Noname manuscript No.
(will be inserted by the editor)

On Interval and Circular-Arc Covering Problems

Reuven Cohen · Mira Gonen

Received: date / Accepted: date

Abstract In this paper we study several related problems of finding optimal interval and circular-arc covering. We present solutions to the maximum k-interval (k-circular-arc) coverage problems, in which we want to cover maximum weight by selecting k intervals (circular-arcs) out of a given set of intervals (circular-arcs), respectively, the weighted interval covering problem, in which we want to cover maximum weight by placing k intervals with a given length, and the k-centers problem. The general sets version of the discussed problems, namely the general measure k-centers problem and the maximum covering problem for sets are known to be NP-hard. However, for the one dimensional restrictions studied here, and even for circular-arc graphs, we present efficient, polynomial time, algorithms that solve these problems. Our results for the maximum k-interval and k-circular-arc covering problems hold for any right continuous positive measure on $\mathbb{R}$.

1 Introduction

This paper presents several efficient polynomial time algorithms for optimal interval and circular-arc covering. Given a set system consisting of a ground set of points with integer weights, subsets of points, and an integer k, the *maximum coverage problem* selects up to k subsets such that the sum of weights of the covered points is maximized. We consider the case that the ground set is a set of points on a circle and the subsets are circular-arcs, and also the case in which the ground set is a set of points on the real line and the subsets are intervals. The *maximum coverage by placing intervals problem* is a variant of the maximum coverage problem in which there is a given length of all the intervals, the intervals are not given in the input and can be placed to

R. Cohen
Department of Mathematics,
Bar-Ilan University,
Ramat-Gan 52900, Israel,
E-mail: reuven@math.biu.ac.il

M. Gonen
Department of Computer Science,
Ariel University,
Ariel, Israel,
E-mail: mirag@ariel.ac.il

maximize the weight of the cover. A generalization of the problem to the problem of maximum covering of a general measure by a given number of intervals (or a given number of circular-arcs) is discussed as well. Finally, the *k-center problem* is the following: given a set P of points in a measure space and a positive integer k, find a set of k supply points such that the maximum distance between a point in P and its nearest supply point is minimized. For the cases of the L_2 and L_∞-measure the problem is referred to as the Euclidean and rectilinear k-center problem, respectively.

Maximal cover problems arise naturally in many optimization problems. The one-dimensional analog presented here can be used, for example, to find the optimal locations along a road for positioning fuel stations, cellular communication towers or different facilities, given some measure along the road such as traffic density, the locations of farms or the expected revenue.

The maximum coverage problem on the real line and on circular-arc graphs is obviously a specific instance of the maximum coverage problem, and is related to the set cover problem. Minimum set cover of integers on the real line was discussed by Hochbaum and Levin [24]. They showed that packing and covering optimization problems over constraints in consecutive and circular 1s have efficient polynomial algorithms. Specifically, Hochbaum and Levin showed that the set cover problem, where the sets are restricted to be continuous intervals of integers can be solved in linear time. Circular-arc graphs are a strict generalization of interval graphs. There are several problems, such as graph coloring, that have a polynomial-time solution for interval graphs, but do not admit an efficient algorithm, unless P=NP, for circular-arc graphs [14].

The maximum coverage problem on the real line is a special case of geometric covering problems. Geometric covering problems have been extensively studied as they have applications to real-world engineering and optimization problems. It has been shown [25] that even for very simple classes of objects such as unit disks or unit squares in the plane, computing the exact minimum set cover is strongly NP-hard. As a result, much of the research regarding geometric set cover has focused on approximation algorithms, specifically of low VC-dimension [6, 17]. The approximability of the rectangle cover problem and its generalization, the 4-sided box cover problem, have connections to related capacitated covering problems [8]. Mustafa and Ray [32] proposed a PTAS for a wide class of unweighted geometric hitting set and set cover problems via a local search technique. Erlebach and van Leeuwen [16] have obtained a PTAS for the weighted version of geometric set cover for the special case of unit squares. Chan and Grant [10] studied several geometric set cover and set packing problems involving configurations of points and geometric objects in Euclidean space. They proved the APX-hardness of computing a minimum cover of a set of points in the plane by a family of axis-aligned fat rectangles and several other classes of objects. In addition they gave a polynomial-time dynamic programming algorithm for geometric set cover for some classes of objects. Carmi et al. [7] discussed the problem of covering a given set of points in the plane by a given set of unit disks. They showed that while this special case of geometric set cover is still NP-hard, it can be approximated within a constant factor of 38. Gonzalez [22] discussed several variants of covering a set of points in a given space. He considered covering a set of points in the plane by fixed-size orthogonal squares, covering points in d-dimensional space by orthogonal hypersquares with a given dimension, covering points in d-dimensional space by fixed-size hyperrectangles with given dimensions, and covering points by hyperdiscs with a given diameter. These problems are all known to be NP-hard for $d \geq 2$ [39, 29, 19]. Gonzalez presented a fast approximation algorithm that generates provably good solutions and an improved polynomial-time approximation scheme for these problems.

The general problems of set cover and maximum k coverage are well known to be NP-hard [20, 33, 18]. The set cover problem can be approximated within $\ln n - \ln \ln n + \Theta(1)$ [37, 28, 38].

By [36, 3] it follows that unless P = NP, there exists a constant $0 < c < 1$ so that set cover cannot be efficiently approximated to within any number smaller than $c \log_2 n$. Feige [18] has shown hardness of approximating set cover in $(1 - o(1)) \ln n$. According to Feige [18] for all $\epsilon > 0$, the maximum k coverage problem cannot be approximated in polynomial time within a ratio of $1 - 1/e + \epsilon$, unless P = NP. In addition the greedy algorithm (iteratively selecting the sets that cover the largest number of yet uncovered elements) approximates the maximum k coverage problem within a ratio of at least $1 - 1/e$.

A variation of the maximum k coverage problem is the following: given a family of subsets we want to find a subcollection of size of at most k in order to maximize the number of subsets having nonempty intersections with the subcollection. It is well known that the decision version of this problem is NP-hard, and that a simple greedy algorithm approximates the problem within a factor of $1 - (1 - 1/k)^k$. Feige [18] proved that no polynomial time algorithm can have a better performance guarantee unless P = NP. Moreover, Cornuejols, Nemhauser and Wolsay [13] showed that the greedy algorithm almost always finds an optimal solution in the case of two-element sets. Ageev and Sviridenko [2] presented an approximation algorithm that gives a factor of $1 - (1 - 1/p)^p$, where p is the maximal size of a subset. In addition, the problem still remains NP-hard for any fixed $p \geq 2$. In the case when $p = 2$ the performance guarantee of the greedy algorithm has the same value of $1 - 1/e$ [13].

The general minimum k-center problem is approximable within 2 [21, 23], and is not approximable within $2-\varepsilon$ for any $\varepsilon > 0$ [26, 34]. The variation where the maximum distance from each vertex to its center is given by some L and the number of centers is to be minimized, is approximable within $\log L + 1$ [4]. The vertex weighted version, where the objective is to minimize the maximum weighted distance is approximable within 2 [35]. Efficient polynomial-time algorithms have been found for the planar k-center problem when k is a small constant [9, 15]. Also, the rectilinear 2-center problem has a polynomial-time solution, even when d is part of the input [30]. However both the Euclidean and rectilinear problems are NP-hard for $d = 2$ when k is part of the input [19, 31], while the Euclidean 2-center and rectilinear 3-center are NP-hard when d is part of the input [30]. Hwang et al. [27] gave an $n^{O(\sqrt{k})}$-time algorithm for the Euclidean k-center problem in the plane. Agarwal and Procopiuc [1] presented an $n^{O(k^{1-1/d})}$-time algorithm for solving the k-center problem in R^d under L_2 and L_∞-measures, and a $(1 + \varepsilon)$-approximation algorithm with running time $O(n \log k) + (k/\varepsilon)^{O(k^{1-1/d})}$. Brass et al. [5] considered several instances of the k-center problem on a line, namely, the centers lie on a line ℓ. They first discussed a version of the problem where the line ℓ is given in advance, and gave an $O(n \log^2 n)$ algorithm. In addition, they investigated the cases where only the orientation of the line ℓ is fixed, and where the line ℓ can be arbitrary. They presented $O(n^2 \log^2 n)$ and expected $O(n^4 \log^2 n)$ algorithms, respectively. In a recent paper [11] Chen and Wang presented an efficient algorithm for the weighted k-center problem on a real line. Their definition of the target function is different than ours, as will be discussed in Section 5.

The general exact k-coverage problem for the special cases of intervals and circular-arcs was discussed by Cohen et al. [12]. This problem is a generalisation of the problem discussed in this paper, in which each point i has an integer demand d_i which is not necessarily 1, and point i is covered if exactly d_i subsets containing it are selected. Cohen et al. [12] studied this problem and some related optimization problems. They proved that the exact k-coverage problem with unbounded demands is NP-hard even for intervals on the real line and unit weights. parameters is In addition the authors proved that if any parameter of the problem is restricted to be a constant, then the problem is polynomial time solvable. The case where each demand is exactly 1, and a point can be covered by more intervals than its demand is not considered in [12]. where one of the parameters is a constant).

Our Results. In this paper we consider several related problems of finding optimal interval and circular-arc covering. We discuss maximum coverage of points by intervals, maximum coverage of points by circular-arcs, maximum coverage by placing intervals, and one dimensional k-centers. We provide polynomial time algorithms, based on dynamic programming. The variants we discuss were not discussed before. For the maximum coverage problems for interval and circular-arc graphs when the intervals and circular-arcs are given, our results hold for any right continuous positive measure. For interval graphs we present an algorithm with time complexity of $O(n \cdot k^2 + n \log n + M)$, where n is the number of given intervals, and M is the time complexity of computing the intervals' measure. For the special case that the measure is the sum of weights of given m points on the real line covered by the intervals, our algorithm has time complexity of $O(n \cdot k^2 + n \log n + m \log m)$. For circular-arc graphs we present an algorithm with time complexity of $O(n^2 \cdot k^2 + n^2 \log n + M \cdot n)$, where n is the number of given circular-arcs, and M is the time complexity of computing the arcs' measure. For the maximum coverage by placing intervals problem we introduce an $O(m \cdot n)$ algorithm, where m is the number of points and n is the number of intervals. Finally, we present an $O(n^2 \log n)$ algorithm for the one dimensional k-centers problem, where n is the number of points on the real line.

In the following we assume, as customary, that mathematical operations on integers and reals including comparisons and storage require constant time. If one assumes that these operations require time proportional to the number of digits appropriate adjustments to the results is needed.

Organization: In Section 2 we design an efficient algorithm for the interval maximum coverage problem. The solution uses an algorithm for the weighted independent set with a cost for intervals. In Section 3 we generalize the algorithm to circular-arc maximum coverage problem by reducing the arcs to intervals. In Section 4 we introduce an $O(m \cdot n)$ algorithm for the problem of covering maximum weight of n points by m intervals, and then present an algorithm for covering general measure. In Section 5 we show an $O(n^2 \log n)$ algorithm and an $O(n \cdot k)$ algorithm for two variants of the one dimensional k-centers problem. We conclude with Section 6.

2 Maximum coverage by given intervals

In this section we show an efficient algorithm for the interval maximum coverage problem. The solution uses an algorithm for the weighted independent set with a cost for intervals.

Consider the following problem:

Problem 1 real interval maximum coverage: *for any right continuous positive measure, μ on $\mathbb{R}$, given a collection of n real intervals, $\mathcal{I} = \{I_1, \ldots, I_n\}$, $I_i = (a_i, b_i]$, find a subset $\mathcal{J} \subseteq \mathcal{I}$ of cardinality K such that $\mu(\mathcal{J}) := \mu(\cup_{I \in \mathcal{J}} I)$ is maximum. Other types of intervals, such as closed or open intervals may also be considered, at the price of treating startpoints and endpoints differently. For simplicity we consider only open-closed intervals. We refer to $\mu(\mathcal{J})$ as* maximum cover. *Examples include:*

1. *Given a universe $P = \{p_1, \ldots, p_m\}$ of m integers or reals with weights $\{w_1, \ldots, w_m\}$, and a collection of n real intervals, $\mathcal{I} = \{I_1, \ldots, I_n\}$, $I_i = (a_i, b_i]$, find a subcollection $\mathcal{I}'$ of $\mathcal{I}$ of size K such that $\max_{i=1}^{n} w_i \cdot \delta_i$ is maximum, where $\delta_i = 1$ if $\cup_{I_j \in \mathcal{I}'} I_j$ covers element p_i. This problem can also be formulated by considering a list of intervals ending at the points p_i. However, the solution in Sec. 4 is more efficient since one needs not consider intersecting intervals.*
2. *Find the subset $\mathcal{J} \subseteq \mathcal{I}$ such that $|\mathcal{J}| = K$ and the L^p norm of $\cup_{J \in \mathcal{J}} J$ is maximum.*

Note that the integer interval problem is reducible to the real interval problem with any integer interval $\{a, a+1, \ldots, b\}$ replaced by the real interval $(a-1/2, b+1/2]$. Therefore the integer interval maximum coverage problem can be solved using the same algorithm designed for the real interval maximum coverage problem.

Given a sorted vector of N real points, A, the following function calculates the vector $\mu((x, A_i])$ for $1 \leq i \leq N$, for some arbitrary $x \leq A_i$. This allows the evaluation of $\mu((a, b]) = \mu((x, b]) - \mu((x, a])$. Denote the time required to complete function *CalcMeasure* by M.

For the specific case described in Item 1 of Problem 1 the function *Function CalcMeasure* given bellow calculates the weights up to a given point on the line, namely the sum of weights of points covered up to a given point on the line.

Function CalcMeasure //For problem 1.1
Input: a vector of reals x of length N and the universe $P = \{p_1, \ldots, p_m\}$ of real points, and a set of weights $\{w_1, \ldots, w_m\}$.

1. $j \leftarrow 0$.
2. $W \leftarrow 0$.
3. For $i = 1$ to N
   (a) While $p_{j+1} < x_i$ do $j \leftarrow j + 1$, $W \leftarrow W + w_j$.
   (b) $y_i = W$.
4. return y. //For this measure $\mu((-\infty, x]) = \sum_{\{r|p_r<x\}} w_r$.

The time complexity of computing this function is $M = O(N + m \log m)$ time to complete, or $M = O(N + m)$ if the universe P is already sorted.

For the specific case described in Item 1 of Problem 2 the function *Function CalcMeasure* is given bellow:

Function CalcMeasure //L^p version
Input: a vector of reals x.
return x. //For the L^p measure $\mu((0, x]) = x$.

The function *CalcMeasure* in this case is actually a stub, requiring no time to complete.

We now introduce an efficient algorithm for the real interval max coverage problem. Consider the following high-level description of the algorithm. First we take the instance of max coverage that we have, i.e., a set of real intervals, and create another instant of real intervals in the following way: first we remove all intervals that are subsets of other intervals. Then, for every pair of real intervals $(a, b]$, $(c, d]$ such that $a < c \leq b < d$, we add the real interval $(a, d]$. Then we assign integer cost for each real interval in the following way: the cost of a real interval is the minimum number of intervals in the original instance which are needed to (fully) cover the real interval. We show how to efficiently compute the cost of each of these intervals. We also assign a weight for each real interval using the integrals on the measure. We then find a maximum weight independent set of the new instance of intervals of cost at most K.

Computing the new instance of intervals and their costs and weights: For any pair of intervals $(a, b], (c, d]$ such that $a < c \leq b < d$ Algorithm 1 computes the cost and weight of the new interval $(a, d]$, without explicitly keeping the interval itself. The algorithm computes the weights by using the function *CalcMeasure*, and computes the costs by computing maximum continuous cover. Namely, for each endpoint b_i of an interval $I_i = (a_i, b_i] \in \mathcal{I}$ and cost h, it computes the index of the minimum endpoint that is covered by at most h intervals that also cover b_i, such that each

pair of consecutive intervals intersect. Sort the list $\mathcal{I}$ by ascending order of b_i. Define the vector B as the vector of distinct endpoints (excluding duplicates) and the vector S as the vector of intervals' indexes in the following manner. $S_i = i$ and $B_i = b_i$ if the interval $(a_i, b_i]$ is not a sub-interval of any other interval, and else $S_i = j$ and $B_i = b_j$, where j is the smallest index of an interval $(a_j, b_j]$ that is a super-interval of $(a_i, b_i]$ and is not a subinterval of any other interval. Let A and B be the vectors of startpoints and endpoints, respectively, where $A_i = a_j$, $B_i = b_j$, and $j = S_i$. Define the matrix $C_{i,h}$ whose entries are indices of minimum endpoint covered by a continuous covering containing also the ith smallest endpoint using at most h intervals.

We use the following algorithm for computing the costs and the weights of the new instance of intervals. Namely, the algorithm computes the vectors S, A, B, and their weights WA, WB, and the matrix C.

Algorithm 1 (Cost-Weight)
Input: $\mathcal{I}, K$.

1. *Sort the intervals by ascending order of the* b_i.
2. *Set* $j \leftarrow 0$.
3. *For all* i *set* $A_i \leftarrow \infty$.
4. *For* $i = 1$ *to* n
   (a) *If* $j = 0$ *or* $b_i > B_j$
      i. *Set* $j \leftarrow j + 1$.
      ii. *Set* $B_j \leftarrow b_i$.
      iii. *Set* $A_j \leftarrow a_i$, $S_j \leftarrow i$.
      // Whenever a new endpoint is encountered set S_j *to be the interval's index and* A_j *and* B_j *to be the respective start and endpoint.*
5. *For* $i = j - 1$ *down to 1 do if* $A_i \geq A_{i+1}$ *set* $A_i \leftarrow A_{i+1}$, $S_i \leftarrow S_{i+1}$.
6. $h = 1$.
7. *For* $i = 1$ *to* j *do*
   (a) $C_{i,0} \leftarrow i$.
   (b) *While* $A_i > B_h$ *set* $h \leftarrow h + 1$.
   (c) $C_{i,1} \leftarrow h$.
8. *For* $h = 2$ *to* K
   (a) *For* $i = 1$ *to* j *set* $C_{i,h} \leftarrow C_{C_{i,h-1},1}$.
9. $WA = CalcMeasure(A)$, $WB = CalcMeasure(B)$.
10. *Return* S, A, B, C, WA, WB.

Theorem 1 *Algorithm 1 finds the minimum startpoint continuous covering for each endpoint* i *and cost* j *in time* $O(nK + n \log n + M)$

Proof: The proof is by induction on the cost. For cost 1 the algorithm gives the minimum startpoint continuous cover for each point i by finding the interval containing endpoint i with minimum startpoint. For any cost h the correctness follows from the correctness of the algorithm for cost $h-1$ by extending this cover by the smallest startpoint single interval that contains the leftmost endpoint in the cover with cost $h-1$.

The time complexity of all steps is linear, except for the sorting which takes $O(n \log n)$ steps and step 8a which is performed $O(nK)$ times. Step 9 requires $O(M)$ operations, depending on the measure.

Corollary 1 *For problem 1.1 the time complexity of Algorithm 1 is* $O(nK + n \log n + m \log m)$.

We now design our algorithm for max coverage.

Let $D(i,h)$ be the cost of the ith interval if it is in the cover, for the intervals $I_1,\ldots,I_i$, with maximum cost h. Let $W(i,h)$ be the sum of weights of the intervals in the cover for the intervals $I_1,\ldots,I_i$, with maximum cost h. The algorithm stores the $D(i,h)$ and $W(i,h)$ values in a table, that is, a two dimensional array, $D(0\ldots n,0\ldots K)$, $W(0\ldots n,0\ldots K)$, whose entries are computed in a row-major order. That is, the first row of D and W is filled in from left to right, then the second row, and so on. At the end of the computation, $W(n,K)$ contains the weight of the optimal solution. The algorithm then outputs the optimal solution as a set, M, computed using $D(0\ldots n,0\ldots K)$.

Algorithm 2 (Dynamic weighted coverage)
Inputs: vectors A, B, WA,WB, S, matrix C, integer K.

1. $D(0,0)\leftarrow 0$, $W(0,0)\leftarrow 0$.
2. *For* $h=1$ *to* K
   (a) *set* $D(0,h)\leftarrow 0$, $W(0,h)\leftarrow 0$.
   (b) *For* $i=1$ *to* n
      i. $W(i,h)\leftarrow \max\{W(i-1,h), \max_{1\le t\le h}\{WB_i - WA_{C_{i,t-1}} + W(C_{i,t}-1, h-t)\}\}$.
      //check if the interval $(a_{C_{i,h-1}}, b_i]$ *should be added to the cover.*
      ii. *If* $W(i,h)=W(i-1,h)$ *set* $D(i,h)\leftarrow 0$.
      Else, $D(i,h)\leftarrow \arg\max_{1\le t\le h}\{WB_i - WA_{C_{i,t-1}} + W(C_{i,t}-1,h-t)\}$.
3. *Let* $i=n$, $h=K$, $M=\phi$.
4. *While* $i>0$ *do*
   (a) $c\leftarrow D(i,h)$.
   (b) *If* $c>0$ *do*
      i. *For* $t=0$ *to* $c-1$ *do*
         A. $M\leftarrow M\cup\{S_{C_{i,t}}\}$.
      ii. $i\leftarrow C_{i,h-1}$, $h\leftarrow h-c$.
   (c) $i\leftarrow i-1$
5. *Output* M.

Theorem 2 *Algorithm 2 computes maximum cover with cost at most K with time complexity* $O(n\cdot K^2 + n\log n + M)$.

Proof: The correctness of the Algorithm 2 follows by induction and from the claim that Algorithm 1 gives the best continuous coverage for points up to i with cost h (Theorem 1). Every coverage consists of a disjoint union of continuous intervals. For $n=1$ and every cost K the algorithm returns a cover of the point with cost K. Now assume that the algorithm gives the best solution up to cost K for all points up to n. Then, the best cover for $n+1$ points with cost K is either the best cover of the first n points with cost K or some cover containing the $n+1$st point.

If the optimal cover contains the $n+1$st point it must consist of a union of a continuous cover of some points including the $n+1$st point of some cost t and some cover of other points (possibly 0 points) with cost $K-t$. All such covers for all possible values of t are considered, such that the cover consists of a union of a maximum continuous cover of cost t up to and including point $n+1$ and an optimal cover up to (and not necessarily including) the first endpoint not covered by the continuous cover.

We now turn to show that if the optimal cover up to point $n+1$ includes point $n+1$ it necessarily consists of a disjoint union of an optimal maximum cover up to point $n+1$ (for some

cost t) and the optimal cover up to the first endpoint not covered by this continuous cover. For any given t, assume that the rightmost continuous part of optimal cover ends at point $n+1$ has cost t and is not an optimal continuous cover. Then, the total weight of the cover can always be improved by replacing this continuous interval by the optimal continuous interval of the same cost, as it will cover at least all points covered by this cover[1]. Now, assume that the left part of the cover (up to the last point not covered by the rightmost continuous part) is non optimal, then changing it to the optimal will improve the total weight of the cover, contradicting the assumption of optimality. Thus, the optimal cover is always in one of the cases checked by the algorithm.

The time complexity stems from the minimum calculation requiring $\Theta(K)$ steps. This step is performed $\Theta(nK)$ times. The complexity of the construction of M from the matrix D is $O(n+K)$, as the outer loop is performed at most n times and the inner loop is performed at most K times.

Thus, the total cost of the calculation is $O(nK^2 + n\log n + M)$.

3 Maximum coverage by given arcs

In this section we show an efficient algorithm for the maximum coverage problem on circular-arc graphs. The solution uses the algorithm for the interval maximum coverage problem given in the previous section.

Consider the maximum coverage problem for circular-arc graphs:

Problem 2 circular-arc maximum coverage: *for any right continuous positive measure, μ, given a circular-arc graph with n arcs, $\mathcal{A} = \{A_1, \ldots, A_n\}$, $A_i = \widehat{a_i b_i}$, where $a_i \notin A_i$, $b_i \in A_i$, find a subset $\mathcal{J} \subseteq \mathcal{A}$ of cardinality K such that $\mu(\mathcal{J}) := \mu(\cup_{A\in\mathcal{J}} A)$ is maximum. We refer to $\mu(\mathcal{J})$ as* maximum cover.

We now present an algorithm for circular-arc graphs that is based on the algorithm for interval graphs given in the previous section.

The idea of the algorithm is the following: first remove all arcs that are sub-arcs of arcs in $\mathcal{A}$. Then, for each arc $j \in \mathcal{A}$ find an optimal solution that contains arc j. Since no arc strictly contains arc j, for each other arc $i \in \mathcal{A}$ it holds that $i \setminus j$ is a single continuous arc which is a sub-arc of the arc $C \setminus j$, where C is the circle. Thus, for each such $j \in \mathcal{A}$ one can compute an optimal solution with $K-1$ arcs in the set $\{i \setminus j | i \in \mathcal{A}\}$. For computing such optimal solution use analogs of Algorithms 1 and 2, where $\{i \setminus j | i \in \mathcal{A}\}$ is the set of arcs and $K-1$ is the number of arcs need to be selected. Notice that for each j the arcs in $\{i \setminus j | i \in \mathcal{A}\}$ can be easily modified to intervals. Taking the optimal solution over all j, this algorithm will always find an optimal solution.

Algorithm 3 (Max cover of arcs)

1. *Remove from $\mathcal{A}$ all arcs that are sub-arcs of other arcs in $\mathcal{A}$.*
2. *For all $j \in \mathcal{A}$*

[1] Notice that this replacement may lead to an intersection with intervals to the left. However, this does not affect the argument, as there still exists a better cover, possibly for a different t

(a) $\mathcal{I}_j \longleftarrow \mathcal{A}$.
(b) For all $i \in \mathcal{I}_j$ *do* $i \longleftarrow i \setminus j$.
(c) Remove all arcs in $\mathcal{I}_j$ *that are sub-arcs of arcs in* $\mathcal{I}_j$.
(d) $J_j \longleftarrow DynamicWeightedCoverage(Cost - Weight(\mathcal{I}_j, K-1), K-1)$.
3. $j_{\max} \longleftarrow \arg\max_{j \in \mathcal{A}} (\mu(J_j) + \mu(j))$.
4. Return $J_{j_{\max}} \cup \{j_{\max}\}$.

Theorem 3 *Algorithm 3 returns an optimal weighted cover, and its time complexity is* $O(n^2 \cdot K^2 + n^2 \log n + M \cdot n)$.

Proof: Proving correctness: let $\mathcal{A}_{opt} \subseteq \mathcal{A}$ be an optimal solution, and let $\mathcal{A}_{alg}$ the solution returned by Algorithm 3. Let $j \in \mathcal{A}_{opt}$ be some arbitrary arc. Then,

$$\begin{aligned}\mu(\mathcal{A}_{opt}) &= \mu\left(\cup_{i \in \mathcal{A}_{opt}} i\right) = \mu\left(\left(\cup_{i \in \mathcal{A}_{opt} \setminus \{j\}} i \setminus j\right) \cup j\right) \\ &= \mu\left(\cup_{i \in \mathcal{A}_{opt} \setminus \{j\}} i \setminus j\right) + \mu(j) \leq \mu\left(J_j\right) + \mu(j) \leq \mu(\mathcal{A}_{alg})\end{aligned}$$

(The inequality in the second line follows from the fact that for each $i \in \mathcal{A}_{opt}$ $i \setminus j$ is a sub-arc in the circle minus arc j, so it is considered by Algorithm 3, and by the correctness of Algorithms 1 and 2).

The time complexity of the algorithm is $O(n^2 \cdot K^2 + n^2 \log n + M \cdot n)$, since the time complexity of Algorithms 1 and 2 is $O(n \cdot K^2 + n \log n + M)$, and these algorithms are run $O(n)$ times by Algorithm 3.

4 Maximum coverage by placing intervals

In this section we introduce an $O(mn)$ algorithm for the problem of covering maximum weight of m points by n intervals of a given length ℓ, where the intervals are not given in the input. Unlike the problem discussed in the previous section, the intervals can be placed to maximize the weight of the cover. We then modify this algorithm to solve the problem of maximum covering of a general measure by a given number of intervals. The above problem is different from the problem in Section 2 since the intervals are not given, and are all of the same length.

4.1 Covering points using intervals

Consider the following problem.

Problem 3 *Given a real number,* ℓ*, a natural number,* n*, and a set of* m *points,* $P = \{p_1, \ldots, p_m\}$*, with weights* $\{w_1, \ldots, w_m\}$*, find the optimal location of* n *intervals of length* ℓ *that cover the maximum possible total weight. Namely, find a location of the* n *intervals of length* ℓ *such that* $\sum_{i=1}^{m} w_i \cdot \delta_i$ *is maximum, where* $\delta_i = 1$ *if* p_i *is covered and otherwise* $\delta_i = 0$.

As in the previous section we refer by "maximum cover" to the maximum sum of weights of the points covered by the ℓ intervals.

Denote by I_i the interval $I_i = [p_i - \ell, p_i]$. We mark by $D(a, b)$ a set of a intervals that covers the maximum possible weight of points out of the leftmost b points. For a set of intervals X

define $W(X) = \sum_{t \in \{j | p_j \in P \cap \cup_{I_i \in X} I_i\}} w_t$, i.e., $W(X)$ is the total weight of points covered by the intervals in X. To evaluate these quantities we will define matrices D, O that will be used by the algorithm. $O_{a,b}$ will contain the weight of the optimal cover of points up to p_b using a intervals and $D_{a,b}$ will be 1 if this optimal cover contains the interval s_b and 0 otherwise.

The suggested algorithm is the following:

Algorithm 4 (Covering points using intervals)

1. *$TW(0) = 0$. For $i = 1$ to m, $TW(i) = TW(i-1) + w_i$.*
   // $TW(i) =$ Total weight of all points up to p_i.
2. *Set $q = 0$. For each $i \in \{1, \ldots, m\}$ while $p_{q+1} < p_i - \ell$ do $q \leftarrow q + 1$. Set $q_i = q$, $W(i) = TW(i) - TW(q_i)$.*
   // $q_i = \max(0, \max\{j | p_j < p_i - \ell\})$
   // $W(i) =$ weight of points covered by an interval ending at point i.
3. *Set $O_{a,0} = 0$, $a = \{0, \ldots, n\}$.*
4. *Set $O_{0,b} = 0$, $b = \{0, \ldots, m\}$.*
5. *For each $a = \{1, \ldots, n\}$, and for each $b = \{1, \ldots, m\}$ if $W(b) + O_{a-1,q_b} > O_{a,b-1}$ set $D_{a,b} = 1$, $O_{a,b} = W(b) + O_{a-1,q_b}$ else set $D_{a,b} = 0$, $O_{a,b} = O_{a,b-1}$.*
6. *Return the set $D(n,m)$. (for a pair a, b, if $D_{a,b} = 0$ then $D(a,b) = D(a,b-1)$ and otherwise $D(a,b) = \{s_b\} \cup D(a-1, q_b)$).*

To prove the correctness of the algorithm we first present the following lemma.

Lemma 1 *There exists an optimal cover such that:*

1. *Every interval ends at a point in P.*
2. *The intervals are pairwise disjoint.*

Proof: For every cover one may move every interval $[a - \ell, a]$ to end at the rightmost point which it intersects, i.e., to move it to $[p - \ell, p]$, such that $p = max\{p_i | p_i \in P \wedge p_i \leq a\}$. Every point that is intersected by the original interval is still intersected by the new interval.

To see that one of the maximal covers has no intersections assume that all maximal covers have intersections. Now, take one of the maximal covers such that the two leftmost intervals intersecting are the rightmost among all maximal covers. Assume that these intervals are $[p_i - \ell, p_i]$ and $[p_j - \ell, p_j]$ and that $i \leq j$. Now, replacing $[p_i - \ell, p_i]$ by $[p_{q_j} - \ell, p_{q_j}]$ (or removing it altogether if $q_j = 0$) covers all points that are covered by the original cover, and has no intersections to the right of p_{q_j} in contradiction to the maximality of the chosen cover. Thus, there is a maximal cover with no intersections.

We have the following

Theorem 4 *For any m and n, the algorithm above finds a cover of the points by at most n intervals of length ℓ such that the sum of weights of the covered points is maximum. The time complexity of the algorithm is $O(nm)$.*

Proof: For $m = 1$ and any n the algorithm returns one interval ending at the point, which is optimal. The proof now follows by induction: consider now moving from $m-1$ to m points. One needs only consider adding an interval ending at p_m, as such a segment covers all the points (up to m) covered by any interval going further right. Now, if this interval intersects any other

interval in the optimal configuration, then, the other interval may be moved to end at point q_m, without exposing any point (as all points right of q_m are covered by the new interval). As no overlap exists between the last interval and previous ones, the optimal cover containing the interval $[p_m - \ell, p_m]$ is the cover containing this interval and the intervals of the optimal cover of $m-1$ segments up to point q_n. Thus, the optimal cover up to point m is the maximum between the above cover and the optimal cover not containing the interval $[p_m - \ell, p_m]$, which is the optimal cover of $m-1$ points with n intervals.

We now turn to prove the time complexity of the algorithm: Step 2 is done in time linear in m. Step 5 repeats mn times, where each iteration requires only a constant time to perform. Step 6 can be computed in linear time.

It is clear that this algorithm is optimal for calculating all sets $D(a,b)$ for $a = 1, \ldots, n$ and $b = 1, \ldots, m$. For given values of m and n we make the following conjecture.

Conjecture 1 For given values of m and n, the optimal algorithm for segment cover has time complexity of $O(mn)$.

4.2 Maximum covering of a measure

We now generalize the results of the previous section to any positive measure.

Consider now the following problem:

Problem 4 *Given a segment $[a,b]$, equipped with a positive measure $\mathrm{d}\mu$, a be real number, ℓ, and a natural number, n, find the optimal location of n intervals of length ℓ (assume $b - a > n\ell$) s_i, $1 \le i \le n$, such that $\int_{\cup_{i=1}^{n} s_i} \mathrm{d}\mu$ is maximum. Denote such an optimal solution by* maximum cover.

Define the cumulative weight $W(x) = \int_a^x \mathrm{d}\mu$. Assume $W(x)$ is continuous and differentiable. Let $W_r(x) = W(x) - W(x - r\ell)$. Let P_r be the set of all solutions of the equation $W_r'(x) = 0$ and the beginning and end of the interval $[a,b]$, i.e.,

$$P_r = \{x | W_r'(x) = 0\} \cup \{a + r\ell, b\}.$$

Let $d = |\cup_{r=1}^{n} P_r|$.

Algorithm 5 (Covering measure using intervals)

1. *For all* $1 \le r \le n$
   (a) *Compute* P_r.
   (b) *For each* $x_i \in P_r$ *do*
      i. $O_0(x_i) = 0$
      ii. $O_r(x_i) = \max_{0 \le r' \le r}\{W(x_i) + O_{r-r'}(x_{q_{i,r'}})\}$, *where* $q_{i,r'} = \arg\max\{x_j \in P_r | x_j < x_i - r'\ell\}$.
2. *Let* $r = |\cup_{r=1}^{n} P_r|$. *W.l.o.g assume that the* P_r *are disjoint.*
3. *Set* $O_{r,0} = 0$ *for every* $r \in \{0, \ldots, n\}$.
4. *Set* $O_{0,k} = 0$ *for every* $k \in \{0, \ldots, d\}$.
5. *For each* $j \in \{1, \ldots, n\}$, $k \in \{1, \ldots, d\}$
   (a) *Let* $r \in \{1, \ldots, n\}$ *be such that* $x_k \in P_r$.

(b) Let $t_k = \arg\max_{0 \le r' \le r}\{W_{r'}(x_k) + O_{r-r'}(x_{q_{k,r'}})\}$.
(c) If $W_{t_k}(x_k) + O_{j-t_k,q_{k,t_k}} > O_{j,k-1}$ *set* $A_{j,k} = 1$, $O_{j,k} = W(x_k) + O_{j-t_k,q_{k,t_k}}$.
(d) Else set $A_{j,k} = 0$, $O_{j,k} = O_{j,k-1}$

6. *Return the set* $A(n,d)$. *(for a pair* j,k, *if* $A_{j,k} = 0$ *then* $A(j,k) = A(j,k-1)$ *and otherwise* $A(j,k) = \{[x_k - t_k\ell, x_k]\} \cup A(j - t_k, q_{k,t_k})$*).*

We have the following

Theorem 5 *For any* n, ℓ, *the algorithm above finds a maximum cover with at most* n *segments of length* ℓ. *The time complexity of the algorithm is* $\sum_{r=1} nT_r + O(nd)$, *where* T_r *is the time complexity of computing* P_r, *and* $d = |\cup_{r=1}^{n} P_r|$.

Proof: The correctness of the algorithm follows from the correctness of Algorithm 5, and the fact that $\int_{\cup_{i=1}^{n} I_i} \mathrm{d}\mu$ is maximum when the total weight of elements in $\cup_{r=1}^{n} P_r$ is maximum.

As for the time complexity of the algorithm: the main issue is Step 1(b)ii, which can be computed in $O(|P_r|)$ for all r by going over P_r once and updating the maximum.

5 One dimensional generalized *k*-centers

In this section we introduce an $O(n^2 \log n)$ algorithm for the one dimensional k-centers problem.

Consider the following problem.

Problem 5 *Let* $P = \{p_1, \ldots p_n\}$ *be a set of points on the real line. We refer to these points as* clients. *Each client* p_i *has weight* w_i. *Let* $d_{i,j} = |p_i - p_j|$, *the distance between the points* p_i *and* p_j. *For a set* $F = \{f_1, \ldots, f_k\}$ *of facility points and a point* $p_i \in P$ *define* $d(p_i, F) = \min_{1 \le j \le k} |p_i - f_j|$. *For a given number* ℓ, *we say that a set of facility points* $F = \{f_1, \ldots, f_k\}$ *serves a point* p_i *if* $d(p_i, F) \le \ell/2$.

We consider two variants of target functions:

1. *Assuming that* ℓ *is given, find a set of facility points* $F = \{f_1, \ldots, f_k\}$ *such that*

$$\sum_{p_i \in P} w_i \cdot \delta(d(p_i, F) \le \ell/2)$$

is maximum, where $\delta(d(p_i, F) \le \ell/2) = 1$ *if* $d(p_i, F) \le \ell/2$, *and 0 otherwise.*
Note that the above is the dual problem to the segment covering problem of the previous section.
2. *Given a number* W, *find a number* ℓ *and a set of facilities* F_ℓ *such that*

$$\sum_{p_i \in P} w_i \cdot \delta(d(p_i, F_\ell) \le \ell/2) \ge W,$$

and ℓ *is minimum.*

Notice that this version is different then that of Brass et. al [5], since we discuss segments and not disks in the plan as in [5].

Solving the first target function, notice that this is exactly the problem of covering points using intervals. Therefore we can run Algorithm 4 and get a set of k intervals that maximize $\sum_{p_i \in P} w_i \cdot \delta(d(p_i, F) \le \ell/2)$. The desired set of facilities F is the set of centers of these intervals. The time complexity of computing F is the time complexity of Algorithm 4, namely $O(n \cdot k)$.

Now we propose the following algorithm for the second target function: (W.l.o.g assume that the sum of weights of any subset of points of size k is smaller than W - otherwise $\ell = \epsilon$ for any $\epsilon > 0$, since we can take F as a set of such k points.)

Algorithm 6 (One dimensional k-centers)

1. *Set D to be the ordered list of $d_{i,j} = |p_i - p_j|$, for $i > j$.*
   // The distances between clients
2. *Do a binary search on $\ell \in D$ for the minimal ℓ such that $\sum_{p_i \in P} w_i \cdot \delta(d(p_i, F_\ell) \leq \ell/2) \geq W$. For each considered ℓ run Algorithm 4 for checking if $\sum_{p_i \in P} w_i \cdot \delta(d(p_i, F_\ell) \leq \ell/2) \geq W$.*
3. *Return the locations of the centers of the intervals in $A(k, n)$[2] returned by Algorithm 4.*

Theorem 6 *The algorithm above solves the k-centers problem in $O(n^2 \log n)$ time.*

Proof: First notice that only distances in D should be considered, as if $\ell \notin D$ then all intervals do not have both endpoints at P and therefore may be shortened without exposing clients. In addition, recall that, given ℓ, Algorithm 4 returns k intervals that cover maximum weight of points. A point p_i is covered if its distance from some interval's center is at least $\ell/2$. This implies the correctness of the algorithm.

Computing the time complexity: building the distance list requires $O(n^2)$ time and sorting it requires $O(n^2 \log n)$ steps. The binary search requires $O(\log(n^2)) = O(\log n)$ steps, each of which solves the interval covering problem, having time complexity $O(n \cdot k)$.

Alternatively, if a constant bit accuracy of r bits is desired, one can make a binary search on the distances with $O(r)$ steps and obtain a solution with time complexity $O(nkr)$ (Or, $O(nr)$ if a covering of all points is sought using the algorithm in [24]).

As discussed in the introduction, Chen and Wang [11] presented an efficient algorithm for the weighted k-center problem on a real line. Their target function is the following: find a set of facilities F of size k such that $\max_{p_i \in P} w_i \cdot d(p_i, F)$ is minimum. This function is different from both our target functions so their time complexity cannot be compared to ours.

6 Conclusions

We presented here polynomial time algorithms for the maximum k-interval coverage problem, the maximum k-circular-arc coverage problem, the weighted interval covering problem, and the k-centers problem. In the general setting all these problems are NP-hard. However, we have shown that the one dimensional restriction of these problems is solvable in polynomial time using methods based on dynamic programming.

Acknowledgments

Reuven Cohen thanks the BSF for support. Science and Technology of Israel.

[2] Recall that $A(a, b)$ is a set of a intervals that covers the maximum possible weight of points out of the leftmost b points.

References

1. P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33:201–226, 2002.
2. A. A. Ageev and M. I. Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *Proc. IPCO*, pages 17 – 30, 1999.
3. N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. In *ACM Transactions on Algorithms (TALG)*, pages 153 – 177, 2006.
4. J. Bar-Ilan and D. Peleg. Approximation algorithms for selecting network centers. In *Proc. 2nd Workshop on Algorithms and Data Structures, Lecture Notes in Comput. Sci.*, pages 343–354, 1991.
5. P. Brass, C. Knauer, H.S. Na, C.S. Shin, and A. Vigneron. Computing k-centers on a line, arxiv:0902.3282v1, 2009.
6. H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC dimension. *Discrete Comput. Geom.*, 14:263–279, 1995.
7. P. Carmi, M.J. Katz, and N. Lev-Tov. Covering points by unit disks of fixed location. In *Proc. 18th International Symposium on Algorithms and Computation (ISAAC)*, pages 644–655, 2007.
8. D. Chakrabarty, E. Grant, and J. Köenemann. On column-restricted and priority covering integer programs. *Integer Programming and Combinatorial Optimization*, pages 355–368, 2010.
9. T. Chan. Geometric applications of a randomized optimization technique. *Discrete and Computational Geometry*, 22:547–567, 1999.
10. T.M. Chana and E. Grant. Exact algorithms and apx-hardness results for geometric packing and covering problems. *Computational Geometry, 2012*, 2012.
11. D. Z. Chen and H. Wang. Efficient algorithms for the weighted k-center problem on a real line. In *Proc. 22nd International Symposium on Algorithms and Computation (ISAAC)*, pages 584 – 593, 2011.
12. R. Cohen, M. Gonen, A. Levin, and S. Onn. On nonlinear multi-covering problems. *Journal of Combinatorial Optimization*, pages 1–15, 2015.
13. G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey. Worst-case and probabilistic analysis of algorithms for a location problem. *Operations Research*, 28:847–858, 1980.
14. D. de Werraa, C. Eisenbeisb, S. Lelaitc, and E. Stöhr. Circular-arc graph coloring: On chords and circuits in the meeting graph. *European Journal of Operational Research*, 136:483–500, 2002.
15. D. Eppstein. Fast construction of planar two-centers. In *Proc. of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 131–138, 1997.
16. T. Erlebach and E. J. van Leeuwen. PTAS for weighted set cover on unit squares. In *Proc. of the 13th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) and of the 14th Intl. Workshop on Randomization and Computation (RANDOM)*, pages 166–177, 2010.
17. G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Information Processing Letters*, 95:358–362, 2005.
18. Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
19. R. Fowler, M. Paterson, and S. Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12:133–137, 1981.
20. M. R. Garey and D.S. Johnson. *computers and intractability: A guide to the Theory of NP-Completeness*. Freeman, 1978.

21. T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Comput. Sci.*, 38:293–306, 1985.
22. T.F. Gonzalez. Covering a set of points in multidimensional space. *Information Processing Letters*, 40:181–188, 1991.
23. D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33:533–550, 1986.
24. Dorit S. Hochbaum and Asaf Levin. Optimizing over consecutive 1's and circular 1's constraints. *SIAM J. on Optimization*, 17(2):311–330, 2006.
25. D.S. Hochbaum and W. Maass. Fast approximation algorithms for a nonconvex covering problem. *J. Algorithms*, 8:305–323, 1987.
26. W. L. Hsu and G. L. Nemhauser. Easy and hard bottleneck location problems. *Disc. Appl. Math.*, 1:209–216, 1979.
27. R. Z. Hwang, R. C. T. Lee, and R. C. Chang. The slab dividing approach to solve the euclidean pcenter problem. *Algorithmica*, 9:1–22, 1993.
28. Lovász. On the ratio of optimal integral and fractional covers. *SIAM J. on Discrete Mathematics*, 13:383–390, 1975.
29. S. Masuyama, T. Ibaraki, and T. Hasegawa. The computational complexity of the m-centers problem on the plane. *Trans. IECE of Japan*, pages 57–64, 1981.
30. N. Megiddo. On the complexity of some geometric problems in unbounded dimension. *Journal of Symbolic Computation*, 10:327–334, 1990.
31. N. Megiddo and K. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13:182–196, 1984.
32. N.H. Mustafa and S Ray. Improved results on geometric hitting set problems. *Discrete Comput. Geom.*, 44:883–895, 2010.
33. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
34. J. Plesník. On the computational complexity of centers locating in a graph. *Aplikace Matematiky*, 25:445–452, 1980.
35. J. Plesnk. A heuristic for the p-center problem in graphs. *Disc. Appl. Math.*, 17:263–268, 1980.
36. R. Raz and S. Safra. A sub-constant error-probability PCP characterization of NP. In *Proc. 29th Symposium on the Theory of Computing (STOC)*, pages 475 – 484, 1997.
37. P. Slavik. Improved approximations of packing and covering problems. In *Proc. 27th Symposium on the Theory of Computing (STOC)*, pages 268 – 276, Baltimore, MD, USA, May 1995.
38. A. Srinivasan. Improved approximations guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999.
39. K.J. Supowit. Topics in computational geometry. Technical Report UIUCDCS-R-81-1062, Department of Computer Science, University of Illinois, Urbana, IL, 1981.