Les Cahiers du GERAD ISSN: 0711–2440

An exact algorithm for a class of geometric set-cover problems

C. Contardo,
A. Hertz

G–2020–17

February 2020

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

An exact algorithm for a class of geometric set-cover problems

Claudio Contardo [a,b]
Alain Hertz [a,c]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of Management and Technology, ESG UQÀM, Montréal (Québec) Canada, H3C 3P8*

[c] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7*

contardo.claudio@uqam.ca
alain.hertz@polymtl.ca

February 2020
Les Cahiers du GERAD
G–2020–17

Abstract: Given a set $\mathcal{R}$ of m disjoint finite regions in the 2-dimensional plane, all regions having polygonal boundaries, and given a set $\mathcal{D}$ of n discs with fixed centers and radii, we consider the problem of finding a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that every point in $\mathcal{R}$ is covered by at least one disc in $\mathcal{D}^*$. We show that this problem can be solved by using an iterative procedure that alternates between the solution of a traditional set-cover problem and the construction of the Laguerre-Voronoi diagram of a circle set.

1 Introduction

Geometric set-cover problems refer to problems related to finding a subset of minimum size from a set of geometric objects so as to cover all points of a given region of the considered space. Such problems naturally arise in numerous practical situations such as the location of wireless transmission antennas or the location of surveillance cameras. Most of these problems are NP-hard, even in the case of simple geometric objects (e.g., unit discs or squares in the plane), and simple regions to cover (e.g., rectangles or circles in the plane) [14].

In this paper, we consider the following general variant of the geometric set cover problem. Given a set $\mathcal{R}$ of m disjoint finite regions in the 2-dimensional plane, all regions having polygonal boundaries, and given a set $\mathcal{D}$ of n discs with fixed centers and radii, we consider the problem of finding a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that every point in $\mathcal{R}$ is covered by at least one disc in $\mathcal{D}^*$. As illustrated in Figure 1, the regions in $\mathcal{R}$ are not necessarily convex, they can have holes, and some of them can be reduced to a single point.

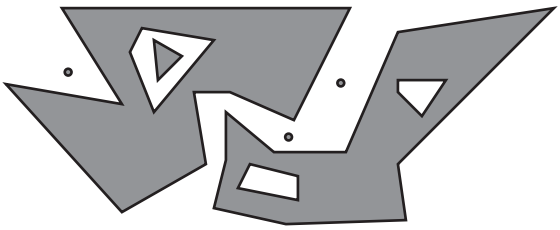


Figure 1: Example of a set $\mathcal{R}$ of six regions to be covered with discs.

Effort has been devoted to devise algorithms for variants of this general *region cover problem with discs* (RCD). For example, when every region in $\mathcal{R}$ is a point (i.e., $\mathcal{R}$ is a set of m points), we get the *set-cover problem for discs* (SCD) for which various approximation algorithms have been proposed when all discs have equal radius. Such approximation algorithms are however of little practical use since the solutions they provide are usually of low quality. In this article, we propose to determine the optimal solution of the RCD by iteratively solving SCDs of increasing size. The set of points that a particular SCD has to cover is determined using Laguerre-Voronoi diagrams of circle sets and their intersection with the boundaries of the regions in $\mathcal{R}$. Our main contributions are:

1. A mathematical model for a class of geometric set cover problems that encompasses multiple cases relevant in practice.
2. A decomposition of the considered problem into two easier subproblems that are solved in an iterative manner, namely a traditional set-cover problem with a finite set of constraints and the construction of the Laguerre-Voronoi diagram of a circle set.

The remainder of this article is structured as follows. In Section 2 we provide a comprehensive literature review of geometric set-cover problems. In Section 3 we present a mathematical model and introduce an exact method for the RCD. In Section 4 we provide computational experiments while Section 5 concludes the article.

2 Literature review

It is important to distinguish between various categories of geometric set-cover problems. One such category includes all problems where the precise locations of the objects used to cover a region of the Euclidian space are not given a priori. For example, one may be interested in covering a set of n points in the plane with a minimum number of discs of unit radius [19, 18]. The goal may also be to cover a region with k discs so that the largest radius is as small as possible [1]. The problem studied in this paper does not belong to this category since we assume that the geometric objects used to cover the given region are discs whose locations and sizes are fixed.

The type of region to be covered induces another important distinction between geometric set-cover problems. In some cases, the geometric objects have to cover a finite set of n points, while in other cases, the region to be covered contains an infinite number of points. For example, Basappa et al. [4] consider a finite sets $\mathcal{D}$ of units discs, with fixed centers and radii, and try to determine a minimum size subset of $\mathcal{D}$ that covers either a finite set of n points, or a rectangular region (with an infinite number of points).

RCD is very general in the sense that the region to be covered can contain a finite or an infinite number of points. It is NP-hard since several of its variants are known to be NP-hard problems. Indeed:

- Given a set $\mathcal{D}$ of unit discs, with known centers, Ko [17] has proved that the problem of determining whether a region with a polygonal boundary can be covered with a fixed number k of discs chosen in $\mathcal{D}$ is NP-complete. This is a special case of the decision version of RCD since the region $\mathcal{R}$ that we consider can have holes and is not necessarily connected. Also, the discs we use to cover $\mathcal{R}$ do not necessarily all have the same radii.
- Masuyama et al. [19] has proved that it is NP-hard to determine the minimum number of unit discs needed to cover a set of n points, even when each disc must have its center on one of the given n points. Again, this problem is a special case of the RCD since the region we aim to cover can have an infinite number of points, the discs we consider may be centered at arbitrary points in the plane, and their radii can differ.

Most of the algorithms proposed to solve geometric set-cover problems are approximation algorithms. For example, Hochbaum and Maass [15] have designed a polynomial-time approximation scheme (PTAS) for covering n points with unit discs which may be centered at arbitrary points in the plane. For this same problem, Biniaz et al. [5] recently proposed a 4-approximation algorithm which runs in $O(n\log n)$-time. If the centers of the unit discs must be chosen from a given list of possible locations, Das et al. [13] have proposed an 18-approximation algorithm which runs in $O(n\log n + m\log m + mn)$, while Basappa et al. [4] recently proposed a $(9+\epsilon)$-approximation algorithm that runs in $O(m^{3(1+\frac{6}{\epsilon})}n\log n)$ time for $0 < \epsilon \leq 6$. This is an improvement on previous approximation algorithms proposed by other authors [3, 9, 7, 20, 21].

From an algorithmic point of view, most effort on geometric set-cover problems was devoted to the development of approximation algorithms, as those mentioned above. Exact methods are rare and were designed only for very particular geometric set-cover problems. For example, Chan and Grant [8] have shown that given a set $\mathcal{D}$ of m discs with fixed centers and radii, the problem of determining a mimimum size subset of $\mathcal{D}$ that covers a given set of n points can be solved in polynomial time when there is a point in the plane that belongs to all discs of $\mathcal{D}$. In this paper, we propose an exact algorithm for the RCD. While it is of course not polynomial (since the RCD is NP-hard), we show that computing times are not too high, which allows to solve large problems.

The method proposed in this article is a special case of the more general concept of *relaxation algorithms* for combinatorial optimization. More precisely, given a minimization problem P, a relaxation algorithm iteratively builds *relaxed problems* P' obtained from P by removing or aggregating some constraints. The value of an optimal solution s' for a relaxed problem P' is clearly a lower bound on the optimal value for P. A heuristic procedure then plays on s' to try to recover a feasible solution s for P of the same value as this bound, in which case optimality is proven. If the constraint sets of the relaxed problems P' are significantly smaller than the constraint set of the original problem P, relaxation algorithms can possibly solve P whereas conventional optimization methods fail if they are applied directly to P. This idea has proven to be successful in a large number of applications such as clustering and facility location [2, 10, 12], interdiction games [11] and network design [6].

3 Exact algorithm

In this section we introduce an exact algorithm for the solution of the RCD. In a first subsection, we present a mathematical formulation for the RCD that contains an infinite number of constraints, and is therefore intractable by commercial solvers. In a second subsection we show how to decompose the mathematical model into a master problem and a subproblem that are solved in an alternating way. The master problem is a conventional (finite) set-cover problem, while the subproblem is to determine the Laguerre-Voronoi diagram of a finite set of circles.

3.1 Mathematical formulation

Let $\mathcal{R}$ be a set defined by the union of m disjoint finite regions in the 2-dimensional plane, all regions having polygonal boundaries. Also, let $\mathcal{D}$ be a set of n discs with fixed centers and radii. For every point $p \in \mathcal{R}$, we define $S(p)$ as the set of discs in $\mathcal{D}$ that cover p:

$$S(p) = \{d \in \mathcal{D} : p \in d\}$$

Note that $S(p)$ is finite for all $x \in \mathcal{R}$ as $\mathcal{D}$ contains a finite number of discs. For every $d \in \mathcal{D}$, let x_d be a binary variable indicating whether disc d is chosen in the optimal covering. The following is a valid formulation for the RCD:

$$\min \quad \sum_{d \in \mathcal{D}} x_d \tag{1}$$

subject to

$$\sum_{d \in S(p)} x_d \geq 1 \qquad p \in \mathcal{R} \tag{2}$$

$$x_d \in \{0,1\} \qquad d \in \mathcal{D}. \tag{3}$$

3.2 Iterative algorithm

Formulation (1)–(3) possibly contains an infinite number of constraints since $\mathcal{R}$ may contain an infinite number of points. Replacing $\mathcal{R}$ by a set $\mathcal{P} \subset \mathcal{R}$ of points in equation (2) yields however a relaxation and thus provides a lower bound on the optimal solution of the problem. For a given set $\mathcal{P}$ of points in $\mathcal{R}$, let $x^*(\mathcal{P})$ be an optimal solution of (1)–(3) where $\mathcal{R}$ is replaced by $\mathcal{P}$ and let $z^*(\mathcal{P})$ be its value (i.e., $z^*(P) = \sum_{d \in \mathcal{D}} x^*(\mathcal{P})_d$). Also, let $\mathcal{D}^*(P)$ be the subset of discs $d \in \mathcal{D}$ such that $x^*(\mathcal{P})_d = 1$. Note that determining $x^*(\mathcal{P})$ and $z^*(\mathcal{P})$ is a conventional (finite) set-cover problem where $\mathcal{P}$ is finite. This observation is of main importance for the scalability of our method. Clearly, $z^*(\mathcal{P})$ is a lower bound on $z^*(\mathcal{R})$. Observe also that if the discs in $\mathcal{D}^*(\mathcal{P})$ cover $\mathcal{R}$ then $z^*(\mathcal{P})$ is the optimal solution of (1)–(3) since $z^*(\mathcal{P})$ is then also an upper bound on $z^*(\mathcal{R})$.

The proposed algorithm starts by selecting an initial point $p_1 \in \mathcal{R}$ and by setting $\mathcal{P}_1 = \{p_1\}$. For instance, p_1 might me picked randomly in $\mathcal{R}$. We then iteratively generate sets $\mathcal{P}_2, \mathcal{P}_3, \ldots$ so that each $\mathcal{P}_i$ $(i > 1)$ is obtained from $\mathcal{P}_{i-1}$ by adding a set A of at most s points in $\mathcal{R}$ not covered by $\mathcal{D}^*(\mathcal{P}_{i-1})$, where s is a parameter. The process stops when $\mathcal{D}^*(\mathcal{P}_{i-1})$ covers $\mathcal{R}$.

In order to determine points in $\mathcal{R}$ that are not covered by $\mathcal{D}^*(\mathcal{P}_{i-1})$, we use the Laguerre-Voronoi diagram associated with the boundaries of the discs in $\mathcal{D}^*(\mathcal{P}_{i-1})$. Kim et al. [16] have shown how to efficiently compute the positions of the vertices and the equations of the edges in such diagrams. Let $E^*(\mathcal{P}_{i-1})$ be the edge set of the Laguerre-Voronoi diagram, let $V^*(\mathcal{P}_{i-1})$ be the set of Laguerre-Voronoi vertices in $\mathcal{R}$ and let $W^*(\mathcal{P}_{i-1})$ be the set of points located on the intersection of a boundary of $\mathcal{R}$ and an edge of $E^*(\mathcal{P}_{i-1})$. Also, let C be the set of corner points of the boundaries of $\mathcal{R}$. If some or the m regions in $\mathcal{R}$ contain a single point, then these points belong to C. Finally, let $U^*(\mathcal{P}_{i-1})$ be the set of points in $V^*(\mathcal{P}_{i-1}) \cup W^*(\mathcal{P}_{i-1}) \cup C$ that are not covered by $\mathcal{D}^*(\mathcal{P}_{i-1})$. The first point which is put in the set A of points added to $\mathcal{P}_{i-1}$ to obtain $\mathcal{P}_i$ is the farthest one in $U^*(\mathcal{P}_{i-1})$ from a boundary of a

disc in $\mathcal{D}^*(\mathcal{P}_{i-1})$. Then, while $|A| < min\{s, |U^*(\mathcal{P}_{i-1})|\}$, we add points in A, greedily, always choosing a point in $U^*(\mathcal{P}_{i-1})$ with largest minimum distance to a point in A.

This process is illustrated in Figure 2 with $\mathcal{P}_{i-1} = \{p_1, p_2, p_3, p_4\}$ (white squares) and $\mathcal{D}^*(\mathcal{P}_{i-1})$ made of the four discs on the left side of the figure. The Laguerre-Voronoi diagram contains two vertices a, b (white hexagons), and $V^*(\mathcal{P}_{i-1}) = \{b\}$ since a is outside $\mathcal{R}$. The edge set of the Laguerre-Voronoi diagram intersects the boundary of $\mathcal{R}$ at 9 points (white circles) so that $W^*(\mathcal{P}_{i-1}) = \{c, d, \ldots, k\}$. The set of corner points (black squares, except p_1, p_2) on the boundary of $\mathcal{R}$ is $C = \{l, \ldots, z, p_1, p_2\}$ so that $U^*(\mathcal{P}_{i-1}) = \{b, c, d, e, f, g, h, i, j, k, l, m, q, s, t, v, w, z\}$. Among these points, the farthest one from a boundary of a disc in $\mathcal{D}^*(\mathcal{P}_{i-1})$ is e so that $\mathcal{P}_i$ is set equal to $\mathcal{P}_{i-1} \cup \{e\}$ if $s = 1$. If $s = 2$, then point i is also added to $\mathcal{P}_{i-1}$. A possible solution to the set-cover problem with $\mathcal{P}_i = \{p_1, p_2, p_3, p_4, p_5\}$ is represented on the right side of Figure 2. It can be obtained by swapping the disc that covers p_1 with the disc that covers p_1 and p_5.

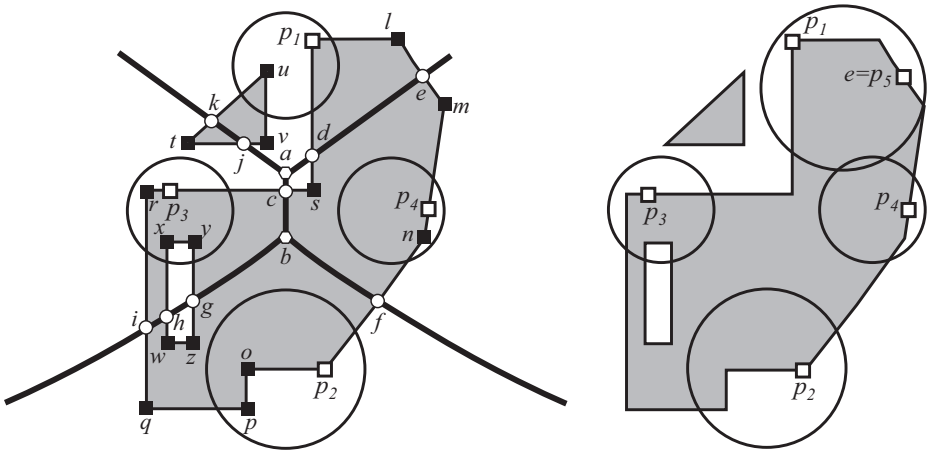


Figure 2: Illustration of the construction of $\mathcal{P}_i$ and $\mathcal{D}^*(\mathcal{P}_i)$ from $\mathcal{P}_{i-1}$ and $\mathcal{D}^*(\mathcal{P}_{i-1})$.

Algorithm 1 describes the general steps of the proposed method. Its validity follows from the following Property.

Algorithm 1 Exact method for the RCD

Choose $p_1 \in \mathcal{R}$ and set $\mathcal{P}_1 \leftarrow \{p_1\}$ and $i \leftarrow 1$.
Determine the set C of corner points of the boundaries of $\mathcal{R}$.
repeat
  Let *SetCover*($\mathcal{P}_i$) be the integer program (1)–(3) where $\mathcal{P}_i$ replaces $\mathcal{R}$ in (2).
  if *SetCover*($\mathcal{P}_i$) has no feasible solution **then**
    STOP : there is no feasible solution to the RCD.
  else
    Let $x^*(\mathcal{P}_i)$ be an optimal solution to *SetCover*($\mathcal{P}_i$).
    Set $\mathcal{D}^*(\mathcal{P}_i)$ equal to the subset of discs $d \in \mathcal{D}$ such that $x^*(\mathcal{P}_i)_d = 1$.
  end if
  Determine the Laguerre-Voronoi diagram associated with the boundaries of the discs in $\mathcal{D}^*(\mathcal{P}_i)$.
  Determine the set $V^*(\mathcal{P}_i)$ of Laguerre-Voronoi vertices in $\mathcal{R}$.
  Determine the set $W^*(\mathcal{P}_i)$ of points located on the intersection of a boundary of $\mathcal{R}$ and on an edge of the Laguerre-Voronoi diagram.
  Determine the set $U^*(\mathcal{P}_i)$ of points in $V^*(\mathcal{P}_i) \cup W^*(\mathcal{P}_i) \cup C$ that are not covered by $\mathcal{D}^*(\mathcal{P}_i)$.
  if $U^*(\mathcal{P}_i) \neq \emptyset$ **then**
    Determine a farthest point p in $U^*(\mathcal{P}_i)$ from a boundary of a disc in $\mathcal{D}^*(\mathcal{P}_i)$ and set $A \leftarrow \{p\}$.
    while $|A| < \min\{s, |U^*(\mathcal{P}_i)|\}$ **do**
      Determine the point $p \in U^*(\mathcal{P}_i)$ with largest distance to a point in A, and set $A \leftarrow A \cup \{p\}$.
    end while
    Set $i \leftarrow i + 1$ and $\mathcal{P}_i \leftarrow \mathcal{P}_{i-1} \cup A$.
  end if
until $U^*(\mathcal{P}_i) = \emptyset$.

Property 1 *$\mathcal{D}^*(\mathcal{P}_i)$ covers $\mathcal{R}$ if and only if $U^*(\mathcal{P}_i) = \emptyset$.*

Proof. *Necessity.* If the set $\mathcal{D}^*(\mathcal{P}_i)$ of discs covers $\mathcal{R}$ then it covers all points in $V^*(\mathcal{P}_i) \cup W^*(\mathcal{P}_i) \cup C$ (since they all belong to $\mathcal{R}$), which means that $U^*(\mathcal{P}_i) = \emptyset$.

Sufficiency. Assume $U^*(\mathcal{P}_i) = \emptyset$ and suppose there is a point p on a boundary of $\mathcal{R}$ that is not covered by $\mathcal{D}^*(\mathcal{P}_i)$. Since all corner points of the boundary of $\mathcal{R}$ are covered by $\mathcal{D}^*(\mathcal{P}_i)$, we know that p is not a corner point. So let a and b be the endpoints of the segment containing p on the boundary of $\mathcal{R}$. There is a point u on the segment linking a to p and a point v on the segment linking p to b such that u is covered by a disc $d_u \in \mathcal{D}^*(\mathcal{P}_i)$, v is covered by a disc $d_v \in \mathcal{D}^*(\mathcal{P}_i)$, and no other point on the segment linking u to v is covered by a disc in $\mathcal{D}^*(\mathcal{P}_i)$. Hence the edge of the Laguerre-Voronoi diagram that separates d_u and d_v intersects the segment linking u to v, which means that $W^*(\mathcal{P}_i)$ is not empty, a contradiction. Therefore, we can assume that all points on the boundaries of $\mathcal{R}$ are covered by $\mathcal{D}^*(\mathcal{P}_i)$. Suppose now that a point $p \in \mathcal{R}$ lies outside the boundaries of $\mathcal{R}$ and is not covered by $\mathcal{D}^*(\mathcal{P}_i)$. This means that the face of the Laguerre-Voronoi diagram that contains p has a zone Z not covered by $\mathcal{D}^*(\mathcal{P}_i)$. The point of Z that is the farthest from a boundary of a disc in $\mathcal{D}^*(\mathcal{P}_i)$ is a Laguerre-Voronoi vertex v. Since Z cannot cross the boundaries of $\mathcal{R}$ (because all points on the boundaries are covered by $\mathcal{D}^*(\mathcal{P}_i)$), vertex v belongs to $\mathcal{R}$, which implies $v \in V^*(\mathcal{P}_i)$ and $U^*(\mathcal{P}_i) \neq \emptyset$, a contradiction. □

Note that if $\mathcal{R}$ is finite (i.e., each one of the m regions in $\mathcal{R}$ contains a single point), then Algorithm 1 can be simplified. In such a case, $C = \mathcal{R}$ (at line 2), and there is no need to build the Laguerre-Voronoi diagram. Hence, lines 11-13 can be skipped, and line 14 can be replaced by : determine the set $U^*(\mathcal{P}_i)$ of points in $\mathcal{R}$ that are not covered by $\mathcal{D}^*(\mathcal{P}_i)$. Since $\mathcal{R}$ may contain an infinite number of points, it is not obvious that the proposed algorithm is finite. The next Property shows that this is the case.

Property 2 *Algorithm 1 terminates in at most* $2^{|\mathcal{D}|}$ *iterations.*

Proof. Since $\mathcal{D}^*(\mathcal{P}_{i-1})$ does not cover the points in $\mathcal{P}_i \setminus \mathcal{P}_{i-1}$, while $\mathcal{D}^*(\mathcal{P}_j)$ covers them for all $j \geq i$, we have $\mathcal{D}^*(\mathcal{P}_j) \neq \mathcal{D}^*(\mathcal{P}_{i-1})$ for all $j \geq i$. The number of iterations of the proposed algorithm is therefore bounded by the the number of subsets of $\mathcal{D}$. □

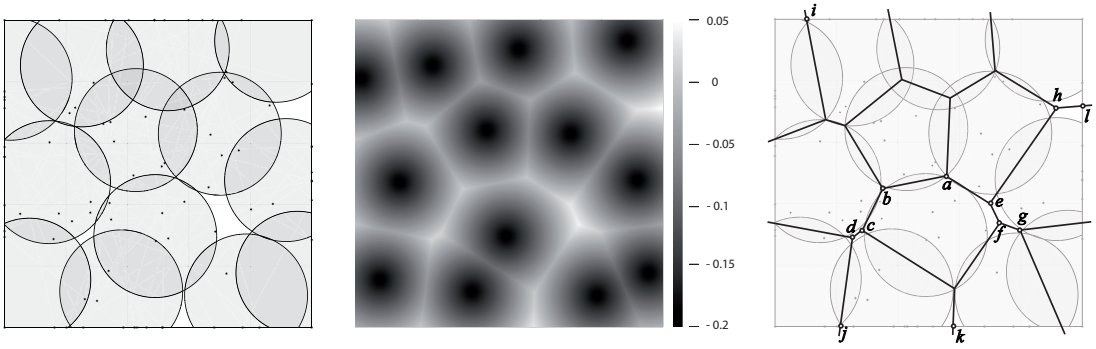
In Figure 3, we illustrate an iteration of the proposed algorithm for the covering of a 1×1 square $\mathcal{R}$ with discs of radius 0.2. We show in Figure 3(a) the points in the current set $\mathcal{P}_i$ as well as the 13 discs in $\mathcal{D}^*(\mathcal{P}_i)$ that cover them. For a point p in the square and a disc $d \in \mathcal{D}$, Let $\delta(p, d)$ be the distance from p to the boundary of d, and let $v(p, d)$ be defined as follows:

$$v(p,d) = \begin{cases} -\delta(p,d) & \text{if } p \text{ is covered by } d \\ \delta(p,d) & \text{otherwise.} \end{cases}$$

For every point p in the square, let us define $w(p) = \min_{d \in \mathcal{D}^*(\mathcal{P}_i)} v(p, d)$. These values are represented in Figure 3(b) with colors that change from white to black when $w(p)$ varies from 0.05 to -0.2. The centers p of the discs in $\mathcal{D}^*(\mathcal{P}_i)$ are the darkest points with value $w(p) = -0,2$, and we are looking for the clearest point with $w(p) > 0$. As shown in Figure 3(c), our algorithm determines a set $U^*(\mathcal{P}_i)$ that contains eight Laguerre-Voronoi vertices (points $a, \ldots, h$) and four points on the intersection of an edge of the Laguerre-Voronoi diagram and the boundary of the square (points i, j, k, l). The farthest point in $U^*(\mathcal{P}_i)$ from a boundary of a disc in $D^*(\mathcal{P}_i)$ is point l. Hence, we have determined that $\arg\max_{p \in \mathcal{R}} w(p) = \arg\max_{p \in U^*(\mathcal{P}_i)} w(p) = l$. If $s = 4$ (i.e., up to four points can be added to $\mathcal{P}_i$), then $\mathcal{P}_{i+1}$ is obtained from $\mathcal{P}_i$ by adding points l, j, i, a.

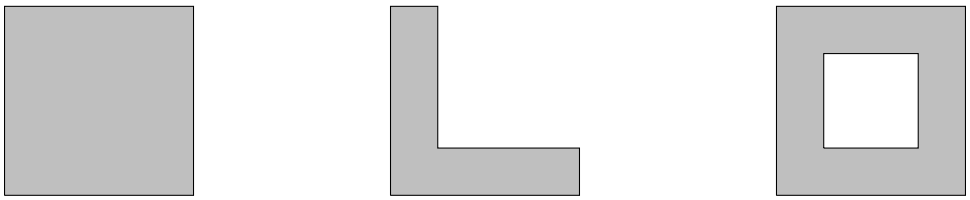
4 Computational experiments

To illustrate the effectiveness of the proposed method, we consider three different solid shapes, namely the `square`, the `L-shape` and the `ring`. The longest straight line of each shape has a unit length. The width of the `L-shape` and the `ring` is 0.25. These shapes are represented in Figure 4.



(a) (b) (c)

Figure 3: Illustration of an iteration of the algorithm.



(a) `square` (b) `L-shape` (c) `ring`

Figure 4: Three shapes.

For each shape, we consider five types of instances which differ according to the content of $\mathcal{D}$. Instances of the first type have 50 discs of radius $r = 0.3$, while there are 150 discs of radius $r = 0.2$ in the instances of the second type, 500 discs of radius $r = 0.1$ in those of the third type and 1,500 discs of radius $r = 0.05$ for the instances of the fourth type. The instances of the fifth type have 50 discs of radii picked uniformly in $\{0.05, 0.1, 0.2, 0.3\}$. The locations of the centers of the discs are picked at random from within the considered region $\mathcal{R}$.

We have created five instances of each type for each shape, which gives a total of 75 instances. Eleven of these instances are proven infeasible by our method in the sense that the discs in $\mathcal{D}$ do not allow to cover the considered region $\mathcal{R}$. This is done on purpose to assess the effectiveness of our algorithm when it has to prove that the available discs are not sufficient to cover the entire region. Also, it has determined an optimal solution for 47 instances, and a lower bound on the optimal value for the 17 remaining instances. Note that some of these 17 instances may not be feasible. All the instances that we used for our tests are available on the website http://claudio.contardo.org.

We have implemented our method in Julia 1.3, and executed it on a machine equipped with an Intel(R) Xeon(R) CPU E5-2637 v2 @ 3.50GHz with 128 GB of RAM, and running the Oracle Linux operating system. We have used Gurobi 8.1 as general-purpose MIP solver for the solution of the set-cover problems. The number of threads was fixed to one, and we have tested values in $\{1, 5, 10\}$ for parameter s. A time limit of one hour was given to our algorithm, after which the method was aborted and the problem deemed unsolved.

The results produced by our algorithm are reported in Tables 1, 2, and 3. For each type of instance, each shape, and each value of parameter s, we indicate in part (a) of these tables the number of proven infeasible solutions (#In), the number of proven optima (#Opt), and the average lower bound on the optimal value. This average lower bound is calculated by only considering the instances for which we have not been able to prove infeasibility. Hence, if #In+#Opt=5, then the average lower bound is the average optimal value of the feasible instances. In part (b) of Tables 1, 2, and 3, we analyze the proven infeasible instances and the proven optima by indicating the average number of iterations (#It) and the average time in seconds (CPU) needed to obtain these proofs. All other instances ran for one hour without our being able to determine an optimal solution or prove infeasibility.

We observe that the radius plays an important role on the efficiency of the method. The method is indeed much faster for large radii than for small ones. For example, consider the `square` and $s = 10$. Optimal solutions are found in 0.16 seconds with discs of radius 0.3, in 1.1 seconds with discs of radius 0.2, and no optimal solution is found with discs of radius 0.1 or 0.05. This is directly related to the average number of discs needed to cover the considered region $\mathcal{R}$ (reported under columns LB) and to the number of iterations needed to find an optimal solution. Indeed, the set-cover problem solved at line 5 of our algorithm has to cover s additional points at each iteration. When the optimal solution uses a very small number of discs, the algorithm finds it within a fraction of a second. As the size of the optimal solution increases, much more time is required. This confirms past observations from our experience developing relaxation-based exact methods for other classes of problems [2, 10, 12, 11].

Table 1: Results for the `square`.

s	$r = 0.3$			$r = 0.2$			$r = 0.1$			$r = 0.05$			random r		
	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB
1	0	5	8.4	0	5	15.2	0	0	49.2	2	0	143.0	5	0	-
5	0	5	8.4	0	5	15.2	0	0	52.6	2	0	161.7	5	0	-
10	0	5	8.4	0	5	15.2	0	0	53.2	2	0	165.7	5	0	-

(a) Proven infeasible instances, proven optima, and lower bounds.

s	Proven optima				Proven infeasible			
	$r = 0.3$		$r = 0.2$		$r = 0.05$		random r	
	#It	CPU	#It	CPU	#It	CPU	#It	CPU
1	42.0	0.59	183.0	10.4	158.0	28.5	8.2	0.07
5	11.0	0.12	33.4	1.3	15.5	1.4	3.0	0.03
10	10.6	0.16	26.2	1.1	16.5	2.4	2.8	0.03

(b) Number of iterations and CPU time for the proven optima and infeasible instances.

Table 2: Results for the `L-shape`.

s	$r = 0.3$			$r = 0.2$			$r = 0.1$			$r = 0.05$			random r		
	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB
1	0	5	4.8	0	5	7.6	0	5	25.2	0	0	71.4	0	5	5.4
5	0	5	4.8	0	5	7.6	0	5	25.2	0	0	75.2	0	5	5.4
10	0	5	4.8	0	5	7.6	0	5	25.2	0	0	77.8	0	5	5.4

(a) Proven infeasible instances, proven optima, and lower bounds.

s	Proven optima							
	$r = 0.3$		$r = 0.2$		$r = 0.1$		random r	
	#It	CPU	#It	CPU	#It	CPU	#It	CPU
1	12.6	0.26	45.4	0.85	771.6	668.9	11.2	0.27
5	5.2	0.05	13.8	0.40	105.6	84.9	5.0	0.04
10	5.2	0.05	11.0	0.25	57.4	48.4	4.8	0.06

(b) Number of iterations and CPU time for the proven optima.

To illustrate this further, we compare the behaviour of our algorithm on two instances of the covering of the `square` : the first instance is of the first type and therefore has 50 discs of radius 0.3 in $\mathcal{D}$, while the second instance is of the fourth type with 1,500 discs of radius 0.05. In Figure 5 we show, at each iteration i, the elapsed CPU time, the number of discs in $\mathcal{D}^*(\mathcal{P}_i)$ and the largest distance, denoted ℓ_i, from a point in $U^*(\mathcal{P}_i)$ to its closest boundary of a disc in $D^*(\mathcal{P}_i)$. We observe the following differences:

- There is a linear increase in the computation time of the first instance, which is due to the few iterations required by our algorithm to determine an optimal solution. The second instance is a much more difficult one since our algorithm does not succeed in determining an optimal solution in less than an hour. As you would expect, the growth in CPU time is then exponential;

Table 3: Results for the `ring`.

s	$r = 0.3$			$r = 0.2$			$r = 0.1$			$r = 0.05$			random r		
	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB	#In	#Opt	LB
1	0	5	7.6	0	5	13.0	1	4	43.75	0	0	120.8	2	3	10.33
5	0	5	7.6	0	5	13.0	1	4	43.75	1	0	130.5	2	3	10.33
10	0	5	7.6	0	5	13.0	1	4	43.75	1	0	135.5	2	3	10.33

(a) Proven infeasible instances, proven optima, and lower bounds.

s	Proven optima								Proven infeasible instances					
	$r = 0.3$		$r = 0.2$		$r = 0.1$		random r		$r = 0.1$		$r = 0.05$		random r	
	#It	CPU	#It	CPU	#It	CPU	#It	CPU	#It	CPU	#It	CPU	#It	CPU
1	27.6	0.54	83.2	2.3	871.75	1,207.1	21.00	0.36	2	0.08	-	-	1.0	0.01
5	8.4	0.10	18.0	0.5	122.25	222.9	6.00	0.07	2	0.02	12	0.8	1.0	0.01
10	6.8	0.07	14.8	0.5	64.25	57.5	4.67	0.08	2	0.02	18	2.4	1.0	0.01

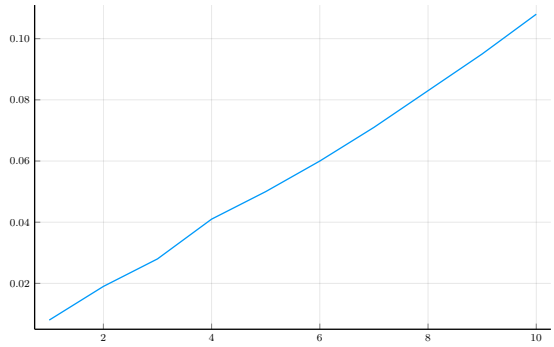
(b) Number of iterations and CPU time for the proven optima and infeasible instances.

- Less than a second is necessary to determine an optimal solution with 9 discs for the first instance, while the second instance remains unsolved after one hour of computation, with a lower bound of approximately 160 discs;
- Three iterations are needed with discs of radius 0.3 to decrease ℓ_i below the value 0.1, and 7 additional iterations are then sufficient to close the gap. For discs with radius 0.05, the decrease of ℓ_i below value 0.1 is also very fast (approximately 15 iterations), but 85 additional iterations are insufficient to close the gap.

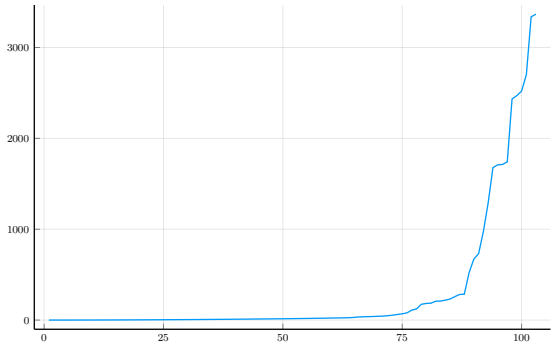
We next observe that parameter s plays an important role. Consider for example the `L-shape`, with 500 discs of radius 0.1: the average time needed to find an optimal solution increases from 48.4 seconds with $s = 10$ to 668.9 seconds with $s = 1$. As other example, no proof of infeasibility is obtained with $s = 1$ for the `ring` with 1,500 discs of radius 0.05, while such a proof is obtained in less than a second with $s = 5$. Also, when no proof of optimality is produced, the lower bound is typically much better with large values of s. For example, considering the `ring` with 1,500 discs of radius 0.05, the average lower bound increases from 120.8 with $s = 1$ to 135.5 with $s = 10$.

While the CPU time needed to solve an instance is of course proportional to the number of discs needed to cover the considered region $\mathcal{R}$, the shape of the region also has its importance. For example, 43.75 discs of radius 0.1 are needed, on average, to cover the `ring`, and this is found in about one minute, with $s = 10$. For comparison, our algorithm fails at finding a feasible solution within one hour of computation when trying to cover the `square` with discs of radius 0.1, the best bound being slightly larger than 50.

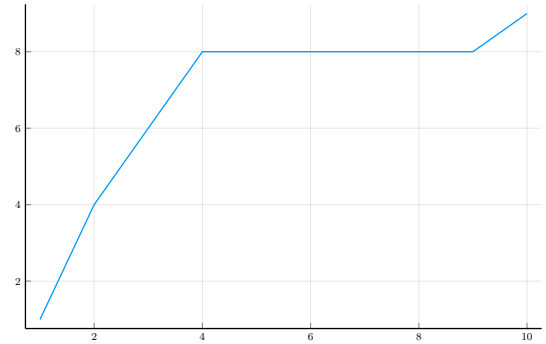
We finally evaluate the performance of our algorithm when it comes to demonstrate that an instance is not feasible. As can be observed in the Tables 1, 2, and 3, eleven instances out of 75 have a set $\mathcal{D}$ of discs which does not cover the entire considered region : 2 for the `square` with $r = 0.05$, 5 for the `square` with r randomly chosen in $\{0.05, 0.1, 0.2, 0.3\}$, 1 for the `ring` with $r = 0.1$, 1 for the `ring` with $r = 0.05$, and 2 for the `ring` with r randomly chosen in $\{0.05, 0.1, 0.2, 0.3\}$. We note that this proof of infeasibility is obtained very quickly, generally in a handful of iterations, which results in computation times below 3 seconds with $s = 10$. This excellent behavior is linked to the fact that at each iteration of our algorithm, we add points which are the furthest from the regions already covered, which pushes the method to cover the most critical points fairly quickly.



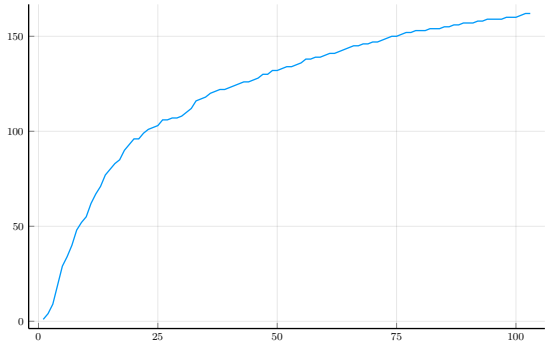
(a) CPU time for 50 discs of radius 0.3.



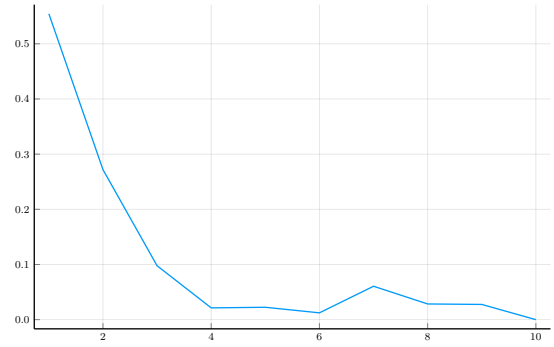
(b) CPU time for 1,500 discs of radius 0.05.



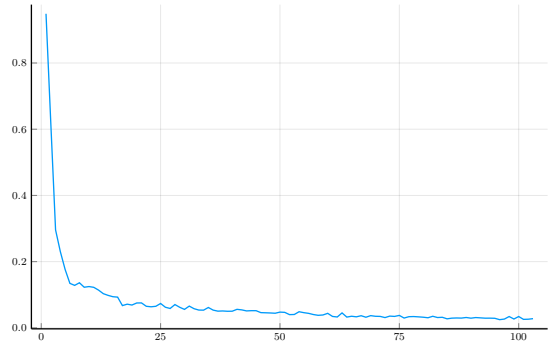
(c) $|\mathcal{D}^*(\mathcal{P}_i)|$ for 50 discs of radius 0.3.



(d) $|\mathcal{D}^*(\mathcal{P}_i)|$ for 1,500 discs of radius 0.05.



(e) ℓ_i for 50 discs of radius 0.3.



(f) ℓ_i for 1,500 discs of radius 0.05.

Figure 5: Comparison of two coverings of the `square`.

5 Conclusions

Given a region $\mathcal{R}$ in the 2-dimensional plane and given a set $\mathcal{D}$ of n discs with fixed centers and radii, we have proposed an algorithm for finding a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that every point in $\mathcal{R}$ is covered by at least one disc in $\mathcal{D}^*$. Our algorithm can be applied to any finite union of disjoint regions having polygonal boundaries. Hence, $\mathcal{R}$ is not necessarily convex, and it possibly contains holes.

This covering problem frequently occurs in telecommunications and security applications and is therefore relevant in practice. It has an infinite set of constraints and is therefore intractable by commercial solvers. The proposed algorithm decomposes this problem into two easier subproblems

that are solved in an iterative manner, namely a traditional set-cover problem with a finite set of constraints, and the construction of the Laguerre-Voronoi diagram of a circle set.

Future work should focus on possible extensions of our method to the use of other types of covering structures such as ellipses or squares, and to regions of non-polygonal shapes such as those with smooth (non-polygonal) boundaries.

References

[1] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. ACM Comput. Surv., 30 (4):412–458, Dec. 1998. ISSN 0360-0300. doi: 10.1145/299917.299918.

[2] D. Aloise and C. Contardo. A sampling-based exact algorithm for the solution of the minimax diameter clustering problem. Journal of Global Optimization, 71:613–630, 2018. doi: 10.1007/s10898-018-0634-1.

[3] C. Ambühl, T. Erlebach, M. Mihalák, and M. Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In J. Díaz, K. Jansen, J. D. P. Rolim, and U. Zwick, editors, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, pages 3–14, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[4] M. Basappa, R. Acharyya, and G. K. Das. Unit disk cover problem in 2d. J. Discrete Algorithms, 33: 193–201, 2015.

[5] A. Biniaz, P. Liu, A. Maheshwari, and M. Smid. Approximation algorithms for the unit disk cover problem in 2d and 3d. Comput. Geom. Theory Appl., 60(C):8–18, Jan. 2017. ISSN 0925-7721. doi: 10.1016/j.comgeo.2016.04.002.

[6] N. Boland, M. Hewitt, L. Marshall, and M. Savelsbergh. The continuous-time service network design problem. Operations Research, 65(5):1303–1321, 2017.

[7] P. Carmi, M. J. Katz, and N. Lev-Tov. Covering points by unit disks of fixed location. In T. Tokuyama, editor, Algorithms and Computation, pages 644–655, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-77120-3.

[8] T. M. Chan and E. Grant. Exact algorithms and apx-hardness results for geometric packing and covering problems. Comput. Geom. Theory Appl., 47(2):112–124, Feb. 2014. ISSN 0925-7721. doi: 10.1016/j.comgeo.2012.04.001.

[9] F. Claude, G. K. Das, R. Dorrigiv, S. Durocher, R. Fraser, A. López-Ortiz, B. G. Nickerson, and A. Salinger. An improved line-separable algorithm for discrete unit disk cover. Discrete Math., Alg. and Appl., 2:77–88, 2010.

[10] C. Contardo. Decremental clustering for the solution of p-dispersion problems to proven optimality. INFORMS Journal on Optimization, 2019. Forthcoming.

[11] C. Contardo and J. A. Sefair. A progressive approximation scheme for the exact solution of sparse large-scale binary interdiction games. Technical Report G–2019–49, Cahiers du GERAD, 2019.

[12] C. Contardo, M. Iori, and R. Kramer. A scalable exact algorithm for the vertex p-center problem. Computers & Operations Research, 103:211–220, 2019. doi: 10.1016/j.cor.2018.11.006.

[13] G. K. Das, R. Fraser, A. Lòpez-Ortiz, and B. G. Nickerson. On the discrete unit disk cover problem. In N. Katoh and A. Kumar, editors, WALCOM: Algorithms and Computation, pages 146–157, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-19094-0.

[14] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are np-complete. Information Processing Letters, 12(3):133–137, 1981. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(81)90111-3.

[15] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and vlsi. J. ACM, 32(1):130–136, Jan. 1985. ISSN 0004-5411. doi: 10.1145/2455.214106.

[16] D. Kim, D. Kim, and K. Sugihara. Voronoi diagram of a circle set constructed from voronoi diagram of a point set. In Algorithms and Computation, 11th International Conference, 2000, Taipei, Taiwan, December 18-20, 2000, Proceedings, pages 432–443, 2000. doi: 10.1007/3-540-40996-3_37.

[17] R. Ko. The complexity of the minimum sensor cover problem with unit-disk sensing regions over a connected monitored region. IJDSN, 8, 2012. doi: 10.1155/2012/918252.

[18] P. Liu and D. Lu. A fast 25/6-approximation for the minimum unit disk cover problem. CoRR, abs/1406.3838, 2014. URL http://arxiv.org/abs/1406.3838.

[19] S. Masuyama, T. Ibaraki, and T. Hasegawa. The computational complexity of the m -center problems on the plane. IEICE Transactions, E64(2), 1981.

[20] N. H. Mustafa and S. Ray. Improved results on geometric hitting set problems. Discrete & Computational Geometry, 44(4):883–895, Dec 2010. ISSN 1432-0444. doi: 10.1007/s00454-010-9285-9.

[21] S. Narayanappa and P. Vojtechovský. An improved approximation factor for the unit disk covering problem. In CCCG, 2006.