.

If p_i lies to the right of ℓ, then we can focus our attention to $\mathsf{bInt}(o_j)$ and $\mathsf{bInt}(o_{j-1})$ by Lemma 1. Let ℓ' be the vertical line through $\mathsf{right}(\mathsf{sInt}(o_{j-1}))$ and let $p_{i'}$ be the rightmost point that is to the left of ℓ'. Moreover, let ℓ'' be the vertical line through $\mathsf{left}(\mathsf{sInt}(o_j))$ and let $p_{i''}$ be the rightmost point that is to the left of ℓ''. Now, if $\mathsf{bInt}(o_j)$ is in the solution, then every point to the right of ℓ' is covered by $\mathsf{bInt}(o_j)$ and no such point is contained in a small interval; hence, the problem is reduced to $B[i', j]$. On the other hand, if $\mathsf{bInt}(o_{j-1})$ is in the solution, then we also take $\mathsf{sInt}(o_j)$ into the solution. Hence, the problem is reduced to $B[i'', j-1] + |p(\mathsf{sInt}(o_j))|$. Therefore, we have $A[i, j] = \max\{B[i', j], B[i'', j-1] + |p(\mathsf{sInt}(o_j))|\}$.

If p_i is to the left of ℓ, then we take $\mathsf{sInt}(o_j)$ into the solution. This is because if there exist a solution with $\mathsf{bInt}(o_j)$, then we can replace $\mathsf{bInt}(o_j)$ with $\mathsf{sInt}(o_j)$ without decreasing the objective value. Now, let ℓ' denote the vertical line through $\mathsf{left}(\mathsf{sInt}(o_j))$ and let $p_{i'}$ be the rightmost point that is to the left of ℓ'. Then, the problem is reduced to $A[i', j-1] + p(\mathsf{sInt}(o_j))$. In summary, we compute $A[i, j]$ as follows. First, assume that $i > 1$ and $j > 1$. If p_n is to the right of ℓ, then $A[i, j] = \max\{B[i', j], B[i'', j-1] + |p(\mathsf{sInt}(o_j))|\}$; otherwise, if p_n is to the left of ℓ, then $A[i, j] = A[i', j-1] + p(\mathsf{sInt}(o_j))$. Now, assume that $i = 1$. If p_1 is contained in at least one small interval, then $A[i, j] = 1$; but, if p_1 is contained in no small interval, then $A[i, j] = 0$. Finally, assume that $j = 1$. If there is at least one point that is not contained in $\mathsf{sInt}(o_1)$, then $A[i, j] = 0$; but, if every point is contained in $\mathsf{sInt}(o_1)$, then $A[i, j] = |p(\mathsf{sInt}(o_1))|$.

Computing $B[i, j]$. To compute $B[i, j]$, let ℓ be the vertical line through $\mathsf{right}(\mathsf{sInt}(o_j))$. We again consider two cases. If p_i is to the right of ℓ, then the problem is simply reduced to $B[i-1, j]$. If p_i is to the left of ℓ, then the problem is reduced to $A[i, j-1]$ because we can remove the object o_j from the instance (as we know that $\mathsf{bInt}(o_j)$ has been selected) and then solve the problem with the same points and the objects $o_1, o_2, \ldots, o_{j-1}$. Therefore,

$$B[i, j] = \begin{cases} B[i-1, j], & \text{if } p_i \text{ is to the right of } \ell, \\ A[i, j-1], & \text{if } p_i \text{ is to the left of } \ell. \end{cases}$$

Moreover, to compute the base cases, assume first that $i = 1$. If at least one of $\mathsf{sInt}(o_1), \ldots, \mathsf{sInt}(o_{j-1})$ contains p_1, then $B[i, j] = 1$; otherwise, $B[i, j] = 0$. Now, if $j = 1$, then $B[i, j] = 0$ because we have taken $\mathsf{bInt}(o_1)$ into the solution.

Running time. The tables A and B each have size mn, and we spend $O(1)$ time to fill one entry of A or one entry of B. Hence, the total time spent to fill out A and B is $O(mn)$ and so we have the following theorem.

Theorem 2 *For a set of n points and m left-aligned objects on the real line, the* sDiskCover *problem can be solved in $O(mn)$ time.*

2.2 Arbitrary Intervals

Here, we remove the "left-aligned" restriction and assume that the small interval of an object i can be anywhere within the big interval of the object as long as $\mathsf{dist}(\mathsf{sCentre}(i), \mathsf{bCentre}(i)) \leq 1/2$ (i.e., the small interval is entirely contained in the big interval). In this subsection, we assume that the objects are ordered from left to right by the x-coordinate of the right endpoint of their *small* interval.

Lemma 3 *There exists an optimal solution for the* sDiskCover *problem such that each point is covered by at most two small intervals.*

Proof. Take any optimal solution OPT for the problem. Let $S(p_i)$ denote the set of small intervals in OPT that cover point p_i for all $i = 1, 2, \ldots, n$. For each point p for which $|S(p)| > 2$, we do the following: let o_ℓ (resp., o_r) be the object for which $\mathsf{sInt}(o_\ell) \in S(p)$ (resp., $\mathsf{sInt}(o_r) \in S(p)$) and $x(\mathsf{left}(\mathsf{sInt}(o_\ell)) \geq x(\mathsf{left}(\mathsf{sInt}(o_j)))$ (resp., $x(\mathsf{right}(\mathsf{sInt}(o_r)) \leq x(\mathsf{right}(\mathsf{sInt}(o_j))))$ for all o_j such that $\mathsf{sInt}(o_j) \in S(p)$. Now, for every small interval in $S(p) \setminus \{\mathsf{sInt}(o_\ell), \mathsf{sInt}(o_j)\}$, we replace the small interval in OPT by its big interval. Clearly, every point is covered by at most two small intervals in the resulting set. Moreover, one can verify that the resulting set of intervals will still cover all the points and has the same objective value as OPT. □

We now describe a dynamic programming algorithm. Let $T[i, j]$ denote the objective value of an optimal solution for covering the points $p_1, p_2, \ldots, p_i$ with the objects $o_1, o_2, \ldots, o_j$ (where the latter ordering is by their small interval). Then, the goal is to compute $T[n, m]$. To compute $T[i, j]$, we assume in the following that the union of the j objects cover all the i points (as otherwise we set $T[i, j]$ to -1). Now, take any optimal solution OPT for $T[i, j]$ and let p_r be the rightmost point for which OPT gets a credit; that is, p_r is the rightmost point that is covered by at least one small interval in OPT. Then, by Lemma 3, p_r is covered by either one or two small intervals in OPT. Let us consider these in two cases.

Point p_r is covered by one small interval in OPT. Let o_a be the object such that $\mathsf{sInt}(o_a) \in$ OPT and $\mathsf{sInt}(o_a)$ covers p_r. Let ℓ (resp., ℓ') be the vertical line through $\mathsf{left}(\mathsf{sInt}(o_a))$ (resp., $\mathsf{right}(\mathsf{sInt}(o_a))$). Moreover, let M_a be the set of objects o_t such that $\mathsf{bInt}(o_t)$ intersects ℓ'. To see which interval of the objects in $M_a \setminus \{o_a\}$ are in OPT, take any object $o_t \in M_a \setminus \{o_a\}$. Observe

that if $\mathsf{sInt}(o_t) \in \mathsf{OPT}$, then $\mathsf{sInt}(o_t)$ does not cover any points in $\{p_{r+1}, \ldots, p_n\}$. Moreover, the points that lie to the right of ℓ and to the left of p_r are already covered by $\mathsf{sInt}(o_a)$ (and so for each of which OPT has gained a credit). This means that, the only way OPT could potentially gain points by having $\mathsf{sInt}(o_t)$ is when $\mathsf{left}(\mathsf{sInt}(o_t))$ lies strictly to the left of ℓ. In that case, among all such $\mathsf{sInt}(o_t)$, OPT must have the one with leftmost left endpoint; consider this object and let t^* be the index of its small interval (in the input ordering defined on small intervals). Notice that for all other objects in $M_a \setminus \{o_a\}$, we can have their big intervals in OPT. Let $p_{r'}$ for some $r' \leq r$, be the leftmost point covered by $\mathsf{sInt}(o_{t^*})$. Then, in this case, we have

$$T[i,j] = \max_{\substack{p_r \in \{p_1,\ldots,p_i\} \\ o_a \in \{o_1,\ldots,o_j\}: \\ p_r \in \mathsf{sInt}(o_a)}} \{f(\mathsf{sInt}(o_a), \mathsf{sInt}(o_{t^*})) + T[r'-1, t^*-1]\},$$

where $f(\mathsf{sInt}(o_a), \mathsf{sInt}(o_{t^*}))$ denotes the number of points covered by at least one of $\mathsf{sInt}(o_a)$ and $\mathsf{sInt}(o_{t^*})$.

Point p_r is covered by two small intervals in OPT. Let o_a and o_b be the two objects such that $\mathsf{sInt}(o_a), \mathsf{sInt}(o_b) \in \mathsf{OPT}$ and they both cover p_r. Assume w.l.o.g. that $x(\mathsf{left}(\mathsf{sInt}(o_a))) \leq x(\mathsf{left}(\mathsf{sInt}(o_b)))$; let ℓ (resp., ℓ') be the vertical line through $\mathsf{left}(\mathsf{sInt}(o_a))$ (resp., $\mathsf{right}(\mathsf{sInt}(o_b))$). Let M_{ab} be the set of objects o_t such that $\mathsf{bInt}(o_t)$ intersects ℓ'. To see which interval of the objects in $M_{ab} \setminus \{o_a, o_b\}$ are in OPT, take any object $o_t \in M_{ab} \setminus \{o_a, o_b\}$. Observe that if $\mathsf{sInt}(o_t) \in \mathsf{OPT}$, then $\mathsf{sInt}(o_t)$ does not cover any point in $\{p_{r+1}, \ldots, p_n\}$. Moreover, the points that lie to the right of ℓ and to the left of p_r are already covered by $\mathsf{sInt}(o_a)$ (and so for each of which OPT has gained a point). This means that, if $x(\mathsf{left}(\mathsf{sInt}(o_t))) \geq x(\mathsf{left}(\mathsf{sInt}(o_a)))$, then OPT does not gain any points by having $\mathsf{sInt}(o_t)$. Therefore, the only way OPT could potentially gain points by having $\mathsf{sInt}(o_t)$ is when $\mathsf{left}(\mathsf{sInt}(o_t))$ lies strictly to the left of ℓ. In that case, among all such $\mathsf{sInt}(o_t)$, OPT must have the one with leftmost left endpoint; consider this object and let t^* be the index of its small interval. Notice that for all other objects in $M_{ab} \setminus \{o_a, o_b\}$, we can have their big interval in OPT. Let $p_{r'}$, for some $r' \leq r$, be the leftmost point covered by $\mathsf{sInt}(o_{t^*})$. Then, in this case, we have

$$T[i,j] = \max_{\substack{p_r \in \{p_1,\ldots,p_i\} \\ \{o_a,o_b\} \subseteq \{o_1,\ldots,o_j\}: \\ p_r \in \mathsf{sInt}(o_a) \cap \mathsf{sInt}(o_b)}} \{f(\mathsf{sInt}(o_a), \mathsf{sInt}(o_b), \mathsf{sInt}(o_{t^*})) + T[r'-1, t^*-1]\},$$

where $f(\mathsf{sInt}(o_a), \mathsf{sInt}(o_b), \mathsf{sInt}(o_{t^*}))$ denotes the number of points covered by at least one of $\mathsf{sInt}(o_a), \mathsf{sInt}(o_b)$ and $\mathsf{sInt}(o_{t^*})$. The base case is $T[1,j]$ for all $j = 1, \ldots, m$; we set $T[1,j] = 1$ if p_1 is covered by at least one small interval in $\{\mathsf{sInt}(o_1), \mathsf{sInt}(o_2), \ldots, \mathsf{sInt}(o_j)\}$ and $T[1,j] = 0$, otherwise.

Running time. Given an instance of the problem, we can compute the order of the points and the intervals (as required by the algorithm) in $O(n \log n)$ and $O(m \log m)$ time, respectively. Moreover, within the same time bound, we can preprocess the input to compute the function $f(\mathsf{sInt}(o))$ for all the input objects o. Each entry of the table T can be computed in $O(m^2 n)$ time, and so we have the following theorem.

Theorem 4 *For a set of n points and m arbitrary intervals on the real line, the* sDiskCover *problem can be solved in $O(m^2 n)$ time.*

3 Two-dimensional Objects

In this section, we consider the sDiskCover problem for objects in the plane and show that the problem is NP-hard. Recall that for each object i, we have two disks: $\mathsf{sDisk}(i)$ whose radius is one and $\mathsf{bDisk}(i)$ whose radius is two. Throughout this section, we assume that $\mathsf{sCentre}(i) = \mathsf{bCentre}(i)$; i.e., the disks are centred at the same point.

We show a polynomial-time reduction from Planar Variable Restricted 3SAT (Planar VR3SAT, for short). Planar VR3SAT is a constrained version of 3SAT in which each variable can appear in at most three clauses and the corresponding *variable-clause graph* is planar. Efrat et al. [2] showed that Planar VR3SAT is NP-hard.

Let I_{SAT} be an instance of Planar VR3SAT with K clauses $C_1, C_2, \ldots, C_K$ and N variables $X_1, X_2, \ldots, X_N$; we denote the two literals of a variable X_i by x_i and $\overline{x_i}$. We construct an instance I_{sDC} of our problem such that I_{sDC} has a solution with objective value of at least $MNK + MK/2$, for some M that we will determine its value below, if and only if I_{SAT} is satisfiable. Given I_{SAT}, we first construct the variable-clause graph G of I_{SAT} in the non-crossing comb-shape form of Knuth and Raghunathan [3]. We assume w.l.o.g. that the variable vertices lie on a vertical line and the clause vertices are connected from left or right of that line; see Figure 2 (left) for an example. This representation has size polynomial in N and K.

Gadgets. For each variable $X_i \in I_{\mathsf{SAT}}$, we replace the corresponding variable vertex in G with two objects as shown in Figure 2 (right); we call this pair of objects the *variable gadget* of X_i. The top object serves as literal $\overline{x_i}$ while the bottom object serves as literal x_i. The variable gadget initially contains three disjoint *group of points*; we call each such group of points a *cloud*. There is one cloud of K points that is shared between $\mathsf{bDisk}(x_i)$ and $\mathsf{bDisk}(\overline{x_i})$, called a *variable-shared cloud*. Moreover,

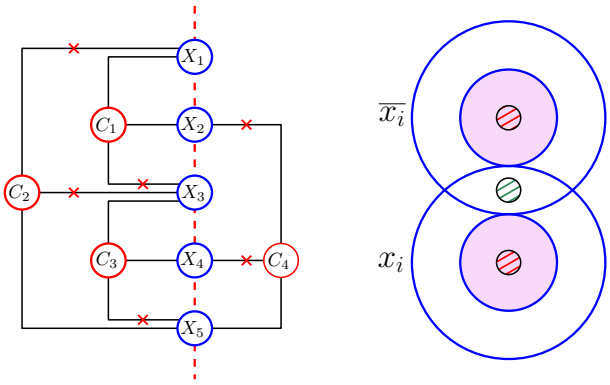

Figure 2: **Left**: an instance of the Planar VR3SAT problem in the comb-shape form of Knuth and Raghunathan [3]. Crosses on the edges indicate negations; for example, $C_2 = (\overline{x_1} \vee \overline{x_3} \vee x_5)$. **Right**: A variable gadget. The three clouds of the gadget are shown as small shaded disks.

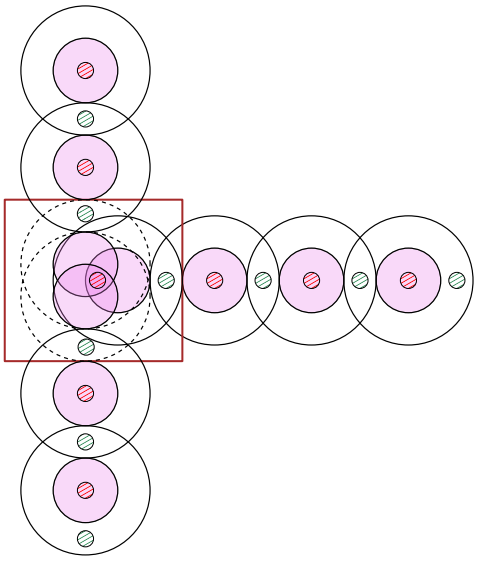

Figure 3: An illustration of a clause gadget.

each of $\mathsf{sDisk}(x_i)$ and $\mathsf{sDisk}(\overline{x_i})$ contains one cloud, each of which we refer to as a *variable-small cloud*. We determine the number of points in a variable-small cloud later.

Observation 1 *Given any feasible solution S for the* sDiskCover *problem, at most one of* $\mathsf{sDisk}(x_i)$ *and* $\mathsf{sDisk}(\overline{x_i})$ *can be in S, for any variable X_i.*

The idea behind the variable gadget (corresponding to a variable X_i) is to ensure that also at most one of $\mathsf{bDisk}(x_i)$ and $\mathsf{bDisk}(\overline{x_i})$ appears in any feasible solution. Then, we set the variable to true if and only if $\mathsf{bDisk}(x_i)$ is in the solution. However, the gadget as it is now does not enforce this. To enforce this, we must enforce one of the small disks to be selected in any feasible solution (hence, forcing its big disk not to be selected). To this end, we will set the number of points in each variable-small cloud (in the variable gadget) to a large enough value that any feasible solution must contain at least one small disk from every variable gadget in order for its objective value to meet a minimum requirement. We will determine this minimum requirement later.

If the literal x_i (resp., $\overline{x_i}$) appears in a clause, then the bottom object (resp., top object) of the gadget is connected to the corresponding clause by a chain of objects, called a *wire*. A wire starting from the bottom object (resp., top object) of a variable gadget and ending at a clause has the following structure. (i) Every object i in the wire has a cloud of K points in $\mathsf{sDisk}(i)$. (ii) The big disk of the first object of the wire shares one cloud of K points with the big disk of the bottom object (resp., top object) in the variable gadget. (iii) The big disks of every two consecutive objects in the wire share one cloud of K points. We call the first object of a wire (that shares a cloud with one of the objects in the variable gadget) the *starting object* of the wire.

For the clause gadget, where three wires meet, the small disks of the last three objects (each arriving from one of the wires) will have a non-empty intersection in which we place one cloud containing K points; see Figure 3. We call this K points a *clause cloud*.

Construction details. Let S be a feasible solution for the problem, and consider the object x_i shown in Figure 2 (right). The variable-small cloud of this object is covered by either $\mathsf{sDisk}(x_i)$ or $\mathsf{bDisk}(x_i)$ in S. If it is covered by $\mathsf{sDisk}(x_i)$ in S, then we must cover the other clouds in this object by the starting objects of the wires connected to x_i. But, if it is covered by $\mathsf{bDisk}(x_i)$ in S, then we can select the small disk of the starting object of each wire. Consequently, depending on whether $\mathsf{sDisk}(x_i)$ or $\mathsf{bDisk}(x_i)$ is selected, one can see that the clouds in the wires connected to this object are covered in S in one of the two possible ways. (Here, we are assuming that S needs to meet the minimum requirement for its objective value.) By re-scaling and adjusting the length of a wire, we can ensure that exactly one of these two possible coverings will let S to select the small disk of the last object in the wire; the other will only let S to select the big disk of the last object. In other words, exactly one of these two possible ways allows S to gain K points for covering the corresponding clause cloud.

By the discussion above, we set a variable X_i to true if and only if $\mathsf{bDisk}(x_i)$ is selected. By having an appropriate number of objects in each wire (while keeping the size polynomial), we can assume that S gains K points for covering the clause cloud (i.e., the small disk of the last object in the wire is selected) if and only if $\mathsf{bDisk}(x_i)$ is selected (i.e., variable X_i is set to true). Notice that

selecting $\mathsf{bDisk}(x_i)$ forces S to select $\mathsf{sDisk}(\overline{x_i})$. Now, by adjusting the number of objects in a wire connecting $\overline{x_i}$ to a clause gadget, we also ensure that the big disk of the last object of the wire is selected. That is, $\mathsf{sDisk}(\overline{x_i})$ is selected if and only if the big disk of the last object of the corresponding wire is selected (i.e., S does not gain any points from the corresponding clause cloud). Finally, we require by re-scaling that the number of objects in each wire is even. This concludes the consistency for the truth assignment of X_i.

We first prove that the number of objects in each wire is polynomial in N and K. Consider an edge in the graph and let L be its length. Notice that since the drawing is polynomial in N and K, so is L. Moreover, this edge can have either no bends or one bend. Our goal is to have each wire consistent with the drawing of its corresponding edge in G; hence, making each wire having no bends or one bend. Suppose first that the edge has no bends. Since the distance between every two consecutive centres of the disks in the wire is three, we have at most $\lfloor L/3 \rfloor$ objects in the wire. Now, suppose that the edge has one bend and let L_1 and L_2 denote the lengths of its segments (i.e., $L = L_1 + L_2$). Then, by a similar argument, the corresponding wire will have at most $\lfloor L_1/3 \rfloor + \lfloor L_2/3 \rfloor$ objects. We therefore conclude that the number of objects in both cases is polynomial in N and K.

In the full version of the paper, we prove that the wires can be connected to variable gadgets such that the objects from different wires do not intersect each other (except at clause gadgets and/or slightly at variable gadgets). To ensure this, we might require to "reroute" some of the wires; hence, making new bends. However, one can verify that the number of objects in each wire remains polynomial in N and K. The proof of the following lemma is given in the full version of the paper.

Lemma 5 *Let S be a feasible solution for the problem with objective value of at least $MNK + MK/2$. Then, S has exactly one big disk and exactly one small disk from every variable gadget.*

Lemma 5 gives us the minimum objective value for a feasible solution that we will use to argue that then the instance I_{SAT} is satisfiable.

Lemma 6 *There exists a feasible solution S for I_{sDC} with objective value of at least $MNK + MK/2$ if and only if I_{SAT} is satisfiable.*

Proof. ($\Rightarrow$) Let S be a feasible solution for I_{sDC} with objective value of at least $MNK+MK/2$. By Lemma 5, we know that there is exactly one big disk from every variable gadget in S. For each variable X_i, $1 \leq i \leq N$, we set the variable to true if and only if $\mathsf{bDisk}(x_i)$ is in S; otherwise, we set X_i to false. To show that this results in a truth assignment, suppose for a contradiction that there exists a clause C that is not satisfied by this assignment. Take any variable $X \in C$. If $x \in C$, then the variable X is set to false (resp., true) by the assignment. This means that S has selected the small disk of object x. Consequently, the big disk of the last object in the wire connecting C to x_i is selected by S: the solution S did not gain K points from the cloud of C. Analogously, if $\overline{x} \in C$, then the variable X is set to true by the assignment. This means that S has selected $\mathsf{sDisk}(\overline{x_i})$. Consequently, the big disk of the last object in the wire connecting C to $\overline{x_i}$ is selected by S: again, the solution S did not gain K points from the cloud of C. Therefore, S cannot have an objective value of $MNK + MK/2$—a contradiction.

($\Leftarrow$) Given a truth assignment for I_{SAT}, we construct a feasible solution S for I_{sDC} with objective value $MNK + MK/2$ as follows. For each variable X_i in I_{SAT}, where $1 \leq i \leq N$: if X_i is set to true, then we add $\mathsf{bDisk}(x_i)$ to S; otherwise, we add $\mathsf{sDisk}(x_i)$ to S. This selection ensures that we get MK points from each variable gadget. Moreover, by selecting the corresponding disk of x_i, we will select the disks in the wires connected to x_i accordingly by alternating between small and big disks. The same also happens for the wires that are connected to $\overline{x_i}$. One can consequently argue that, within a wire, exactly half of the small disks are selected; that is, we will gain $MK/2$ points by covering these clouds using small disks. Therefore, S has the objective value $MNK + MK/2$. □

By Lemma 6, we have the following theorem.

Theorem 7 *The* sDiskCover *problem is* NP*-hard for concentric disks in the plane.*

References

[1] D. Bhaumick and S. C. Ghosh. Efficient multicast association to improve the throughput in IEEE 802.11 WLAN. *MONET*, 21(3):436–452, 2016.

[2] A. Efrat, C. Erten, and S. G. Kobourov. Fixed-location circular arc drawing of planar graphs. *J. Graph Algorithms Appl.*, 11(1):145–164, 2007.

[3] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.