A Survey of the Geometric Set Cover Problem with Unit Disks in the Plane

Henry Vu

April 29, 2025

Abstract

The Geometric Set Cover Problem (GSCP) is a fundamental problem in computational geometry where the goal is to cover a set of points using the minimum number of geometric objects from a given family. This survey focuses on the specific case of covering points in the plane ($\mathbb{R}^2$) with unit disks. I will review the problem's formal definition, its connection to linear programming and duality (hitting sets), the role of range spaces, VC-dimension, nets and samples, various algorithmic approaches (greedy, LP-based, local search, PTAS), key theoretical approximation bounds, and representative pseudocode.

Contents

1 Problem Definition

Definition 1.1 (Geometric Set Cover)**.** *We are given a set X of n points in the plane $\mathbb{R}^2$ and a set $\mathcal{D}$ of m unit disks (radius 1) in the plane. Each disk $D \in \mathcal{D}$ covers the subset of points $x \in X$ that lie inside it (or on its boundary). The goal is to find a minimum-cardinality subset $C \subseteq \mathcal{D}$ such that every point in X is covered by at least one disk in C. Formally, we seek C minimizing $|C|$ subject to $\bigcup_{D \in C} D \supseteq X$. This is the set cover formulation: points X are the "universe" of elements to cover, and each disk D corresponds to a set of points. This problem is known as the Unit Disk Set Cover problem.*

Its decision version (given k, decide if k disks suffice to cover all points) is NP-complete. Thus, we do not expect an efficient algorithm for the exact optimum in the worst case (unless P=NP), motivating approximation algorithms.

Primal and Dual Linear Programs: The set cover problem can be formalized with an integer linear program (ILP). Introduce a binary variable x_D for each disk $D \in \mathcal{D}$, where $x_D = 1$ means we select disk D in the cover (and $x_D = 0$ means not selected). The ILP is:

Primal (Set Cover ILP):

$$\begin{aligned} \min \quad & \sum_{D \in \mathcal{D}} x_D \\ \text{subject to} \quad & \sum_{D : x \in D} x_D \geq 1, \forall x \in X, \\ & x_D \in \{0,1\}, \quad \forall D \in \mathcal{D}. \end{aligned}$$

$$\begin{aligned} \min \quad & \sum_{D \in \mathcal{D}} x_D \\ \text{subject to} \quad & \sum_{D : x \in D} x_D \geq 1, \forall x \in X, \\ & 0 \leq x_D \leq 1, \quad \forall D \in \mathcal{D}. \end{aligned}$$

The constraint means each point $x \in X$ must be covered by at least one chosen disk. Relaxing $x_D \in \{0,1\}$ to $0 \leq x_D \leq 1$ yields the set cover LP relaxation.

Dual (Hitting Set LP): By linear programming duality, one can derive the dual of the above LP. Introduce a nonnegative dual variable $y_x \geq 0$ for each point $x \in X$. The dual objective maximizes $\sum_{x \in X} y_x$ subject to $\sum_{x \in D} y_x \leq 1$ for each disk $D \in \mathcal{D}$ (each dual constraint corresponds to a primal variable x_D). This dual LP can be interpreted as a fractional hitting set: we assign weights y_x to points such that no single disk has total weight exceeding 1, and we maximize the total weight. In integral terms, the dual suggests the Geometric Hitting Set problem: choose a set $H \subseteq \mathbb{R}^2$ of points (not necessarily the same as X) of minimum size such that every disk $D \in \mathcal{D}$ contains at least one chosen point (i.e., H "hits" all disks). The hitting set of the dual range space $(X^\perp, \mathcal{D}^\perp)$ is equivalent to the set cover of $(X, \mathcal{D})$.

Range Spaces, VC-Dimension, Samples and Nets: The pair $\Sigma = (X, \mathcal{D})$ forms a range space, where X is set of points and $\mathcal{D}$ is a family of subsets of X (ranges). The VC-dimension of $(X, \mathcal{D})$ is the size d of the largest subset of X that can be shattered by $\mathcal{D}$ (i.e., for which $\mathcal{D}$ induces

all 2^d possible intersections). Planar regions like disks have finite VC-dimension; in fact, the VC-dimension of unit disks in $\mathbb{R}^2$ is at most 3, since any three points can define a disk boundary, but no four points can be shattered by unit disks. Finite VC-dimension implies the existence of small ε-nets by classical theorems (Haussler–Welzl theorem). An ε-net for $(X, \mathcal{D})$ is a subset $N \subseteq X$ such that for every range (disk) $D \in \mathcal{D}$ that contains at least $\varepsilon|X|$ points of X, D contains at least one point of N. In other words, N "hits" all heavy ranges (those covering an ε fraction of points). A fundamental result is that for any range space of VC-dimension d, there exists an ε-net of size $O(\frac{d}{\varepsilon} \log \frac{1}{\varepsilon})$ Matoušek et al. (1990). For points and disks in the plane (d constant), this means even for very small ε, one can find relatively small hitting sets for large ranges. In particular, if a point set X can be covered by k unit disks, one can show (using an ε-net with $\varepsilon = 1/k$) that there is a hitting set of size $O(k \log k)$ that intersects every disk of an optimal cover (Agarwal and Pan (2014)). This connection between set cover optimal solutions and ε-nets is the basis of several approximation algorithms discussed later.

Approximation Measures: An algorithm for the unit disk cover problem that runs in polynomial time and always produces a cover using at most $\alpha \cdot \mathrm{OPT}$ disks (where OPT is the minimum number in an optimal solution) is called an α-approximation algorithm (with approximation ratio α). A family of algorithms achieving $(1+\varepsilon)$-approximation for every $\varepsilon > 0$ (with runtime typically exponential in $1/\varepsilon$ but polynomial in n for fixed ε) is a PTAS. On the other end, a simple greedy strategy yields an $O(\log n)$-approximation in general. Between these extremes, researchers have developed constant-factor approximations (e.g., 5-approx, 2-approx, etc.) and improved algorithms leveraging geometry. I will discuss these in detail.

2 Algorithmic Approaches

Designing efficient algorithms for unit disk set cover has been a rich area of study. I organize known approaches by technique and the approximation ratios they achieve, roughly moving from simpler (but weaker approximation) methods to more complex methods.

2.1 Greedy Algorithm (Baseline Logarithmic Approximation)

The classic greedy algorithm for set cover provides a baseline solution. In each iteration, it picks the disk that covers the largest number of *uncovered* points, marks those points as covered, and repeats until all points are covered. For unit disk cover, the greedy algorithm can be shown to achieve $O(\log n)$ approximation in the worst case. Intuitively, if every chosen disk covers at least a constant fraction of the remaining points, the number of rounds (and disks) needed is $O(\log n)$. A worst-case arrangement of points can force each disk to cover only one new point, resulting in no improvement over trivial solutions, hence $\Theta(\log n)$ is worst-case.

Geometric instances often fare better in practice than the worst case. The greedy algorithm tends to perform closer to optimal on random distributions of points because disks can cover many points at once. Nonetheless, without additional structure, greedy cannot guarantee better than logarithmic approximation in theory. Early research in geometric set cover aimed to leverage structure to beat this bound. We will see that by incorporating sampling and primal-dual ideas, one can indeed get sublogarithmic or constant approximation ratios in the unit disk case.

2.2 Primal-Dual and LP-Based Approaches

A large class of algorithms for geometric set cover work by solving or approximating the linear programming relaxation. These include iterative reweighting (multiplicative weight update) methods, ε-net based rounding, and other primal-dual strategies. The high-level idea is to use fractional solutions or dual weights to guide the selection of disks, typically achieving improved approximation ratios by leveraging VC-dimension and sampling properties.

2.2.1 Iterative Reweighting and ε-Net Method

Bronnimann (1995) proposed an influential algorithm for set cover in finite VC-dimension set systems. Their algorithm (generalizing (Clarkson, 1993)) maintains a weight on each point in X and iteratively finds a small hitting set for the "heaviest" ranges. In the context of hitting set, the algorithm can be described as follows:

1. Start with all points having equal weight (say $w(x) = 1$ for $x \in X$). Let total weight $W = \sum_{x \in X} w(x)$.
2. While there exists an uncovered point (initially all are uncovered):
   (a) Compute an ε-net N of the current weighted point set for a suitable choice of ε. The ε-net N is a subset of points that hits every disk $D \in \mathcal{D}$ carrying weight $w(D) \geq \varepsilon W$ (where $w(D) = \sum_{x \in D} w(x)$). In practice, this can be done by known algorithms for ε-nets (often via random sampling or greedy selection).
   (b) Add the net N (points) to the hitting set under construction (or disks corresponding to those net points will be selected in the cover). If N already hits all disks (or covers all points in the primal view), we are done.
   (c) If some disks are still not hit, it means those disks each contain less than εW weight (otherwise N would have hit them). Increase the weights of all unhit points (points in X that are still not covered by the chosen disks), typically by multiplying their weight by 2. This weight increase is designed so that previously uncovered points become "heavier", forcing the next ε-net to include points hitting the ranges that cover them. Ensure weights remain bounded by capping it at W.
3. Repeat with the updated weights treated as the new instance.

Bronnimann (1995) proved that this algorithm finds a hitting set (and thus a set cover) of size $O(\mathrm{OPT} \log \mathrm{OPT})$ where OPT is the size of an optimal solution. In this context, $\kappa(\Sigma)$ denotes the minimum hitting set size and $\chi(\Sigma)$ the minimum set cover size for the range space $\Sigma = (X, \mathcal{D})$. The algorithm runs in near-linear time $O(n \log n)$ given one can compute ε-nets and answer range queries in polylog time. Essentially, this yields an $O(\log \mathrm{OPT})$-approximation, which is much better than $O(\log n)$ when OPT is small. It is "almost optimal" in the sense that one cannot hope for $o(\log \mathrm{OPT})$ in general without breaking known complexity barriers. Indeed, this result matches the size bound of ε-nets (which is tight up to constants for some range spaces (Agarwal and Pan, 2014)).

For unit disks, this method guarantees an $O(\log \mathrm{OPT})$ solution. For example, if the optimal cover uses k disks, Brönnimann–Goodrich might return on the order of $k \log k$ disks. This is a significant improvement over $O(\log n)$ since typically $k \ll n$.

Implementation considerations: The iterative reweighting approach requires the ability to compute ε-nets efficiently at each iteration and to identify uncovered ranges. Computing an ε-net for weighted points and disks can be done via greedy hitting set on a sample or via existing algorithms. In practice, one might implement a simpler variant: at each step, pick a point that lies in a large number of still-uncovered disks (this is a heuristic for hitting many remaining ranges), then increase weights and repeat. Data structures for range counting (to identify heavy ranges) or for selecting points covering many uncovered ranges (dual of selecting disks covering points) can be complex. Agarwal and Pan (2014) noted that a straightforward implementation of even a greedy or MWU-based algorithm may require dynamic geometric data structures to maintain counts as points become covered.

2.2.2 LP Rounding and Quasi-Uniform Sampling

Another LP-based approach is via linear programming relaxation and randomized rounding or sampling. Varadarajan (2010) introduced the quasi-uniform sampling technique for geometric set cover. The idea is to randomly sample disks with probabilities biased by an optimal fractional solution, but in a way that "uniformizes" the weights so that no region of the plane is too heavily covered by the fractional solution. Varadarajan's result was a breakthrough in obtaining a constant-factor approximation for weighted geometric set cover. Specifically, for disks, he achieved an $O(1)$-approximation by clever sampling that circumvented the difficulty of large weight variance.

To illustrate the approach: suppose we have an optimal fractional solution to the set cover LP. In a fractional solution, each disk D has an assignment $x_D \in [0, 1]$. One could try to independently pick disk D with probability x_D — but this by itself doesn't guarantee a cover (points may get left out at random). However, by conditioning on hitting heavy regions, one can ensure that with high probability all large clusters of points are covered. The quasi-uniform sampling technique adjusts the probabilities so that each point is covered with roughly uniform slack. The end result is a randomized algorithm that outputs a constant-factor cover (with some small probability of failure that can be made negligible).

2.2.3 Multiplicative Weight Update (MWU) and Dual Fitting

Building on these ideas, Agarwal and Pan (2014) developed two algorithms based on the multiplicative weight (MWU) method, tailored for geometric set cover. One algorithm was a refined variant of Brönnimann–Goodrich (with improvements to run in only $O(\log n)$ rounds by processing ranges in groups). The second algorithm viewed the problem as a two-player zero-sum game between an Alice who selects points and a Bob who selects disks. In this game interpretation, Alice's mixed strategy corresponds to a fractional hitting set (point weights) and Bob's to a fractional cover (disk weights). The MWU method is used to approach equilibrium: essentially, both player strategies are updated multiplicatively based on the other's choices. Agarwal and Pan's second algorithm produced a near-optimal fractional solution (dual weights) in a randomized fashion, then extracted a small hitting set using an ε-net as a final rounding step.

The key result of Agarwal and Pan was a near-linear time $O(1)$-approximation for covering points by disks. Specifically, they achieved (with high probability) a constant-factor approximation with running time $O(n \log^4 n)$ (Agarwal and Pan, 2014). This was the first algorithm to break the logarithmic barrier and run in near-linear time for unit disk cover in the plane. The constant factor was not explicitly optimized in their analysis, but is a fixed small number (independent of n). Their method relies on dynamic data structures for range counting (to support weight updates efficiently)

and careful randomization to avoid heavy updates cost. They also gave a deterministic variant at the cost of an extra logarithmic factor in running time.

2.2.4 Further Improvements – Near-Optimal Time Algorithms

More recently, researchers have optimized these approximation algorithms for speed. Chan and He (2020) revisited the MWU approach and introduced techniques like shallow cuttings (a data structure technique for range searching) and random sampling to simplify and speed up the computations. They managed to improve Agarwal and Pan's $O(n \log^4 n)$ to $O(n \log^3 n \log \log n)$ deterministic and even $O(n \log n (\log \log n)^{O(1)})$ randomized time, still for an $O(1)$ approximation. In subsequent work, these results were pushed to optimal $O(n \log n)$ time for constant-factor approximation (He, 2023). This line of work emphasizes that not only is good solution quality achievable, but also with algorithms efficient enough for large-scale implementation.

Summary of LP-based Approaches: The table below summarizes the progression for unit disks:

Table 1: Progression of LP-based/Related Algorithms for Unit Disk Cover

Algorithm/Result	**Approx. Ratio**	**Time Complexity**
Greedy/Basic	$O(\log n)$	$O(nm)$
Bronnimann (1995)	$O(\log \mathrm{OPT})$	$O(n \operatorname{polylog} n)$
Varadarajan (2010)	$O(1)$ (weighted)	Quasi-linear (rand.)
Agarwal and Pan (2014)	$O(1)$	$O(n \log^4 n)$ (rand.)
Chan and He (2020)	$O(1)$	$O(n \log^3 n \log \log n)$ (rand.)
He (2023)	$O(1)$	$O(n \log n)$ (det.)

The approximation ratios here are all constant (often not explicitly stated, but achievable constants range from about 3 up to maybe 16 or more depending on the method). The focus was on reducing running time while maintaining constant approximation.

2.3 Local Search and PTAS Approaches

Local search is a powerful technique that has yielded polynomial-time approximation schemes for many geometric covering problems. The idea is to start with a feasible solution (e.g., by greedy) and then iteratively improve it by local modifications, such as exchanging a few disks for fewer other disks, until a local optimum is reached. Mustafa and Ray (2009) showed that local search yields a PTAS for hitting set and set cover in a broad class of geometric range spaces, including unit disks in the plane.

2.3.1 Local Search PTAS

The algorithm fixes an integer parameter $k \geq 1$ (which will be a function of the desired ε) and starts with any valid set cover C. Then it repeatedly looks for an opportunity to replace ℓ disks in the current solution with r disks not in the solution, where $\ell \leq k$ and $r < \ell$. If such an exchange is found (i.e., we can cover all points that those ℓ disks covered by using r new disks, thereby reducing the total count by $\ell - r$), the algorithm performs the exchange and improves the solution. It keeps doing this until no improving exchange of size $\leq k$ exists. When no local improvement of bounded size is possible, we have a local optimum.

Mustafa and Ray proved that for $k = O(1/\varepsilon)$, any local optimum is in fact a $(1+\varepsilon)$-approximation to the global optimum for geometric set cover problems with "pseudo-disks" (which includes unit disks). Intuitively, if an optimal solution C^* differs from the local-optimal solution C in at most k disks, the local search would have found that exchange. They use a shifting argument on the arrangement of disks to show that one can transform C to C^* via a sequence of exchanges of size at most k, so if C were much worse than C^*, at least one of those exchanges would be beneficial and would have been made. By contradiction, C must be near-optimal.

In complexity terms, letting $k = O(1/\varepsilon)$, the algorithm checks subsets of up to k disks in the current solution to try replacing them. A naive implementation would be $O(n^k)$ which is not polynomial for variable k. However, k is a constant for fixed ε, so the running time is $n^{O(1/\varepsilon)}$, which is polynomial for any fixed ε (but grows quickly as ε shrinks). Mustafa and Ray's analysis gives a running time roughly $O(n^{O(1/\varepsilon^2)})$ for $(1+\varepsilon)$ approximation. For example, a 1% approximation ($\varepsilon = 0.01$) might be impractically slow (since $1/\varepsilon = 100$). But for moderate ε like 0.3, $k \approx 3$ and the local search might involve checking exchanges of size up to 3, which could be feasibles.

Implementation: The local search PTAS is conceptually simple to implement: one needs a subroutine to check if a set of at most k disks can be replaced by a smaller set to still cover the points. This can be done by brute force for small k, or by clever observation (for unit disks, checking an exchange of size 2 or 3 might involve seeing if the union of two disks' coverage can be achieved by one disk, etc., which for unit disks in the plane has geometric meaning). In practice, local search might converge faster than worst-case and often yields very good solutions. Its drawback is worst-case running time with large k.

2.4 Algorithm Pseudocode

To illustrate, here is the pseudocode for two key algorithms: the Greedy set cover and the Local Search algorithm. These highlight the simplicity vs. sophistication trade-off:

Algorithm 1 Greedy Algorithm for Unit Disk Cover

```
procedure GreedyUnitDiskCover(X, 𝒟)        ▷ X: set of points, 𝒟: set of unit disks
    C ← ∅                                   ▷ Cover set
    U ← X                                   ▷ Set of uncovered points
    while U ≠ ∅ do
        Find D_best = arg max_{D_i ∈ 𝒟} |D_i ∩ U|   ▷ Disk covering most uncovered points
        if no disk covers any point in U then      ▷ Remaining points are uncoverable
            break
        end if
        C ← C ∪ {D_best}
        U ← U \ D_best                      ▷ Remove covered points from U
    end while
    return C
end procedure
```

In each iteration, Algorithm 1 picks the disk covering the largest number of currently uncovered points. The removal of points and recomputation of cover counts can be done efficiently by indexing points to disks (for example, using an array of lists of incident disks per point). With careful use of data structures (e.g. a priority queue for candidate disks or Fenwick tree for updating counts), the

greedy algorithm can handle moderately large inputs. The loop runs until all points are covered, so in worst case $|C|$ iterations (which could be up to n in degenerate cases).

Algorithm 2 Local Search PTAS (with exchange up to k disks)

```
procedure LocalSearchUnitDiskCover(X, 𝒟, k)
    C ← GreedyUnitDiskCover(X, 𝒟)                          ▷ Start with an initial cover
    improved ← true
    while improved do
        improved ← false
        for ℓ ← 1 to k do                                   ▷ Try exchanges of size ℓ
            for all subset S ⊆ C such that |S| = ℓ do
                P ← ⋃_{D∈S}(D ∩ X)          ▷ Points covered *only* by disks in S might be better
                T ← FindMinCover(P, 𝒟 \ C, ℓ − 1)   ▷ Find min cover for P using ≤ ℓ − 1 disks outside C
                if T ≠ null and |T| < ℓ then                ▷ Check if a smaller cover was found
                    C ← (C \ S) ∪ T                         ▷ Perform the exchange
                    improved ← true
                    goto :RestartWhileLoop                  ▷ Restart search after improvement
                end if
            end for
        end for
    end while
    return C
end procedure
procedure FindMinCover(P, 𝒟′, max_r)  ▷ Subroutine to find min cover for points P using ≤ max_r disks from 𝒟′
    for r ← 1 to max_r do
        for all subsets T ⊆ 𝒟′ such that |T| = r do
            if P ⊆ ⋃_{D∈T} D then                           ▷ Check if T covers all points in P
                return T                                    ▷ Return the first smallest cover found
            end if
        end for
    end for
    return null                                             ▷ No cover found with size ≤ max_r
end procedure
```

Algorithm 2 attempts to find any exchange of up to k disks in the current solution C with fewer disks outside C that still cover the relevant points. If an improving exchange is found, it updates C and restarts the search for further improvements, until no improvement is possible. FindMinCover$(P, \mathcal{D} \setminus C, \ell - 1)$ can be implemented by brute force when ℓ is small, trying all combinations of at most $\ell - 1$ disks from $\mathcal{D} \setminus C$ to cover points P. Since k is a constant in the PTAS context, this is feasible. The 'goto' is used to restart the outer loop immediately upon finding an improvement, which is common in local search implementations. Mustafa and Ray's analysis guarantees this C is a $(1 + \varepsilon)$-approximation if $k = O(1/\varepsilon)$.

The pseudocode above abstracts away many details for clarity. Real implementations need careful handling of geometric data (like identifying the points P covered by a subset S of disks, which can

be done by maintaining an incidence structure of points to disks). Nonetheless, it illustrates the structure of the algorithms.

3 Theoretical Results

3.1 Computational Complexity

Theorem 3.1 (NP–completeness (Fowler et al., 1981)). *Given a point set $X \subset \mathbb{R}^2$ and a family $\mathcal{D}$ of unit disks, deciding whether there exists a cover of size k $(= |C|)$ is* NP-complete.

Idea. Membership in NP is immediate. Fowler–Paterson–Tanimoto reduce Planar-3SAT to geometric set cover: variable gadgets are implemented by chains of overlapping unit disks, and clause gadgets ensure at least one incident literal disk must be selected. The reduction introduces $|\varphi|^{O(1)}$ points and disks, hence is polynomial in the formula size $|\varphi|$. □

3.2 Range-Space Structure

Corollary 3.2 (ε-nets Haussler and Welzl (1986); Matoušek et al. (1990)). *For any $0 < \varepsilon < 1$ there exists an ε-net of size $O(1/\varepsilon)$ for $(X, \mathcal{D})$. This bound is tight up to constant factors.*

These nets yield the crucial hitting-set size bound $\kappa(X, \mathcal{D}) = O\big(\mathrm{OPT} \log \mathrm{OPT}\big)$ that underlies many approximation algorithms.

3.3 Approximation Algorithms

Greedy. The classical greedy algorithm chooses in each round the disk covering most yet-uncovered points and attains a $(\ln n + 1)$-approximation.

Iterative Reweighting. Bronnimann (1995) combine multiplicative weight updates with ε-nets to obtain a $O(\log \mathrm{OPT})$-approximate hitting set of size $O(\mathrm{OPT} \log \mathrm{OPT})$ in deterministic polynomial time.

Constant-factor via Quasi-uniform Sampling. Varadarajan (2010) introduced quasi-uniform sampling to obtain an $O(1)$-approximation for *weighted* pseudodisks (in particular, unit disks).

Near-linear-time $O(1)$-approximation. Agarwal and Pan (2014) achieved a randomized $O(1)$-approximation in $O(n \log^4 n)$ time by combining MWU with efficient range-count data structures. Chan and He (2020) replaced the data structure by shallow cuttings, lowering the randomized running time to $O(n \log n (\log \log n)^{O(1)})$ and the deterministic time to $O(n \log^3 n \log \log n)$. He (2023) pushes the deterministic bound to $O(n \log n)$.

PTAS via Local Search. For the unweighted case, Mustafa and Ray (2009) obtain a $(1+\varepsilon)$-approximation via k-exchange local search with $k = \Theta(1/\varepsilon)$, running in $n^{O(1/\varepsilon^2)}$ time. Li and Jin (2015) later extended PTAS techniques to the weighted case, giving an $n^{poly(1/\varepsilon)}$-time $(1+\varepsilon)$-approximation. For more general weighted pseudodisks, only a quasi-polynomial-time approximation scheme (QPTAS) is known Mustafa et al. (2015).

4 Research Landscape and Open Problems

The study of unit disk set cover is mature, yet several research directions remain open or active:

- **Improved Practical Algorithms:** While near-linear time $O(1)$-approximation exists (He, 2023), the constant factor might be large. Can we achieve a small constant factor (e.g., 2 or 3) with a simple, fast (near-linear time) deterministic algorithm?
- **Dynamic and Online Set Cover:** Maintaining an approximate set cover as points are inserted/deleted is important. Recent work (Agarwal et al., 2020; Chan et al., 2022) provides $O(1)$-approximation with polylogarithmic updates. Can we achieve $(1 + \varepsilon)$-approximation dynamically, or improve update times further? Online versions (where points arrive one by one) are also relevant.
- **Higher Dimensions and Other Shapes:** Covering points by balls in $\mathbb{R}^d$ for $d > 2$ has PTAS, but the runtime dependency on d and $1/\varepsilon$ is significant. Understanding the approximation threshold for non-disk shapes (e.g., ellipses, rectangles of arbitrary aspect ratio) is ongoing; fatness seems key for PTAS.
- **Combination with Networking Constraints:** In applications like wireless networks, requiring the chosen disks (base stations) to form a connected network leads to harder problems like connected set cover or connected dominating set. Achieving good approximations for these combined objectives is challenging.
- **Experimental Evaluation:** Thorough experimental comparisons of the state-of-the-art algorithms (near-linear $O(1)$-approx, PTAS variants, heuristics) on diverse datasets are needed to guide practical implementations. Which algorithm performs best for typical instance sizes (e.g., $n \approx 10^5$)?

In conclusion, the geometric set cover problem for unit disks exemplifies the fruitful interplay between computational geometry, approximation algorithms, and optimization techniques. While significant progress has led to near-optimal theoretical understanding (PTAS and efficient constant-factor approximations), intriguing questions remain, particularly concerning weighted variants, practical performance, dynamic settings, and fine-grained complexity.

References

Pankaj K Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 271–279, 2014.

Pankaj K. Agarwal, Lars Arge, Sathish Govindarajan, and Jun Yang. Dynamic geometric set cover. In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG)*, pages 4:1–4:16, 2020. doi: 10.4230/LIPIcs.SoCG.2020.4.

H Bronnimann. Almost optimal set covers in finite vc-dimension. *Discrete Comput. Geom.*, 14: 263–279, 1995.

Timothy M Chan and Qizheng He. Faster approximation algorithms for geometric set cover. *arXiv preprint arXiv:2003.13420*, 2020.

Timothy M Chan, Qizheng He, Subhash Suri, and Jie Xue. Dynamic geometric set cover, revisited. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3496–3528. SIAM, 2022.

Kenneth L Clarkson. Algorithms for polytope covering and approximation. In *Workshop on Algorithms and Data Structures*, pages 246–252. Springer, 1993.

Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137, 1981.

David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the second annual symposium on Computational geometry*, pages 61–71, 1986.

Qizheng He. *Geometric set cover and related geometric optimization problems.* PhD thesis, University of Illinois at Urbana-Champaign, 2023.

Jian Li and Yifei Jin. A ptas for the weighted unit disk cover problem. In *International Colloquium on Automata, Languages, and Programming*, pages 898–909. Springer, 2015.

Jiří Matoušek, Raimund Seidel, and Emo Welzl. How to net a lot with little: Small ε-nets for disks and halfspaces. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 16–22, 1990.

Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. In *Proceedings of the 25th Annual Symposium on Computational Geometry (SoCG)*, pages 174–183, 2009. doi: 10.1145/1542362.1542397.

Nabil H Mustafa, Rajiv Raman, and Saurabh Ray. Quasi-polynomial time approximation scheme for weighted geometric set cover on pseudodisks and halfspaces. *SIAM Journal on Computing*, 44(6):1650–1669, 2015.

Kasturi Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 641–648, 2010.