Algorithms for Disk Covering Problems with the Most Points

Bin Xiao
Department of Computing
Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
csbxiao@comp.polyu.edu.hk

Qingfeng Zhuge, Yi He, Zili Shao, Edwin H.-M. Sha [*]
Department of Computer Science
University of Texas at Dallas
Richardson, Texas 75083, USA
{qfzhuge, yxh011010, zxs015000, edsha}@utdallas.edu

ABSTRACT
Usually the covering problem requires all elements in a system to be covered. In some situations, it is very difficult to figure out a solution, or unable to cover all given elements because of resource constraints. In this paper, we study the issue of the partial covering problem. This problem is also referred to the robust k-center problem and can be applied to many fields. The partial covering problem becomes even more harder when we need to determine the subset of the group of all available elements to share resources. Several approximation algorithms are proposed to cover the most elements in this paper. For some real time systems, such as the battlefield communication system, the algorithm presented with polynomial-time complexity can be efficiently applied. The algorithm complexity analysis illustrates the improvement made by our algorithms, which are compared with other papers for the partial covering problem in the literature. The experimental results show that the performance of our algorithms is much better than other existing 3-approximation algorithm for the robust k-center problem.

KEY WORDS
Approximation algorithms, partial covering, k-center problem.

1 Introduction

In the battlefield, there are a lot of communication units and some command centers. The command centers are responsible for the successful information exchange among all mobile units. Because of the mobility for both mobile units and centers, we try to cover the most mobile units by a limited number of control centers with transmission range constraints. This kind of problem is analyzed as the clustering problem, which is clustering a set of points into a few groups. Clustering algorithms have been explored and deployed in many fields [1, 2], such as data compression, information retrieval, databases applications, image processing, facility location, clustering nodes in ad hoc networks etc. Given that the group number is k, the clustering problem is also referred to the k-center problem. We define the k-center problem as follows: *Let S be a set of n objects, generally represented as points in a d-dimensional metric space. Given an integer $k \leq n$, compute a k-clustering of S of the smallest possible size.* In other words, the k-center problem is formulated by covering S by k congruent disks of the smallest possible size. We always assume that k disks have the same radius r. In this paper, we only pay attention to the metric space by a plane (d = 2). However, all algorithms can be easily extended to arbitrary metrics.

Many heuristic algorithms [3, 4, 5] have been studied well for the k-center problem when $k \geq 3$. There are two major directions. One direction is for the number of disks is fixed to k and heuristic algorithms try to minimize the radius r of disks. The other focuses on the radius of each disk is fixed to r while heuristic algorithms explore minimum number of disks to cover all points. For the first approach, Gonzalez [6] gave a 2-approximation algorithm for the k-center problem in any metric space with time complexity $O(k \cdot n)$. In the same paper, Gonzalez proved that there is no polynomial-time algorithm for an approximation factor smaller than 2 unless P=NP. Feder and Greene improved the 2-approximation algorithm with complexity $O(n \log k)$ [7]. Some $(1+\epsilon)$ approximation algorithms [7, 8] are studied with non-polynomial running time. For the second approach, a polynomial-time approximation scheme can be achieved within approximation factors arbitrarily close to 1 [7]. Gonzalez proposed an 8-approximation algorithm for the fixed-size disks covering problem in [9]. Huang devised a 7-approximation algorithm for the 2-dimension metric space, and a 21-approximation algorithm for the 3-dimension metric space [10]. In [11], Franceschetti summarized the best known results in a table. However, to achieve smaller approximation factor ($\alpha < 7$), the polynomial running time will be very huge. For example, the algorithm in [11] for an approximation factor $\alpha = 6$ will require the running time to be $O(k \cdot n)$ with $k \approx 10^{16}$.

A topic has been studied during these years for the k-center problem is to cover part points [12, 13]. In some cases, there is no solution for covering n points in a plane by k disks with fixed radius. Examples are like building facilities to provide service within a fixed radius to a certain fraction of population, or allocating command centers in a battlefield to support communications among mobile units. The problem defined in [12] is called the robust k-center problem, which is to cover at least p points ($p \leq n$) by k disks with radius r. In this paper, we want to cover the

[*]This work is partially supported by TI University Program, NSF EIA-0103709 and Texas ARP 009741-0028-2001, USA.

most points (at least p points) for n points in a plane by k available disks. Those k disks are assumed with the same radius r. This kind of problem is NP-complete problem. The reasons is that to cover at least p points, when we set $p = n$, it will be the same to the k-center problem.

In order to cover as many points as possible (at least p from n) with k disks, in this paper we have made the contributions as follows:

- Come up with a new 2-approximation algorithm–RKC2 to the robust k-center problem. In [12], the authors only presented the best result with a 3-approximation algorithm. When p is close to n, this new 2-approximation algorithm becomes polynomial-time running. (Section 2)
- Propose a greedy algorithm to cover part points. (Section 3.1)
- Modify the RKC2 algorithm to the RKCP2 algorithm for the purpose of covering most points. (Section 3.2)
- Transfer the 3-approximation algorithm for the robust k-center problem in [12] to the RKCP3 algorithm, which is also applicable to the problem of covering most points. (Section 3.3)

The rest of this paper is organized as follows. In Section 2 we introduce the RKC2 algorithm for the robust k-center problem and prove it to be 2-approximation. Section 3 shows different heuristic algorithms to cover the most points and one example is illustrated. Experimental results are presented in Section 4. And conclusions are drawn in Section 5.

2 New RKC2 Algorithm

The robust k-center problem defined in this paper only considers points in a plane (the dimension is 2). Let n be the number of all points, p be a given positive integer such that $p \leq n$. If we have k same disks with radius r, the robust k-center problem can be defined as whether k disks can cover at least p points. This problem is an NP-complete problem. Suppose that n points in a plane area are the clients we want to serve, the center of k disks are the facility location to cover p points among n points, then the question studied here is the same as the facility location problem with outlier defined in [12].

Figure 1 illustrates how robust measures lead to better clustering solutions. In the example, to cover all points requires at least $k = 4$ disks. However, if we need to cover the most points with only 2 disks, Figure 1(b) shows the result to cover $p = 13$ points.

Following the same definition of algorithm approximation factor in [12], we show our RKC2 algorithm below and prove it to be 2-approximation to the optimal cost. For some cases, we can even know that it is impossible to cover p points by k disks. Let V be the set of all points in a plane

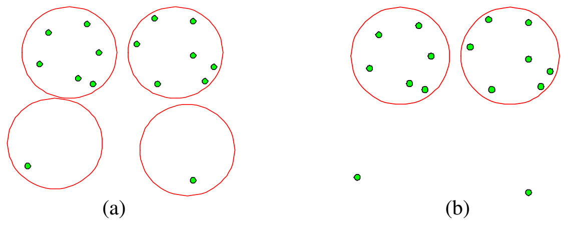


Figure 1. (a) To cover all points with k=4; (b) To cover p points with k=2, p=13.

and $|V| = n$. We suppose that the set of points satisfies the triangular inequality. p is a given positive number and $p \leq n$. Let C be a set in our algorithm that includes all points to be the centers of k disks. T_i is a temporary point set that includes p points from V. Each disk has the same radius r. We use *Flag* below to show whether there is a solution for k disks to cover any p points within the point set V. The $dist(a, b)$ is the Euclidean distance between the point a and b. The algorithm RKC2 is as follows:

```
• Flag = No
• For $i = 1, \dots, \binom{n}{p}$, do
    – Select $p$ different points from $V$, which generates a new point set $T_i$ with $T_i \neq T_j$ (j = 1, . . . , i-1)
    – Arbitrarily select one point from $T_i$, let this point be $c_1$, $C = \{c_1\}$
    – For $j = 2, \dots, k$, do
        ∗ For a point $t \in T_i$ and $t \notin C$, let
          $d_j(t) = min[dist(t, c_l), for \forall c_l \in C]$
        ∗ Let $d_j = max_{t \in T_i} d_j(t)$
        ∗ Let $c_j$ be the point $t$, which makes $d_j$ to have the maximum value
        ∗ $C = C \cup \{c_j\}$
    – If $d_k \leq 4r$
        ∗ If $k$ disks with centers in $C$ by radius $r$ can cover at least $p$ points in $V$
          Return Yes
        ∗ Else
          Flag = Yes
• If Flag = No
  It is impossible to cover $p$ points with $k$ disks
• Return No
```

We explain how this algorithm works. First, *Flag* is reset to be *No*. Whenever it is possible to cover p points with the given k disks for a particular T_i by the above algorithm, *Flag* is set to *Yes*. In the outer loop, p different points from V are selected in every iteration. Those p points yield a new point set T_i. Because there are $\binom{n}{p}$ times of different point sets, this outer loop will not end until $i = \binom{n}{p}$, which means we already check all possible solutions. For every selected point set T_i, it only includes p points. Subsequently, one node is arbitrarily chosen from T_i and this point becomes the center of one disk. We want to find the other $k-1$ center positions for disks. Such k points construct the center set C. The next center included to C is the point in T_i that has the maximum distance from all the points already in C. When there are k points in C, we already find k centers by the RKC2 algorithm for the p points arbitrarily chosen to T_i. d_k means the maximum distance from c_k (the kth center) to every other centers $(c_1, \ldots, c_{k-1})$. In other words, if k disks have the diameter by d_k, it is enough for them to cover all points in T_i. If $d > 4r$, it is impossible for k disks with radius r to cover all points in T_i, which will be proved by Lemma 2.1. Otherwise, *Flag* should be set to *Yes* to show the existence for all points in T_i covered only by k disks. Furthermore, if k disks with centers in C by radius r can cover p points in V, the RKC2 algorithm already reach a solution to the robust k-center problem and is ended by the return of *Yes*. The Theorem 2.1 will prove it.

After $\binom{n}{p}$ cases are tested and *Flag* is still *No*, it guarantees that there is no way to cover p points from V by k disks with radius r. Since we still can not find a method to cover p points, the algorithm will return *No*.

In the RKC2 algorithm, the process to generate a center set C with given p points is similar to the greedy algorithm in [6], which has been proved to be a 2-approximation algorithm. The idea is to execute the main loop another iteration, and let d_{k+1} be the resulting distance to the $k+1$ center to be added. Also the new center point set C' becomes $C \cup \{c_{k+1}\}$ that has $k+1$ points in it. By the definition of d_{k+1}, we can see that the distance between any two points in C' is at least d_{k+1}. Furthermore, any k-clustering must have two points of C' in one same cluster for $|C'| = k+1$. Thus, any k-clustering of T_i must have radius no less than $d_{k+1}/2$. However for the RKC2 algorithm, the radius of the k-clustering with centers at C is exactly d_{k+1}. Then we have the lemma below:

Lemma 2.1. *Given p points in T_i, the RKC2 algorithm provides a factor 2 approximation to the minimum k-clustering of T_i. In other words, the radius by disks with centers at points in C is no larger than 2 times the radius of any k clustering disks of T_i.*

Theorem 2.1. *Given a set V of n points from an arbitrary metric, an integer $k \leq n$, and an integer p, the RKC2 algorithm is a 2-approximation algorithm for the robust k-center problem.*

Proof. From Lemma 2.1, it is obvious that the RKC2 algorithm generates a 2-approximation solution when $d_k \leq 4r$ happened during the algorithm execution. This is because when $d_k \leq 4r$, the k disks with radius $2r$ can cover at least p points. For the case that every time $d_k > 4r$ during p points selection ($\binom{n}{p}$ times) from V, it is impossible for k disks to cover the entire p points with radius r. That means no solution exists. Thus the RKC2 algorithm can not find a center set C with radius $2r$ to cover p points. □

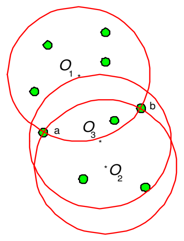


Figure 2. To cover 2 points with a disk by the greedy algorithm.

Given p points in T_i, to generate k centers requires $O(p \cdot k)$ time. A better way to reduce the time complexity to $O(p \cdot \log k)$ is shown in [7]. Since the RKC2 algorithm will execute $\binom{n}{p}$ times for the different point set T_i, the time complexity should be $O(\binom{n}{p} \cdot p \log k)$. When we want to cover the most points, p should be the same as n and the time complexity for the robust k-center problem becomes $O(n \cdot \log k)$. It is a polynomial-time algorithm. Furthermore, when p is close to n, the RKC2 algorithm can still remain polynomial-time running.

3 Different Algorithms to Cover the Most Points

Given n points in a plane, sometimes it is impossible for all points to be covered within k disks by radius r. Even if such cases exist, to find a solution until now will require the computation time to be exponential, since k-center problem is NP-complete. In the real-time system, such as covering all communication unites in a battlefield, this is too costly. To cover at least p points within k disks is also NP-complete. The reason is that when we set $p = n$, the robust k-center problem becomes the standard k-center problem. In this section, we show some polynomial-time algorithms to cover points as many as possible.

3.1 Greedy Algorithm

For n points in a plane to be covered by disks with radius r, there are at most $2\binom{n}{2}$ different cases for disk covering. This is because for any two point a, b, there are at most two center positions, from which the distance to both points (a and b) are exact r. In Figure 2, there are 2 different disk covering for the point a and b, which can be disk o_1 and o_2. Because disk o_3 covers the same points as o_2, it is not necessary for us to pay attention to it any more. In other words, we always can move a disk to uniquely represent two points with those points on its circle edge. For the point c and d in Figure 2, there is only one case for disk covering since the distance between c and d is $2r$ (the diameter). One disk covering is said to be unique only if there is at least one point different from any other disk covering. Let D be the set with different disks covering and G be the disk set returned by the greedy algorithm that only has k units. Below is the greedy algorithm to cover the most points.

- $G = \varnothing$
- For any two points, we generate two new disk units to cover them if possible. Suppose there are m such unique disk covering with $m \leq 2 \cdot \binom{n}{2}$. These disks come up with a disk set $D = \{D_1, D_2, \ldots, D_m\}$.
- For $i = 1, \ldots, k$, do
  - Select the disk that covers the maximum number of points from D. Let this disk be D_i, $D = D - \{D_i\}$, $G = G \cup \{D_i\}$.
  - Remove the points covered by D_i from all disk in D.
- Return G.

There are at most $2 \cdot \binom{n}{2}$ different disk covering given n points. Constructing such disks will consume time $O(2 \cdot \binom{n}{2})$. In one iteration, the time to select the disk that covers the maximum number of points needs time $O(2 \cdot \binom{n}{2})$. After including D_i to be in disk set G, the greedy algorithm will refresh the number of points covered by all other remaining disks in D. Thus it requires an extra time $O(2 \cdot \binom{n}{2})$. In the greedy algorithm, we have k iterations. The computation complexity for the greedy algorithm should be $O(2 \cdot \binom{n}{2}) + k \cdot O(2 \cdot \binom{n}{2} + 2 \cdot \binom{n}{2}) = O(2 \cdot \binom{n}{2}) + O(4k\binom{n}{2}) = O(k \cdot n^2)$.

3.2 RKCP2 Algorithm

The algorithm for the robust k-center problem doesn't request to cover all points with given k disks. With a little change to the RKC2 algorithm to solve the robust k-center problem in Section 2, we can apply it to the case of covering the most points. Here we assume $p = n$, which means the algorithm tries to cover all available points. All points are in the point set V and C is a point set including k units, which are the positions for the center of k disks. Below is the RKCP2 algorithm procedure to cover the most points from the 2-approximation algorithm RKC2 for the robust k-center problem.

- Arbitrarily select one point from V, let this point be $c_1, C = \{c_1\}$
- For $i = 2, \ldots, k$, do
  - For a point $t \in V$ and $t \notin C$, let $d_i(t) = min[dist(t, c_l), for \forall c_l \in C]$
  - Let $d_i = max_{t \in V} d_i(t)$
  - Let c_i be the point t, which makes d_i to have the maximum value
  - $C = C \cup \{c_i\}$
- Return C

The time complexity for the above RKCP2 algorithm can be $O(n \cdot \log k)$ [7].

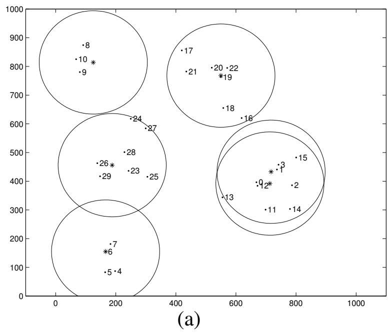
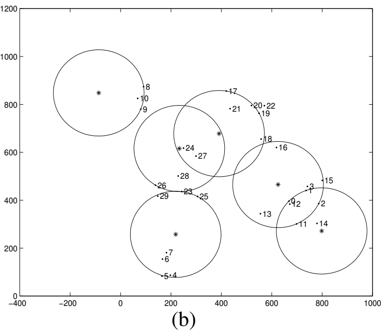



Figure 3. (a) 30 points can be covered by 6 disks with radius 180m; (b) The result by the greedy algorithm.

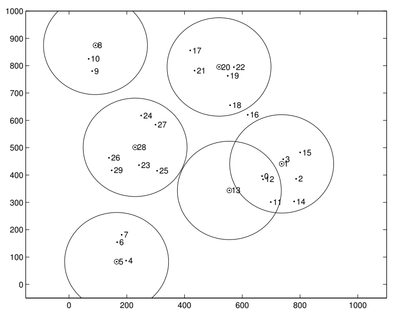
3.3 RKCP3 Algorithm

In [12], the authors illustrate a 3-approximation algorithm for the robust k-center problem, which can also be adopted to solve the problem of covering the maximum number of points. That 3-approximation algorithm has the input with a radius r for k disks and a set of n points V from an arbitrary metric space. It will figure out a solution S such that $cost(S) \leq 3r$. For the n points V, it assumes that there exists an optimal solution O such that $cost(O) = r$.

For each point $v_i \in V$, G_i (E_i, resp.) is denoted as the set of points that are within distance r ($3r$, resp.) from v_i. The set G_i is referred as disks of radius r and the set E_i as the corresponding expanded disks of radius $3r$. The RKCP3 algorithm procedure to cover the most points can be described as follows with a little modification to the original 3-approximation algorithm in [12].

- Construct all disks and corresponding expanded disks.
- For $i = 1, \ldots, k$, do
  - Let G_i be the heaviest disk, i.e. contains the most uncovered points.
  - Mark as covered all points in the corresponding expanded disk E_i.
  - Update all the disks and expanded disks, i.e., remove from them all covered points.
- Return $\{G_1, G_2, \ldots, G_k\}$.

The algorithm above has been proved to be a 3-approximation algorithm for the robust k-center problem


(a)

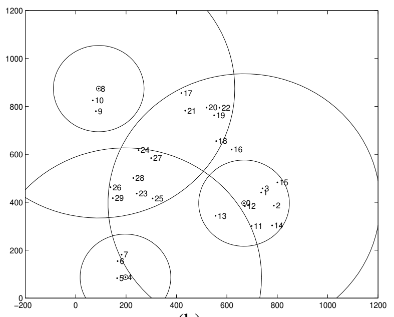

(b)

Figure 4. (a) The result by the RKCP2 2-approximation algorithm; (b) The result by the RKCP3 3-approximation algorithm.

in [12]. The time complexity for the algorithm should be $O(k \cdot n)$.

3.4 Example for the Above Algorithms

To cover the most points in a plane, we show an example here applied with different algorithms mentioned above. Given 30 points randomly scattered in a square by 1,000 meters * 1,000 meters, an optimal solution for all points covered by 6 disks is shown in Figure 3(a). The results with the greedy algorithm are illustrated in Figure 3(b). With the RKCP2 and RKCP3 algorithms to cover the most points, we have the results shown in Figure 4(a) and Figure 4(b) respectively. In the greedy algorithms, the center position of one disk is determined by 2 points on its edge. For the RKCP3 algorithm to cover the most points, we show a bigger circle with radius 720 meters for every covering disk in Figure 4(b). The number of points covered is counted by those in the small disks (with radius 180 meters). All points that are between one small disk and its corresponding big circle are removed from the system by the 3-approximation algorithm. Following this property, point 7 is not covered by the algorithm because it is first removed from the system after the decision made for the first disk.

4 Simulation Results

We evaluate the performance of different algorithms through covering different number of points scattered in an area by disks. The performance of each algorithm is measured by the number of points covered by given disks and how many disks are used for the system. In the simulation, we compare the results for different algorithms.

The position of all points are randomly distributed in an area by 1,000 * 1,000 meters. The number of points is varied from 20 to 279 during the simulation. All disks have the same radius by 180 meters. During the simulation, the center position of a disk can be anywhere inside the square area. Different number of disks is assigned according to how many points in all. The relationship between the number of points and disks follows the equation 1 below. D represents how many disks available in the system while P denotes the number of points inside the simulated square area.

$$D = \lceil \frac{P}{20} \rceil + 3 \qquad (1)$$

In Table 1, the performance of the greedy, 2-approximation (RKCP2) and 3-approximation (RKCP3) algorithms are listed with the model defined above. Inside the square, 20, 50, 100, 200 and 270 points are randomly scattered for one simulation event respectively. Under Column "# p", the data shows the number of points covered by a special algorithm and the following Column "%" represents the percentage of the covering part based on all available points. During the simulation, the number of available disks is changed according to the number of all points in the system, which is defined in Equation 1. To cover the points as many as possible, different algorithms will consume different number of disks. The data in the Column "# d" is the result of used disks by the execution of those three algorithms.

For the system with 270 points, there are 16 disks available. However, the greedy algorithm can cover all points only with 15 disks for some cases. The RKCP3 algorithm from the 3-approximation algorithm in [12] is not efficient because it only uses part of available disks. We can see from Table 1 that when the system has 270 points, 4 disks with radius 720 meters (180 meters * 3) is probably enough to cover them all. On the contrary, the 2-approximation algorithm always tries to cover points by all disks. Under a few circumstance, such as less number of points in the area and fewer disks available, the RKCP3 algorithm yields a better performance than the RKCP2 algorithm, which can be seen from Table 1 for the system with 50 points. Most of the time, the greedy algorithm performs the best while the 3-approximation algorithms generate the worst results.

We show the simulation results for different algorithms in Figure 5 when the number of points increases from 20 to 279. The curves in Figure 5 represent the number of covered points by different algorithms. That straight line represents the total points in the simulation environment. When an algorithm has a better performance to cover more points, the simulated curve will be closer to the straight line. Thus, the greedy algorithm always yields a good performance, especially for a large number

Algo.	20 points			50 points			100 points			200 points			270 points		
	# p	%	# d	# p	%	# d	# p	%	# d	# p	%	# d	# p	%	# d
Greedy	16	80	4	39	78	5	93	93	8	199	99.5	13	270	100	15
2-Appr	7	35	4	12	24	5	51	51	8	174	87	13	257	95.2	16
3-Appr	6	30	2	18	36	4	33	33	4	63	31.5	3	91	33.7	4

Table 1. Covered points and used disks.

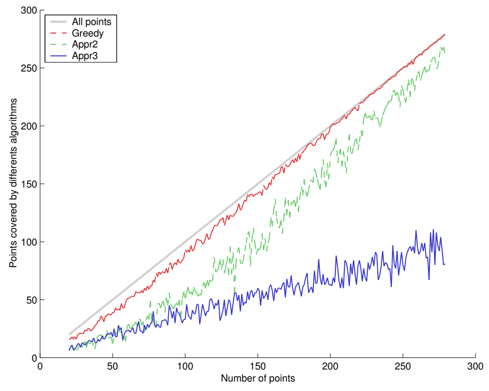


Figure 5. The number of covered points.

of points scattered in the square area. The number of covered points goes up for all three algorithms when there are more points available. But the increasing rate is slow for the 3-approximation algorithm. When points are scarcely scattered in the simulation area, the 3-approximation algorithm sometimes performs better than the 2-approximation algorithm. However in the long run, the 2-approximation algorithm generates much better results.

5 Conclusion

In daily life or some special circumstances (such as the battlefield), it is very hard or impossible to cover all users with limited servers. Thus to cover most users will be a satisfying result. This problem belongs to the robust k-center problem. In this paper we presented a new 2-approximation algorithm (RKC2) for the robust k-center problem, which is polynomial-time running when p is close to n. Based on it, a real polynomial-time algorithm (RKCP2) is proposed to cover the most points. Among those three proposed heuristic polynomial-time algorithms (greedy, RKCP2, RKCP3) for the robust k-center problem to cover the most points, the greedy algorithm yields a good performance indicated by the simulation. The outcome of the 3-approximation (RKCP3) algorithm originally from [12] is poor because the covered points is below 50% for most cases. When more disks are available, the polynomial 2-approximation algorithm (RKCP2) yields close results as the greedy algorithm while the running time is $O(n \cdot \log k)$.

References

[1] D. Z. (Editor), "Facility location: A survey of applications and methods," *Springer Series in Operations Research, Springer Verlag, New York*, 1995.

[2] D. B. Shmoys, É. Tardos, and K. Aardal, "Approximation algorithms for facility location problems (extended abstract)," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pp. 265–274, 1997.

[3] P. K. Agarwal and C. M. Procopiuc, "Approximation algorithms for projective clustering," in *Proceedings of 11th ACM-SIAM Sympos. Discrete Algorithms*, pp. 538–547, 2000.

[4] D. Johnson, "Approximation algorithms for combinatorial problems," *Journal of Computing and Systems Sciences*, vol. 9, pp. 256–278, 1974.

[5] H. Bronninamm and M. Goodrich, "Almost optimal set covers in finite vcdimension," *Discrete Computational Geometry*, vol. 14, pp. 463–479, 1995.

[6] T. Gonzalez., "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[7] T. Feder and D. Greene, "Optimal algorithms for approximate clustering," in *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing (STOC)*, pp. 434–444, 1988.

[8] P. K. Agarwal and C. M. Procopiuc, "Exact and approximation algorithms for clustering (extended abstract)," in *Proceedings of 9th ACM-SIAM Sympos. Discrete Algorithms*, pp. 658–667, 1998.

[9] T. Gonzalez., "Covering a set of points in multidimensional space," *Information Processing Letters*, vol. 40, pp. 181–188, 1991.

[10] H. Huang, A. W. Richa, and M. Segal, "Approximation algorithms for the mobile piercing set problem with applications to clustering in ad-hoc networks," *ACM Journal on Mobile Networks (MONET)*, pp. 52–61, 2002.

[11] M. Franceschetti, M. Cook, and J. Bruck, "A geometric theorem for approximate disk covering algorithms," 2001.

[12] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan, "Algorithms for facility location problems with outliers," in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 642–651, 2001.

[13] R. Gandhi, S. Khuller, and A. Srinivasan, "Approximation algorithms for partial covering problems," in *Proceedings of the Twenty-Eighth International Colloquium on Automata, Languages, and Programming (ICALP), LNCS 2076*, pp. 225–236, Jul. 2001.