Distributed Unit Clustering

Kian Mirjalali*  Seyed Ali Tabatabaee*  Hamid Zarrabi-Zadeh*

Abstract

Given a set of points in the plane, the unit clustering problem asks for finding a minimum-size set of unit disks that cover the whole input set. We study the unit clustering problem in a distributed setting, where input data is partitioned among several machines. We present a \((3 + \varepsilon)\)-approximation algorithm for the problem in the Euclidean plane, and a \((4 + \varepsilon)\)-approximation algorithm for the problem under general \(L_p\) metric \((p \geq 1)\). We also study the capacitated version of the problem, where each cluster has a limited capacity for covering the points. We present a distributed algorithm for the capacitated version of the problem that achieves an approximation factor of \(4 + \varepsilon\) in the \(L_2\) plane, and a factor of \(5 + \varepsilon\) in general \(L_p\) metric. We also provide some complementary lower bounds.

1 Introduction

The exponential growth of data in real-world applications and the incapability of individual computers to store and process the whole data have motivated the research in the area of distributed algorithms. In this paper, we study the distributed version of the following unit clustering problem. Given a set of \(n\) points in the plane, partition the points into clusters, each enclosable by a unit disk, so as to minimize the number of clusters used. An instance of the problem is illustrated in Figure 1. The problem has applications in various areas including image processing [14, 19] and wireless sensor networks [18, 20].

The unit clustering problem is known to be NP-hard in the Euclidean plane [11]. The first polynomial-time approximation scheme (PTAS) for the problem was given by Hochbaum and Maass [14]. The runtime of the PTAS was later improved by Feder and Greene to \(n^{O(1/\varepsilon^d)}\) in any fixed \(d\) dimensions [10]. A PTAS for the capacitated version of the problem is recently given in [12]. Online variants of the problem are also studied in the literature [6, 9].

For massive datasets, where no single machine can store the whole data, distributed models such as MapReduce have been introduced and extensively used over the past decade [2, 4, 8, 13, 16]. In the distributed unit clustering problem, the input set \(S\) is partitioned among a set of machines, where each machine \(i\) has a subset \(S_i\) of the input, and the goal is to compute collaboratively a unit clustering of the whole set \(S = \bigcup_i S_i\). The notion of composable coresets introduced in [15] has been proved to be useful in designing distributed algorithms that take \(O(1)\) rounds of MapReduce. In this framework, each machine performs a computation on its portion of data, and sends a small subset of its data (called a coreset) to a central machine. The central machine then composes the coresets and finds an approximate solution based on the information carried by the coresets. This framework has been successfully used to derive approximation algorithms for several optimization problems [1, 3, 7, 17].

In this paper, inspired by the idea of composable coresets, we design distributed algorithms for the capacitated and uncapacitated versions of the unit clustering problem. For the uncapacitated version, we provide a \((3 + \varepsilon)\)-approximation algorithm in the Euclidean plane, and a \((4 + \varepsilon)\)-approximation algorithm in the plane under general \(L_p\) metric, for any real number \(p \geq 1\). For the capacitated version, we provide a \((4 + \varepsilon)\)-approximation algorithm in the \(L_2\) plane, and a \((5 + \varepsilon)\)-approximation algorithm under general \(L_p\) metric. We also prove some lower bounds on the approximation factor and communication complexity of any distributed algorithm for the problem under the composable coreset framework. In particular, we show that the unit clustering problem in the Euclidean plane admits no composable coreset with approximation factor better than 2. Moreover, we show that the communication complexity of our algorithms is optimal under this framework.

*Department of Computer Engineering, Sharif University of Technology. Email: {mirjalali,satabatabae}@ce.sharif.edu, zarrabiz@sharif.edu.
2 Preliminaries

Given a real number \( p \geq 1 \), and two points \( a = (x_a, y_a) \) and \( b = (x_b, y_b) \) in the plane, the distance of \( a \) and \( b \) under \( L_p \) metric is defined as
\[
d_p(a, b) = \sqrt[p]{|x_a - x_b|^p + |y_a - y_b|^p},
\]
and \( d_{\infty}(a, b) = \max(|x_a - x_b|, |y_a - y_b|) \). We refer to the plane \( \mathbb{R}^2 \) in which \( L_p \) metric is the distance measure as the \( L_p \) plane. Whenever we state a proposition for all \( L_p \) metrics, \( p \geq 1 \), we implicitly assume that \( L_{\infty} \) is also included.

For \( p \geq 1 \) and \( r \geq 0 \), an \( L_p \) disk of radius \( r \) is defined as the set of points \( \{ a \in \mathbb{R}^2 \mid d_p(a, c) \leq r \} \), where \( c \in \mathbb{R}^2 \) is the center of the disk. An \( L_p \) disk of radius 1 is called a unit \( L_p \) disk. Whenever the underlying metric \( L_p \) is clear from the context, we simply use the terms disk and unit disk.

Given a set of points in the plane under an \( L_p \) metric, the unit clustering problem is to cover the points by congruent disks of radius \( r \), so as to minimize the number of disks used. We refer to this problem as UC\(_r\). Moreover, we denote by UC\(_r\)(\( S \)) an optimal solution to the UC\(_r\) problem on an input set \( S \). Whenever \( r = 1 \), we drop \( r \) from the notation, and simply write UC and UC\(_1\)(\( S \)), instead.

3 Covering Disks With Smaller Ones

In this section, we present some upper bounds on the number of disks of radius \( r < 1 \) needed to cover a unit disk. We will use the following well-known fact as an ingredient: for any \( 1 \leq p \leq q \), a unit \( L_p \) disk can be covered by a unit \( L_q \) disk.

**Lemma 1** Under any \( L_p \) metric, \( p \geq 1 \), a unit disk can be covered by \( [2/r]^2 \) disks of radius \( r \), for \( 0 < r \leq 1 \).

**Proof.** Let \( D \) be a unit \( L_p \) disk, and \( S \) be a unit \( L_\infty \) disk covering \( D \). As \( S \) is a square of side length 2, it can be covered by \( [2/r]^2 \) squares of side length \( r \). On the other hand, each square of side length \( r \) can be covered by an \( L_p \) disk of radius \( r \), which completes the proof.

According to Lemma 1, a unit disk in any \( L_p \) plane can be covered by a constant number of smaller disks, whenever the radius of the smaller disks is fixed. The next two lemmas provide tighter bounds on this constant.

**Lemma 2** Under any \( L_p \) metric, \( p \geq 1 \), a unit disk can be covered by four disks of radius \( \sqrt{2}/2 \).

**Proof.** We prove the lemma in two cases:

**Case 1:** \( 1 \leq p < 2 \). Let \( D \) be a unit \( L_p \) disk, and \( S \) be a unit \( L_2 \) disk covering \( D \). As illustrated in Figure 2, \( S \) can be covered by four diamonds (\( L_1 \) disks) of diameter \( \sqrt{2} \). On the other hand, each of these four diamonds can be covered by an \( L_p \) disk of radius \( \sqrt{2}/2 \). Hence, four \( L_p \) disks of radius \( \sqrt{2}/2 \) can cover a unit \( L_p \) disk in this case.

**Case 2:** \( p \geq 2 \). Let \( D \) be a unit \( L_p \) disk, and \( S \) be a square of side length 2 enclosing \( D \). As illustrated in Figure 3, \( S \) can be covered by four \( L_2 \) disks of radius \( \sqrt{2}/2 \). On the other hand, each of these four \( L_2 \) disks can be covered by an \( L_p \) disk of the same radius. Therefore, four \( L_p \) disks of radius \( \sqrt{2}/2 \) can cover a unit \( L_p \) disk in this case, which completes the proof.

It is worth noting that a unit \( L_1 \) disk cannot be covered by less than four smaller \( L_1 \) disks. Moreover, in \( L_2 \) metric, four disks of radius \( r < \sqrt{2}/2 \) cannot cover a unit disk. Hence, in general \( L_p \) metric, both our bounds of 4 and \( \sqrt{2}/2 \) are essentially tight. Nevertheless, for the special case of \( L_2 \) metric, it is possible to cover a unit disk by a fewer number of smaller disks.

**Lemma 3** In the \( L_2 \) plane, a unit disk can be covered by three disks of radius \( \sqrt{3}/2 \).

**Proof.** The proof is illustrated in Figure 4.
its input data $S$ and sends a subset $T_i$ as a coreset to the central machine. In the second phase, the central machine combines the coresets obtained form local machines into a single set $T$, and computes a disk cover $C$ of $T$, which after a proper adjustment can cover the whole input set.

**Algorithm 1 Distributed Unit Clustering**

1: Let $r = \sqrt{3}/2$ and $\delta = (1 - r)/2$.
2: on each machine $i$ ($1 \leq i \leq m$) in parallel do
3: Find an $O(1)$-approximation $C_i$ to UC$_r(S_i)$.
4: For each disk $D \in C_i$, pick an arbitrary point in $S_i \cap D$, and add it to a set $T_i$.
5: Send $T_i$ to the central machine.
6: on the central machine do
7: Let $T = \bigcup_{i=1}^m T_i$.
8: Find a $(1 + \varepsilon)$-approximation $C$ to UC$_r(T)$.
9: Increase the radii of disks in $C$ from $r$ to $1$.
10: return $C$.

**Theorem 4** Algorithm 1 is a $(3 + \varepsilon)$-approximation algorithm for the unit clustering problem in the $L_2$ plane, and a $(4 + \varepsilon)$-approximation algorithm for the problem under general $L_p$ metric, $p \geq 1$. The runtime of the algorithm is $O(n \log n) + (mk)^{O(1/\varepsilon)}$, and its communication complexity is $O(mk)$, where $n$ is the total number of points, $m$ is the number of machines, and $k$ is the size of an optimal solution.

**Proof.** Let $S = \bigcup_{i=1}^m S_i$ be the input set in the plane, under a given $L_p$ metric, $p \geq 1$. We first prove that the output of the algorithm, $C$, is a feasible solution, i.e., each point in $S$ is covered by a disk in $C$. Fix a point $q \in S_i \subseteq S$. By our algorithm, $q$ is covered by a disk of radius $\delta$ in $C_i$. As we add one point from each disk in $C_i$ to $T_i$, there is point $t \in T_i$, which is within distance $2\delta$ to $q$. On the other hand, each point of $T_i$ is covered by a disk of radius $r$ in $C$. Let $D$ be the disk in $C$ covering $t$. Therefore, the distance of $t$ to the center of $D$ is at most $r + 2\delta = r + (1 - r) = 1$. Therefore, $q$ is covered by $D$ after its radius is increased to one. Hence, $C$ is a feasible solution.

Now, we prove the approximation factor of the algorithm. Let $C^*$ be an optimal solution to UC($S$), and $C'$ be an optimal solution to UC$_r(T)$. By Lemma 2, each disk in $C^*$ can be covered by four disks of radius $r = \sqrt{3}/2 > \sqrt{2}/2$. Therefore, there is a set of $4|C^*|$ disks of radius $r$ covering $S$. Since $T \subseteq S$, we have $|C'| \leq 4|C^*|$. Moreover, the set $C$ computed by the algorithm is a $(1 + \varepsilon)$-approximation to $C'$, and therefore we have $|C| \leq (1 + \varepsilon)|C'| \leq \frac{4 + 4\varepsilon}{4}\leq |C^*|$. By re-adjusting $\varepsilon$ properly (e.g., by running the algorithm with $\varepsilon' = \varepsilon/4$), we get an approximation factor of $4 + \varepsilon$ for the problem, for any $\varepsilon > 0$. In the special case of $L_2$ metric, Lemma 3 states that each disk in $C^*$ can be covered by three disks of radius $r = \sqrt{3}/2$, and hence, the approximation factor of the algorithm is $3 + \varepsilon$ in this case.

The communication complexity of the algorithm corresponds to the size of $T = \bigcup_{i=1}^m T_i$. For $1 \leq i \leq m$, let $C_i^*$ and $C_i'$ be optimal solutions to UC($S_i$) and UC$_r(S_i)$, respectively. Since $S_i \subseteq S$, we have $|C_i'| \leq |C^*|$. Moreover, by Lemma 1, each unit disk in $C_i'$ can be covered by a constant number of disks of radius $\delta$, and hence, $|C_i'| \leq c \cdot |C_i^*|$, for some constant $c \geq 1$. On the other hand, each $C_i$ is an $\alpha$-factor approximation to $C_i'$, for some constant $\alpha \geq 1$, and thus, $|C_i| \leq \alpha |C_i'| \leq \alpha c |C_i^*| \leq \alpha c |C^*|$. As $|T_i| = |C_i|$, we have $|T| = \bigcup_{i=1}^m |T_i| \leq m \cdot \alpha c |C^*|$. Since $|C^*| = k$, the communication complexity of the algorithm is $O(mk)$.

For the runtime, we note that a $(1 + \varepsilon)$-approximation to UC can be computed in $n^{O(1/\varepsilon)}$ time [10], and a constant-factor approximation to UC can be obtained in $O(n \log n) \log |S|)$ time [5]. The runtime of our algorithm on the $i$-th machine is therefore $O(|S_i| \log |S_i|)$, which sums to $O(|S| \log |S|) = O(n \log n)$ on all local machines, and amounts to $T^{O(1/\varepsilon)} = (nk)^{O(1/\varepsilon)}$ on the central machine.

## 5 Capacitated Unit Clustering

In this section, we consider the capacitated version of the unit clustering problem, where each disk has a fixed capacity $L$. We present a distributed approximation algorithm for this version of the problem under any $L_p$ metric, $p \geq 1$. The algorithm is presented in Algorithm 2. The first phase of the algorithm is similar to that of Algorithm 1, except that here, each point $t \in T_i$ is assigned a weight $w(t)$ which specifies the number of points $t$ is representative for. These weights are then used in the second phase to properly limit the number of points assigned to each computed unit disk.
Algorithm 2 Capacitated Unit Clustering

1: Let $r = \sqrt{3}/2$ and $\delta = (1 - r)/2$.
2: on each machine $i$ ($1 \leq i \leq m$) in parallel do
3: Find an O(1)-approximation $C_i$ to UC$_d(S_i)$.
4: Assign each point of $S_i$ to one of its covering disks in $C_i$, with ties broken arbitrarily.
5: For each disk $D \in C_i$, pick an arbitrary point $t \in S_i \cap D$, set its weight $w(t)$ to the number of points assigned to $D$, and add $t$ to $T_i$.
6: Send $T_i$ to the central machine.
7: on the central machine do
8: Let $T = \bigcup_{i=1}^{m} T_i$.
9: Find a $(1 + \varepsilon)$-approximation $C_0$ to UC$_r(T)$.
10: Assign each point of $T$ to one of its covering disks in $C_0$, with ties broken arbitrarily.
11: For each disk $D \in C_0$, add $\lceil w(D)/L \rceil$ copies of $D$ to a set $C$, where $w(D)$ is the total weight of points assigned to $D$.
12: Distribute point weights among their covering disks in $C$, so that each disk receives weight $\leq L$. (A point weight may be split among two disks.)
13: Increase the radii of disks in $C$ from $r$ to $1$.
14: return $C$.

Theorem 5 Algorithm 2 is a $(4 + \varepsilon)$-approximation algorithm for the capacitated unit clustering problem in the $L_2$ plane, and a $(5 + \varepsilon)$-approximation algorithm for the problem under general $L_p$ metric, $p \geq 1$. The runtime of the algorithm is $O(n \log n) + (mk)^{O(1/\varepsilon)}$, and its communication complexity is $O(mk)$, where $n$ is the total number of points, $m$ is the number of machines, and $k$ is the size of an optimal solution.

Proof. Let $S = \bigcup_{i=1}^{n} S_i$ be the input set in the plane, under a given $L_p$ metric, $p \geq 1$. First, notice that the output of the algorithm, $C$, is a feasible solution. This is because each point in $S$ is within distance $r + 2\delta = 1$ to the center of one of the disks in $C$, by an argument similar to what we used in Algorithm 1. Moreover, by our distribution of the weights among disks, no disk in $C$ receives more than $L$ points. Therefore, $C$ is a feasible solution. The runtime and communication complexity of the algorithm are also implied by the same arguments used in the proof of Algorithm 1.

It only remains to prove the approximation factor of the algorithm. Let $C^*$ be an optimal solution to the capacitated unit clustering problem on the set $S$, and let $C'$ be an optimal solution to (uncapacitated) UC($S$). Note that $|C'| \leq |C^*|$. Moreover, $|C^*| \geq n/L$, because all $n$ points in $S$ are covered by $|C^*|$ disks of capacity $L$.

According to the algorithm,

$$|C| = \sum_{D \in C_0} \lceil w(D)/L \rceil$$

$$\leq \sum_{D \in C_0} \lceil 1 + w(D)/L \rceil$$

$$= |C_0| + n/L$$

$$\leq |C_0| + |C^*|.$$  

Moreover, according to the proof of Theorem 4, $C_0$ is a $(4 + \varepsilon)$-approximation to $C'$ under general $L_p$ metric, and a $(3 + \varepsilon)$-approximation to $C'$ under $L_2$ metric. Therefore, $|C| \leq (5 + \varepsilon)|C^*|$ in general $L_p$ metric, and $|C| \leq (4 + \varepsilon)|C^*|$ in the $L_2$ plane, which completes the proof.  

6 Lower Bounds

In this section, we provide lower bounds on the approximation factor of any distributed algorithm for the unit clustering problem in the $L_2$ plane under the composable coreset framework. We also prove a lower bound on the communication complexity of the distributed algorithms for the problem under this framework.

A coreset algorithm receives as input a sequence $S$ of points, and returns as output a subset of $S$, called a coreset. We call a coreset algorithm rotation-invariant if for a fixed sequence $S$ of points, it always returns the same coreset, even if the input is rotated in the plane.

Theorem 6 The unit clustering problem in the $L_2$ plane admits no composable coreset with approximation factor better than 2. If the underlying coreset algorithm is rotation-invariant, the problem admits no $\alpha$-composable coreset, for any $\alpha < 3$.

Proof. Let $A$ be the coreset algorithm used by local machines. Let $S$ be a sequence of points evenly placed on a circle of radius $1/2$. We can pick $S$ sufficiently large so that $|A(S)| < |S|$. Then, by the pigeonhole principle, there exist two distinct subsequences $T_1$ and $T_2$ of $S$ such that $A(T_1) = A(T_2)$. Assume w.l.o.g. that a point $v \in S$ is in $T_1$ but not in $T_2$. Since $A(T_1) = A(T_2)$, we have $v \notin A(T_1)$. Let $C_1$ and $C_2$ be two concentric circles of radius $1$ and $1 + \varepsilon$, respectively, for some $\varepsilon > 0$, such that $v$ is on the boundary of $C_2$, while other points lie inside $C_1$ (see Figure 5). Let $u$ be the point on the boundary of $C_1$ furthest away from $v$.

Consider an instance with two partitions $S_1$ and $S_2$ (on two separate machines), where $S_1 = \{u\}$ and $S_2$ is either $T_1$ or $T_2$. If $S_2 = T_1$, at least two unit disks are needed to cover all the points as $d(u, v) > 2$. On the other hand, if $S_2 = T_2$, the whole input can be covered by a single unit disk, $C_1$. When $A(S_2)$ is sent to the central machine, it cannot distinguish whether the original set has been $T_1$ or $T_2$. Therefore, any solution
The algorithms provided in this paper both have $O(mk)$ communication complexity. The following theorem shows that the communication complexity of our algorithms is indeed optimal.

**Theorem 7** Any distributed algorithm for the unit clustering problem under the composable coreset framework requires $\Omega(mk)$ communication, where $m$ is the number of machines, and $k$ is the size of an optimal solution.

**Proof.** Let $S_i$ be the set of points in the $i$-th machine ($1 \leq i \leq m$), and let $k_i$ be the size of an optimal unit clustering for $S_i$. Suppose that all $S_i$’s are far from each other, so that no disk covering a point in $S_i$ can cover a point in $S_j$, for $j \neq i$. If the coreset sent by the $i$-th machine contains less than $k_i$ points, the central machine receives no enough information to cover all the points in $S_i$, and hence, the final solution will not be feasible. Therefore, the number of points sent by the $i$-th machine must be at least $k_i$.

Now, consider the case where all machines have the same set of points, and hence, $k_i = k$ for all $1 \leq i \leq m$. By the argument provided above, each machine, independently from the others, must send at least $k$ points to the central machine, and hence, the central machine receives at least $mk$ points in this case. \qed

## 7 Conclusions

In this paper, we studied the unit clustering problem in a distributed settings, and presented approximation algorithms for both capacitated and uncapacitated versions of the problem in general $L_2$ metric, $p \geq 1$. Our algorithms can be implemented in $O(1)$ rounds of MapReduce. Moreover, the composable coresets provided in this paper naturally lead to algorithms in the one-pass streaming model. In higher dimensions, our algorithms can be extended in a natural way to obtain constant factor approximations in any fixed $d$ dimensions. It is interesting to see if the approximation factors of our algorithms can be improved, in particular, in the capacitated version.

## References


