Approximation Algorithms Design for Disk Partial Covering Problem

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

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

Abstract

Mobile servers are established to provide services for mobile nodes in an anticipated area. If the distribution of mobile nodes can be foreseen, the location of mobile servers becomes critical to the QoS of wireless systems. Under resource and topology constraints, it is very difficult to figure out a solution, or unable to cover all given mobile nodes within limited number of mobile servers. In this paper, we study the issue of the partial covering problem such that part of mobile nodes to be covered. Several approximation algorithms are proposed to cover the maximum number of elements. For real time systems, such as the battlefield communication system, the proposed algorithms with polynomial-time complexity can be efficiently applied. The algorithm complexity analysis illustrates the improvement made by our algorithms. 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.

1 Introduction

In a battlefield, command servers are built to provide services for communication mobiles. Command servers are responsible for the successful information exchange among all mobile units. Because of the mobility for both mobile units and centers, the covering problem addressed is to cover the most mobile units by a limited number of control centers regarding to some constraints (such as landscape obstacles and short transmission range). A mobile unit covered by a command center can contact others while an isolated one loses its communication. This kind of problem is analyzed as the clustering problem, which is clustering a set of points into a few groups (servers). Given that the group (server) number is k, the clustering problem is also referred to the k-center problem. The k-center problem can be described as follows: *Suppose that S is a set of n objects, generally representing n points in a d-dimensional metric space. Given an integer $k \leq n$, compute a k-clustering of S of the smallest possible size; or given that each cluster with the same size r, calculate the smallest number of k for the k-clustering of S.* In other words, the k-center problem is formulated by covering S with k congruent disks either by the smallest possible size of each disk, or to minimize the number of k for disks with fixed size r. 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).

The k-center problem becomes NP complete when $k \geq 3$. Many heuristic algorithms [1, 2, 3, 4, 5] have been studied well to yield polynomial time running methods. There are two major research directions. One direction assumes that the number of disks is fixed to k and heuristic algorithms aims to minimize the radius r of each disk. 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 investigated for non-polynomial time complexity. 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 explored during these years for the k-center problem is to cover partial points [12, 13]. Given k disks with the same radius r, the partial covering issue investigates the center locations of those k disks to cover the most points among total n points. 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 named by the robust k-center (RKC) problem, which is to cover at least p points ($p \le n$) by k disks with radius r. In this paper, we want to cover the most points (at least p points) from 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: First, a new 2-approximation algorithm–RKC2 to the robust k-center problem is proposed. 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. Second, we present a greedy algorithm to cover part points, and prove it to be a 2-approximation algorithm, such that it can cover at least half points (n/2). The illustrated RKC2 algorithm in Section 2 is adjusted to the RKCP2 algorithm in order to cover most points thereafter. Furthermore, the 3-approximation algorithm for the robust k-center problem in [12] is transfered to the RKCP3 algorithm, which is also applicable to the problem of covering most points.

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 Algorithm for the Robust K-Center Problem

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 $p \le n$. If we have k disks with the same 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

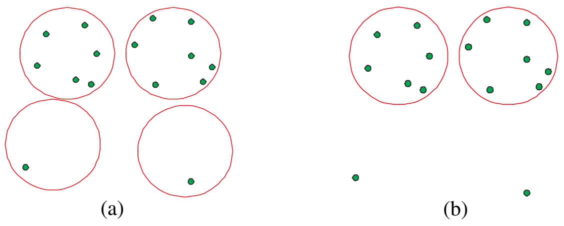

Figure 1. (a) To cover all points with k=4; (b) To cover p points with k=2, p=13.

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 available, Figure 1(b) shows the result to cover $p = 13$ points. When we set $p = n$, the robust k-center problem is equivalent to the standard k-center problem.

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 further know that it is impossible to cover p points by k given disks. Let V be the set of all points in a plane and $|V| = n$. The set of points satisfies the triangular inequality. p is a given positive number and $p \le n$. 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 it is possible for k disks to cover at least p points within the point set V. The $dist(a, b)$ is the Euclidean distance between point a and b. The algorithm RKC2 is as follows:

```
• Flag = No
• For $i = 1, \ldots, \binom{n}{p}$, do
    – Select $p$ different points from $V$, which generates
      a new point set $T_i$ with $T_i \ne T_m$ (m = 1, . . . , i-1)
    – Arbitrarily select one point from $T_i$, let this point
      be $c_1$, $C = \{c_1\}$
    – For $j = 2, \ldots, k$, 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 $d_k \le 4r$
```

```
        ∗ 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
  No solution exists for covering $p$ points with $k$ disks
  Return No
• Else
  It is possible to cover $p$ points with $k$ disks
```

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 of i, 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 cases in the RKC2 algorithm. For every created point set T_i, it contains p points. Subsequently, one node is arbitrarily chosen from T_i and this point becomes the center of one disk. The left task is to decide the other $k-1$ center positions for disks. Such k points construct the center set C. After the decision of the first center by point c_1, the next center classified into C is the point in T_i (not in C) that has the maximum distance from itself to all points in C. When there are k points in C, we already find k centers for the given k disks. The value of 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 $d_k > 4r$, it is impossible for k disks with radius r to cover all p points in T_i, which will be proved by Lemma 2.1. Otherwise, *Flag* should be set to *Yes* to show the possibility that all points in T_i can be covered by the same k disks with radius r. Furthermore, if k disks with centers in C by radius r can cover p points in V, the RKC2 algorithm already yields 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* remains *No*, it guarantees that there is no way to cover p points from V by the same k disks with radius r. Since no solution is available after checking all cases, the algorithm will return *No*.

In the RKC2 algorithm, the process to generate a center set C within p points is similar to the greedy algorithm in [6], which has been proved to be a 2-approximation algorithm. Let d_{k+1} be the resulting distance to the $k+1$ center to be added when executing the main loop with another iteration. 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}, the distance between any two points in C' is at least d_{k+1}. Furthermore, any k-clustering solution must place two points in C' into a cluster for $|C'| = k+1$. Thus, the radius of disks for any kind of k-clustering method in terms of points in T_i must be no less than $d_{k+1}/2$. However for the RKC2 algorithm, the radius of the k-clustering disks with centers at points in C is exactly d_{k+1}. Hence, 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 any 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. □

Given p points in T_i, to generate k centers requires $O(p \cdot k)$ time by the above RKC2 algorithm. 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 set 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

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 positions for disk covering according to the following greedy algorithm. This is because for any two points a and b, there are at most two center positions, from which the distance to both points (a and b) is 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, disk O_3 is said to be duplicated with disk O_2. In other words, a disk can be

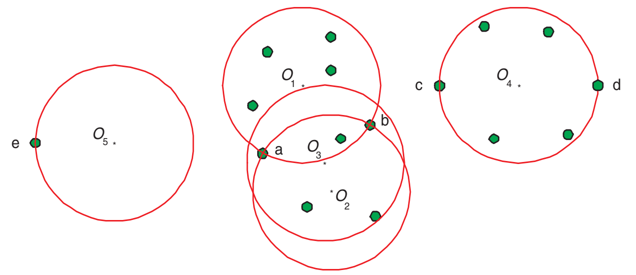

Figure 2. To cover 2 points with a disk by the greedy algorithm.

moved 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 center position for disk covering since the distance between c and d is $2r$ (the diameter). The point e is far away from any other point such that no disk can cover another point beside point e. Thus, the center position for the disk O_5 is arbitrarily selected during the below greedy algorithm. One disk covering is said to be unique if there is at least one point different from any other disk covering inside its covering area. Let D be the disk set such that each contained disk is unique for its disk covering, and G be the disk set returned by the greedy algorithm that has k units. Below is the greedy algorithm to cover the most points.

- $G = \varnothing$
- For any two points to be covered by a disk, two new disk units (D_i, D_j) are built to cover them with those two points on the circle edge. Suppose the number of unique disk covering is m and $m \leq 2 \cdot \binom{n}{2}$. These disks make up with a disk set $D = \{D_1, D_2, \dots, D_m\}$.
- For $i = 1, \dots, k$, 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.

Given n points there are at most $2 \cdot \binom{n}{2}$ different disk covering cases. Constructing all disk covering needs time $O(2 \cdot \binom{n}{2})$. In one iteration, the time to select the disk that covers the maximum number of points is $O(2 \cdot \binom{n}{2})$. After including D_i to be in the 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 by $O(2 \cdot \binom{n}{2})$. In the greedy algorithm, k iterations are executed to generate the final disk set G. Hence, the computation complexity for the greedy algorithm should be $O(2 \cdot \binom{n}{2}) + k \cdot O(2 \cdot \binom{n}{2} + 2 \cdot \binom{n}{2}) = O(2 \cdot \binom{n}{2}) + O(4k\binom{n}{2}) = O(k \cdot n^2)$.

THEOREM 3.1. *Given n points (in the point set V) in a plane and k disks with the same radius r, and suppose that k disks can cover all points, the greedy algorithm is a 2-approximation algorithm such that the number of points covered is at least $n/2$.*

Proof. The similar reasoning process as the proof for the greedy algorithm to the set-cover problem can be applied here. Suppose that m disks are needed to cover all n points by the greedy algorithm. Let D_i be the ith selected disk. Thus we have $|D_1| \geq |D_2| \geq \cdots \geq |D_m|$ and $m \geq k$. Let V' be the subset of V for points that are not covered by the first k-iterations of the greedy algorithms. We assume k disks can cover all points by an optimal solution and let the optimal disk covering be $O_1, O_2, \dots, O_k$. Let the points covered by the optimal solution for point set V' be $O'_1, O'_2, \dots, O'_k$ respectively. Thus, for the number of points in V', we have

$$|V'| = \sum_{i=k+1}^{\infty} |D_i| = \sum_{i=1}^{k} |O'_i| \tag{1}$$

By the definition of the greedy algorithm, we have:

$$|O'_i| \leq |D_{k+1}| \qquad for \qquad i = 1, \dots, k \tag{2}$$

$$|D_{k+1}| \leq |D_i| \qquad for \qquad i = 1, \dots, k \tag{3}$$

From 1, 2 and 3,

$$\sum_{i=1}^{k} |O'_i| \leq k \cdot |D_{k+1}| \leq \sum_{i=1}^{k} |D_i| \tag{4}$$

Thus

$$|V'| \leq \sum_{i=1}^{k} |D_i|$$

Because $|V'| + \sum_{i=1}^{k} |D_i| = n$, we have $\sum_{i=1}^{k} |D_i| \geq n/2$. $\square$

3.2 RKCP2 Algorithm

The algorithm design for the robust k-center problem doesn't request to cover all points for k available disks. The RKC2 algorithm to solve the robust k-center problem in Section 2 can be applied to the problem of covering the most points with little change. Let $p = n$ and that means the RKC2 algorithm need to cover all points in a plane. 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].

3.3 RKCP3 Algorithm

In [12], the authors illustrate a 3-approximation algorithm for the robust k-center problem, which can be applied to solve the problem of covering the maximum number of points in a plane. The proposed 3-approximation algorithm aims at covering p points from a point set V (n points in total) by the same k disks of radius r. Given the condition that an optimal solution O can cover all n points by k disks with radius r ($Cost(O) = r$), that 3-approximation algorithm can generate a solution S with $cost(S) \leq 3r$ for a total covering.

For each point $v_i \in V$, G_i (E_i, respectively) 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 from 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 in [12]. The time complexity for the algorithm is $O(k \cdot n)$.

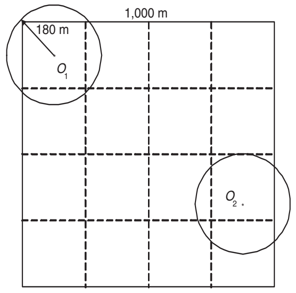

Figure 3. The simulation model.

4 Simulation Results

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

The position of all points are randomly distributed in an area by 1,000 * 1,000 meters, which is shown in Figure 3. The total number of points is increased from 20 to 279 during the simulation process. Suppose that all disks are fixed with the same radius by 180 meters. The center of a disk can be anywhere inside the square area. The number of disks depends on how many points generated in the simulation system. The relationship between the number of points and disks follows the function: $D = \lceil \frac{P}{20} \rceil + 3$, where D represents how many disks available in the system, P is the number of points inside the area defined in Figure 3. According to the relationship of the number between simulated node and disk, we can see that an extra disk is available for every 20 more points in the system. In Figure 3, the disk O_1 is large enough to cover a small square area by 250 * 250 meters since the diagonal distance (354 meters) of the small square area is smaller than the diameter of a disk ($2r = 360$ meters). The whole square is made up with 16 such areas. Hence, 16 disks can cover any number of points in that square theoretically.

The results for different algorithms to cover the most points are illustrated in this section. In Table 1, the performance of the greedy, the 2-approximation (RKCP2) algorithm and the 3-approximation (RKCP3) algorithm are listed with the simulation 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 one 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. To cover the points as many as possible, different algorithms will consume different number of disks. The data

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 consumed disks by different algorithms.

in Column "# d" is the result of used disks by the execution of these 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 for the purpose of covering because only a part of available disks involves in the computation. We can see from Table 1 that when 270 points scatter in the simulated area, 4 disks with radius 720 meters (180 meters * 3) is probably enough to cover them all. In contrast, the 2-approximation algorithm always exhausts all disks to cover more points. Under some special circumstances, such as less number of points in the simulated 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. In most cases, the greedy algorithm performs the best while the 3-approximation algorithm generates the worst results.

5 Conclusion

In this paper, the disk partial covering problem is discussed. This issue is also referred to the robust k-center problem. The other best known heuristic algorithm for the robust k-center problem is the 3-approximation algorithm in [12] in the literature. In this paper a new 2-approximation algorithm (RKC2) for the robust k-center problem is presented, which is polynomial-time running when p is close to n. Based on it, a practical polynomial-time algorithm (RKCP2) is illustrated to cover the most points. Furthermore, we proposed a greedy algorithm that is 2-approximation based on the number of covered points. In other words, the greedy algorithm covers at least half the points compared to an optimal solution. The experimental results demonstrate this property. Among those three heuristic polynomial-time algorithms (greedy, RKCP2, RKCP3) for the robust k-center problem, the greedy algorithm yields a good performance indicated by the simulation. The outcome of the 3-approximation (RKCP3) algorithm in [12] is poor because the percentage of 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] 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.

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

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

[4] Z. Drezner, "The p-center problem: heuristic and optimal algorithms," *J. Oper. Res. Soc.*, vol. 35, pp. 741–748, 1984.

[5] R. Z. Hwang, R. C. T. Lee, and R. C. Chang, "The slab dividing approach to solve the euclidean p-center problem," *Algorithmica*, vol. 9, pp. 1–22, 1993.

[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," in *http://www.paradise.caltech.edu/ETR.html*, 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.