Fully Dynamic k-center Clustering

Mauro Sozio, Telecom Paristech

Google NY, September 4th, 2018

Real-World Datasets

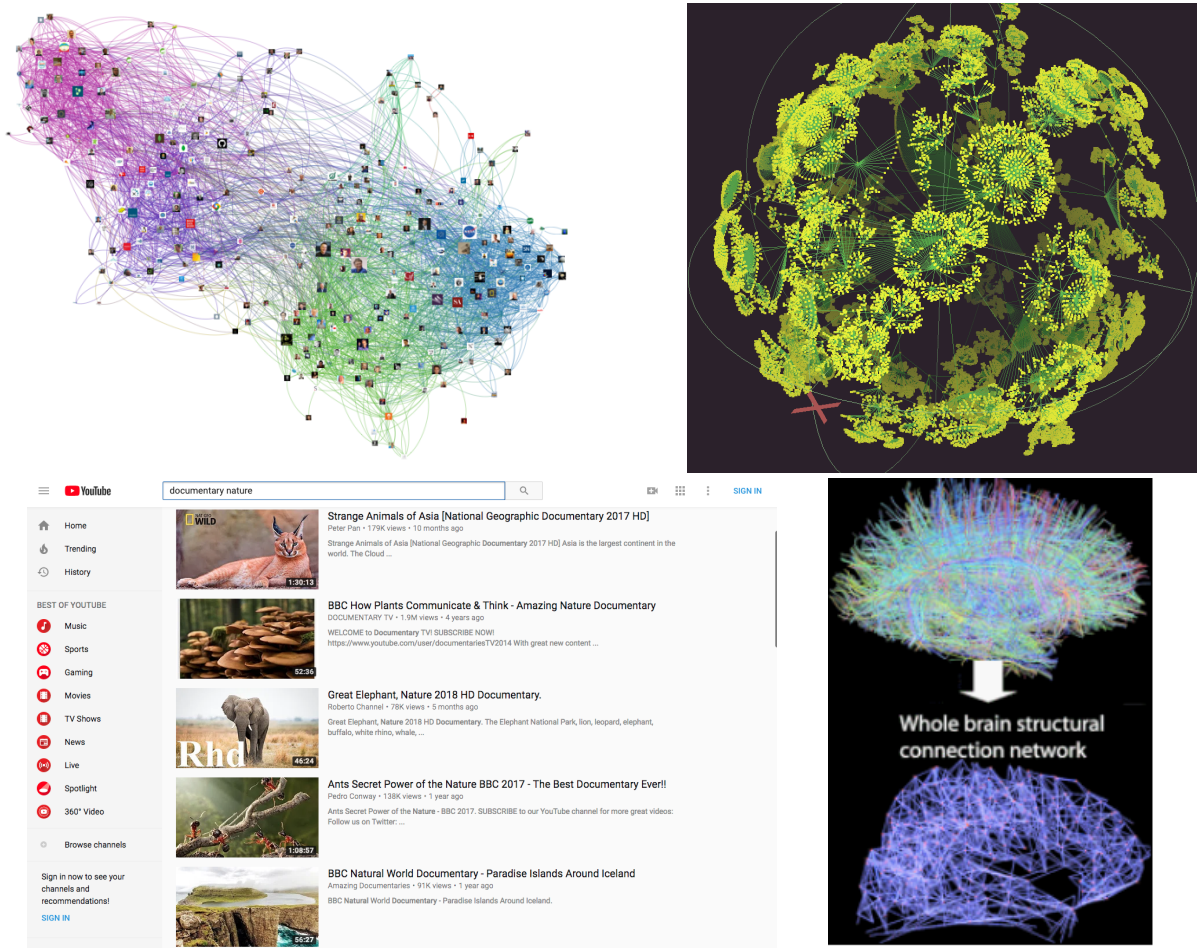


Figure: Twitter, the Internet, Youtube, and the human brain.

Real-World Datasets: large and Dynamic

Facts about modern real-world datasets:

- Google processes over 40000 queries every second, while more than 400 hours worth of Youtube videos are uploaded every minute;
- two billion Facebook users with links being added or removed;
- more than 6000 tweets per sec. are posted on Twitter;
- the human brain contains 86 billion neurons.

Real-World Datasets: large and Dynamic

Facts about modern real-world datasets:

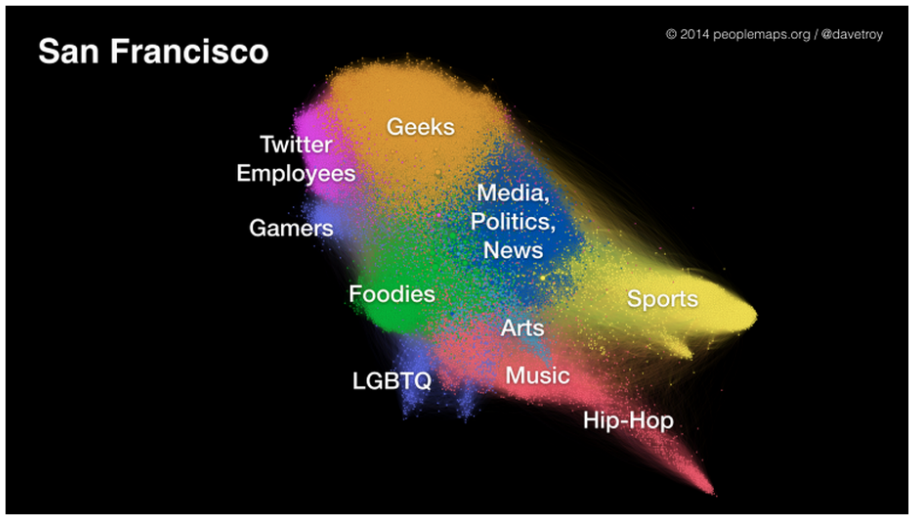
- Google processes over 40000 queries every second, while more than 400 hours worth of Youtube videos are uploaded every minute;
- two billion Facebook users with links being added or removed;
- more than 6000 tweets per sec. are posted on Twitter;
- the human brain contains 86 billion neurons.

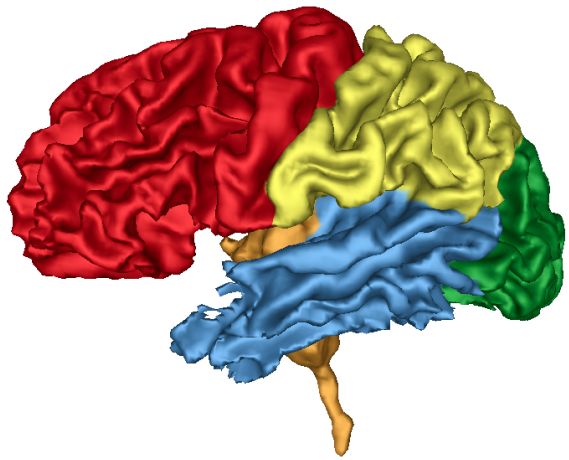
Urgent need for algorithms to make sense of large dynamic datasets!

Roadmap

Clustering

One of the most widely used techniques for exploratory data analysis, with applications ranging from biology to social sciences or psychology.





Dynamic Clustering

Goal: Maintain a "good" clustering of highly-dynamic data: social networks, biological networks, transportation networks, ...

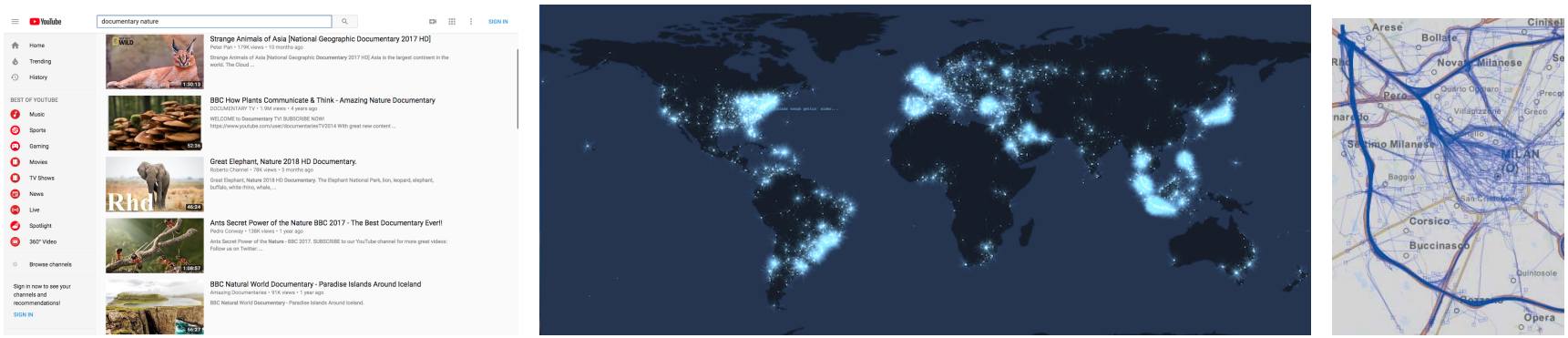


Figure: Youtube (left) Geotagged tweets (middle), car trajectories in Milan (right).

Roadmap

k-Center Clustering: Definition

Definition 1 (k-Center Clustering)

Given a set X and $k \in \mathbb{N}$, find a partition $C_1, \ldots, C_k$ (clusters) of X, and k points $c_1, \ldots, c_k$ (*centers* of the clusters) so as to minimize the maximum cluster radius (max distance between points and their centers).

Distance function: Euclidean distance or any other distance function.

2-approx Greedy for k-center clustering

Let X be the set of input points, $C = \emptyset$ be the set of centers.

Pick one point $x \in X$ arbitrarily; $C \leftarrow C \cup \{x\}$; $X \leftarrow X \setminus \{x\}$;
While $|C| < k$ and $X \neq \emptyset$

1. Pick one point $x \in X$ with maximum $d(x, C) := \min_{c \in C} d(x, c)$
2. $C \leftarrow C \cup \{x\}$
3. $X \leftarrow X \setminus \{x\}$

For each $c \in C$, create a cluster with center c and all points closest to c.

2-approx Greedy for k-center clustering

Let X be the set of input points, $C = \emptyset$ be the set of centers.

Pick one point $x \in X$ arbitrarily; $C \leftarrow C \cup \{x\}$; $X \leftarrow X \setminus \{x\}$;
While $|C| < k$ and $X \neq \emptyset$

1. Pick one point $x \in X$ with maximum $d(x, C) := \min_{c \in C} d(x, c)$
2. $C \leftarrow C \cup \{x\}$
3. $X \leftarrow X \setminus \{x\}$

For each $c \in C$, create a cluster with center c and all points closest to c.

2-approximation guarantee: let $y \in X \setminus C$ be a point with max $d(y, C)$. Let $d_{\min}$ be the min pairwise distance among the points in $C \cup \{y\}$. $d_{\min} = d(y, C)$ ($\max_{x \in X} d(x, C)$ is non-increasing). Every point is at distance at most $d_{\min}$ from its center in C (otherwise we'd have picked another center). There is no clustering with max radius $< \frac{d_{\min}}{2}$ (at least two points in $C \cup \{y\}$ must be in a same cluster).

Greedy in Practice

Well-separated Clusters: We say that $C_1, \ldots, C_k$ are well separated if for any x, y, z, w with x, y belonging to a same cluster and z, w belonging to two different clusters, it holds: $d(x, y) < d(z, w)$.

Greedy in Practice

Well-separated Clusters: We say that $C_1, \ldots, C_k$ are well separated if for any x, y, z, w with x, y belonging to a same cluster and z, w belonging to two different clusters, it holds: $d(x, y) < d(z, w)$.

Equivalent to: two points are in a same cluster if and only if their distance is smaller than D, where D is the max diameter of a cluster.

Greedy in Practice

Well-separated Clusters: We say that $C_1, \ldots, C_k$ are well separated if for any x, y, z, w with x, y belonging to a same cluster and z, w belonging to two different clusters, it holds: $d(x, y) < d(z, w)$.

Equivalent to: two points are in a same cluster if and only if their distance is smaller than D, where D is the max diameter of a cluster.

Suppose there are k well-separated clusters. Which algorithm can find them? **Yes:** Greedy, Single Linkage. **No:** k-means, k-means++.

However, Greedy is sensitive to outliers.

Related Work

Dynamic k-center: V. Cohen-Addad et al. (ICALP '16) focused on the sliding window model.

Many real-world applications require arbitrary update operations:

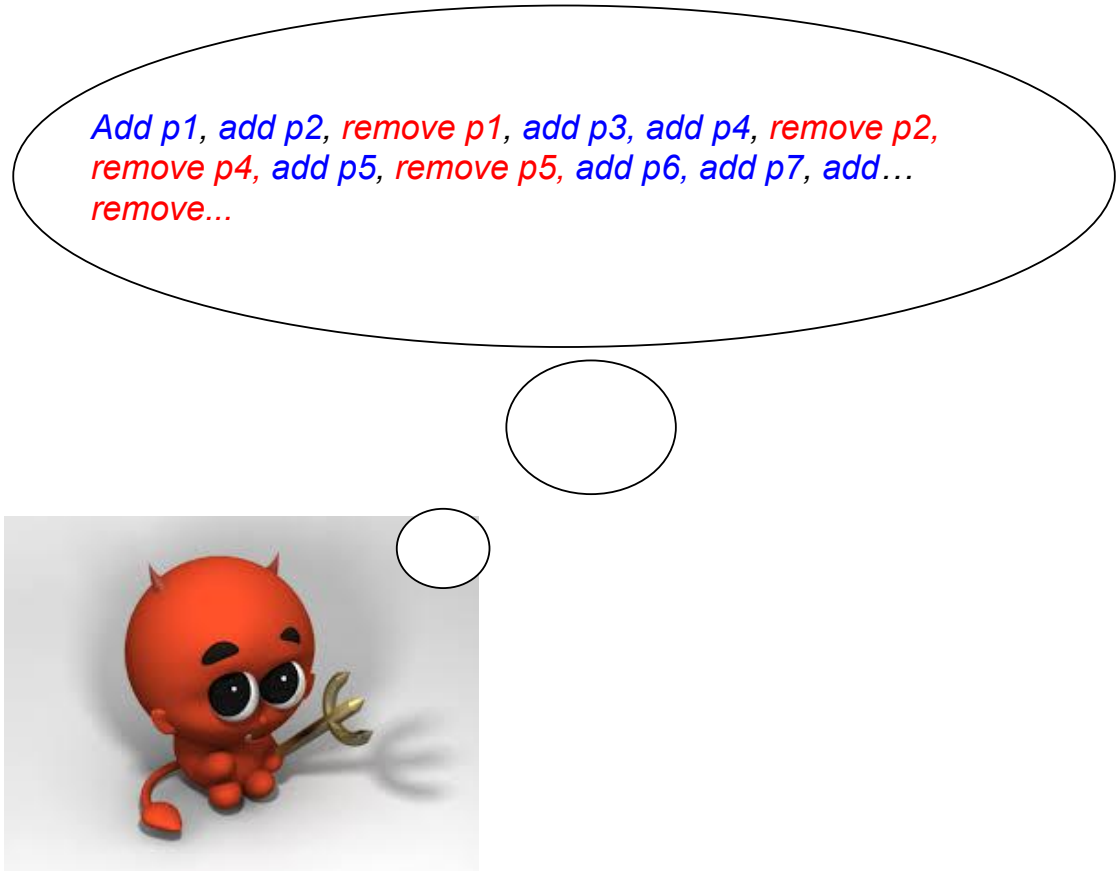
- remove points flagged as inappropriate, due to privacy concerns, ...
- clustering trajectories;
- right-to-be-forgotten principle.

Very few fully dynamic clustering algorithms.

Roadmap

Fully Dynamic Model

An adversary fixes a set O of insert/remove operations before our algo.



Our algo does not know O, adversary cannot predict its random choices.

Naive Algorithm

Naive: Recluster all the points whenever a point is inserted or deleted.

Cost per operation: $\Theta(k \cdot N)$, N max input size.

Too expensive!

Toward a Dynamic Algorithm: Bucketize!

"Guess" the optimum radius r. For any r, either show that there is no clustering with max radius r or compute a clustering with max radius $2 \cdot r$.

Consider radius values in $\Gamma := \{d_{\min}, d_{\min}(1+\epsilon), d_{\min}(1+\epsilon)^2, \ldots, d_{\max}\}$, $\epsilon > 0, i \in \mathbb{N}$, $d_{\min}, d_{\max}$ min and max distance. $|\Gamma| = O(\frac{\log \delta}{\epsilon})$, $\delta := \frac{d_{\max}}{d_{\min}}$.

The clustering with radius $2r$, $r \in \Gamma$ smallest, gives a $2+\epsilon$-approximation.

Static Algorithm, r given

Repeat until X becomes empty or k clusters are formed:

1. Pick one point x from X, arbitrarily and make it a *center*.
2. Create a cluster C with x and all points at distance $\leq 2 \cdot r$ from x.
3. Remove C from X.

Static Algorithm, r given

Repeat until X becomes empty or k clusters are formed:

1. Pick one point x from X, arbitrarily and make it a *center*.
2. Create a cluster C with x and all points at distance $\leq 2 \cdot r$ from x.
3. Remove C from X.

If all points are clustered, we have a clustering with max radius is $2 \cdot r$.

Certificate: Otherwise, there are $k+1$ points at distance $> 2 \cdot r \Rightarrow$ a k-clustering with max radius r cannot be formed.

Dynamic Algorithm: r given, Random Deletions

Points can be inserted arbitrarily but are deleted randomly:

Insertions:

- insert x to any cluster whose center is at distance $\leq 2 \cdot r$ from x.
- if not possible and $< k$ clusters, form a new cluster with center x.
- if the previous two fail $\Rightarrow$ **certificate**: $k+1$ points at distance $> 2 \cdot r$.

Dynamic Algorithm: r given, Random Deletions

Points can be inserted arbitrarily but are deleted randomly:

Insertions:

- insert x to any cluster whose center is at distance $\leq 2 \cdot r$ from x.
- if not possible and $< k$ clusters, form a new cluster with center x.
- if the previous two fail $\Rightarrow$ **certificate**: $k+1$ points at distance $> 2 \cdot r$.

Deletions:

- if x is not a center, delete x from its cluster.
- otherwise RECLUSTER all the points.

Dynamic Algorithm: r given, Random Deletions

Points can be inserted arbitrarily but are deleted randomly:

Insertions:

- insert x to any cluster whose center is at distance $\leq 2 \cdot r$ from x.
- if not possible and $< k$ clusters, form a new cluster with center x.
- if the previous two fail $\Rightarrow$ **certificate**: $k+1$ points at distance $> 2 \cdot r$.

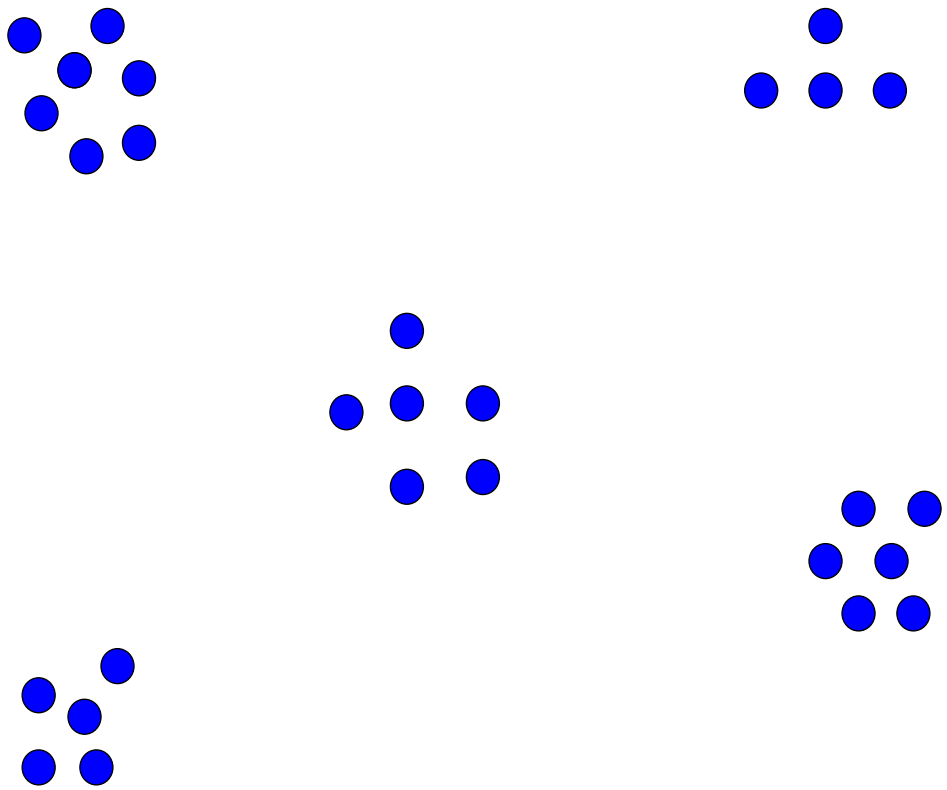
Deletions:

- if x is not a center, delete x from its cluster.
- otherwise RECLUSTER all the points.

Reclustering costs $O(k \cdot |X|)$ operations, but it happens with probability $\frac{k}{|X|} \Rightarrow$ expected amortized cost per operation is $O(k^2)$.

Random Deletions: not a Useful Model

In real-world applications, deletions are not random. The previous algorithm would be very expensive, if deletions are arbitrary.



Dynamic Algorithm, r given

Insertions: (no change)

- insert x to any cluster whose center is at distance $\leq 2 \cdot r$ from x.
- if not possible and $< k$ clusters, form a new cluster with center x.
- if the previous two fail $\Rightarrow$ **certificate**: $k+1$ points at distance $> 2 \cdot r$.

Dynamic Algorithm, r given

Insertions: (no change)

- insert x to any cluster whose center is at distance $\leq 2 \cdot r$ from x.
- if not possible and $< k$ clusters, form a new cluster with center x.
- if the previous two fail $\Rightarrow$ **certificate**: $k+1$ points at distance $> 2 \cdot r$.

Deletions:

- if x is not a center, delete x from its cluster.
- otherwise let $C_1, \ldots, C_k$ be the current clusters , U be the *unclustered* points, i.e. $U := X \setminus \cup_{i=1}^{k} C_i$. Let $x = c_i$. Recluster $\widehat{U} := \bigcup_{j=i}^{k} C_j \cup U$:

Dynamic Algorithm, r given

Insertions: (no change)

- insert x to any cluster whose center is at distance $\leq 2 \cdot r$ from x.
- if not possible and $< k$ clusters, form a new cluster with center x.
- if the previous two fail $\Rightarrow$ **certificate**: $k+1$ points at distance $> 2 \cdot r$.

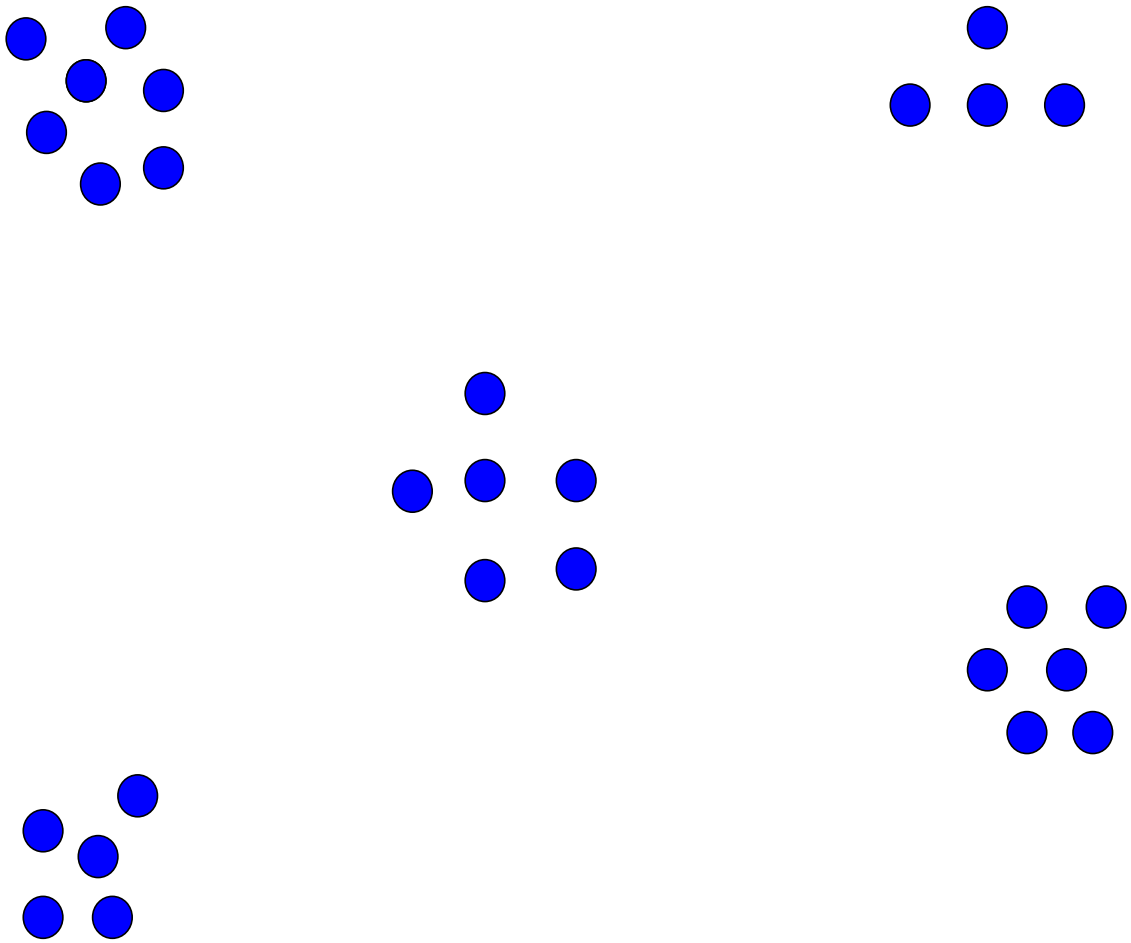
Deletions:

- if x is not a center, delete x from its cluster.
- otherwise let $C_1, \ldots, C_k$ be the current clusters , U be the *unclustered* points, i.e. $U := X \setminus \cup_{i=1}^{k} C_i$. Let $x = c_i$. Recluster $\widehat{U} := \bigcup_{j=i}^{k} C_j \cup U$:

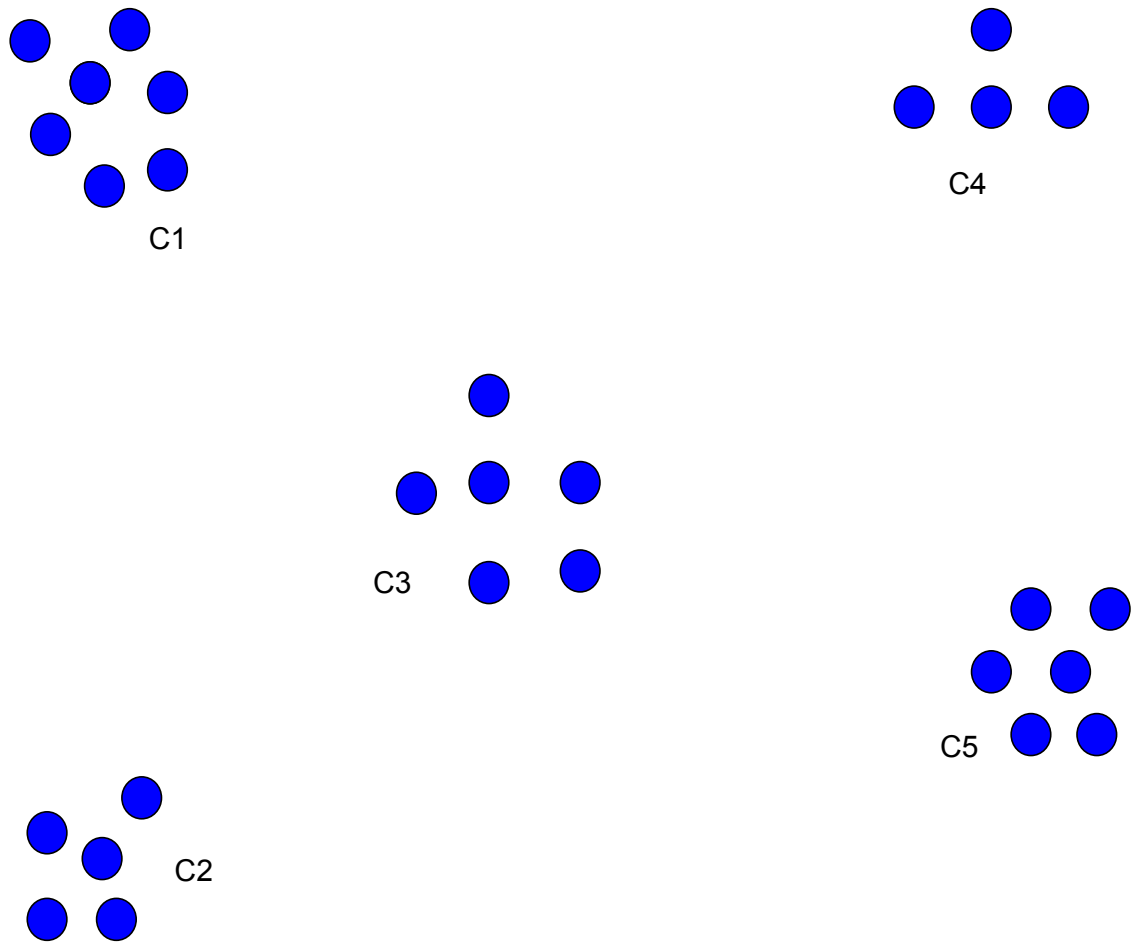
  At each step $j \in [1, k-i+1]$:

  1. pick one center c_j uniformly at random from $\widehat{U}$;
  2. create a cluster C_j with c_j and all points at distance $\leq 2 \cdot r$ from c_j;
  3. remove C_j from $\widehat{U}$.

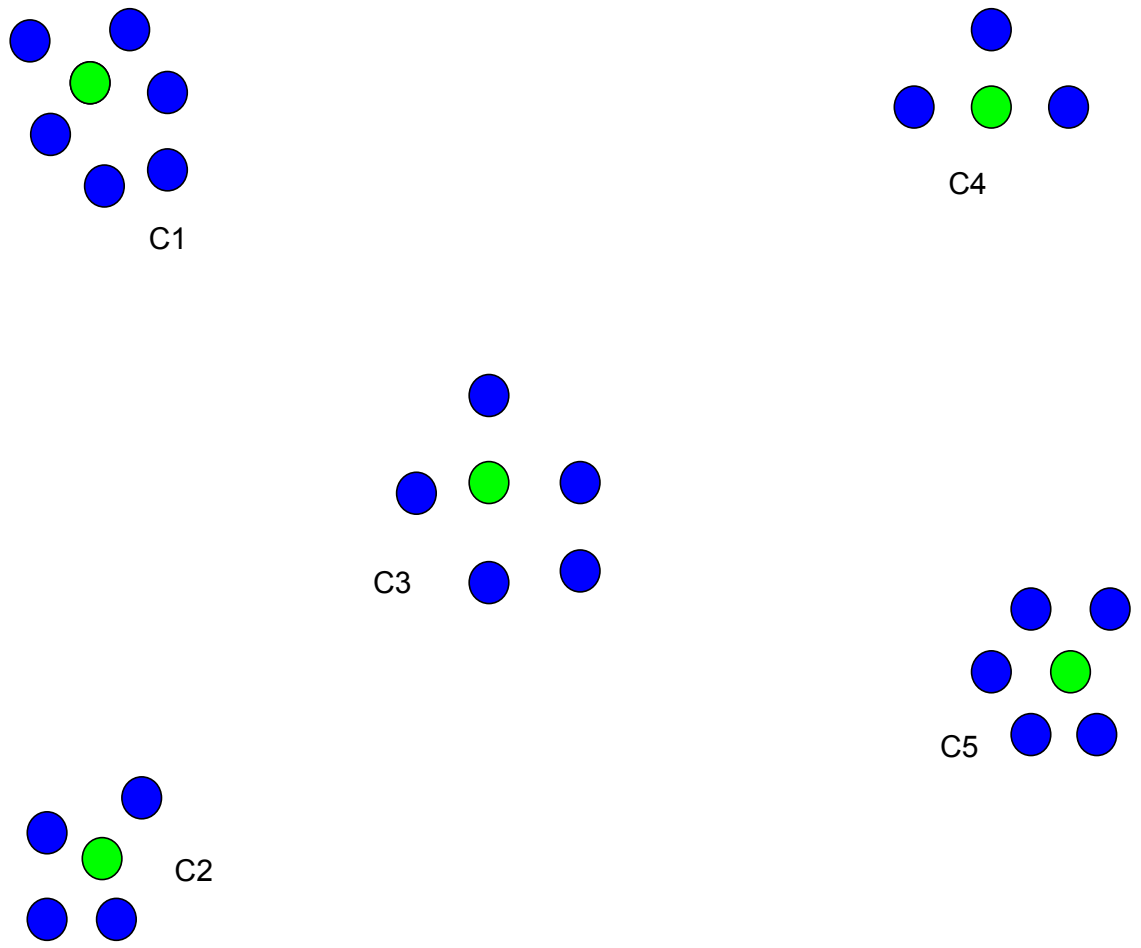
Arbitrary Deletions (given a guess r)



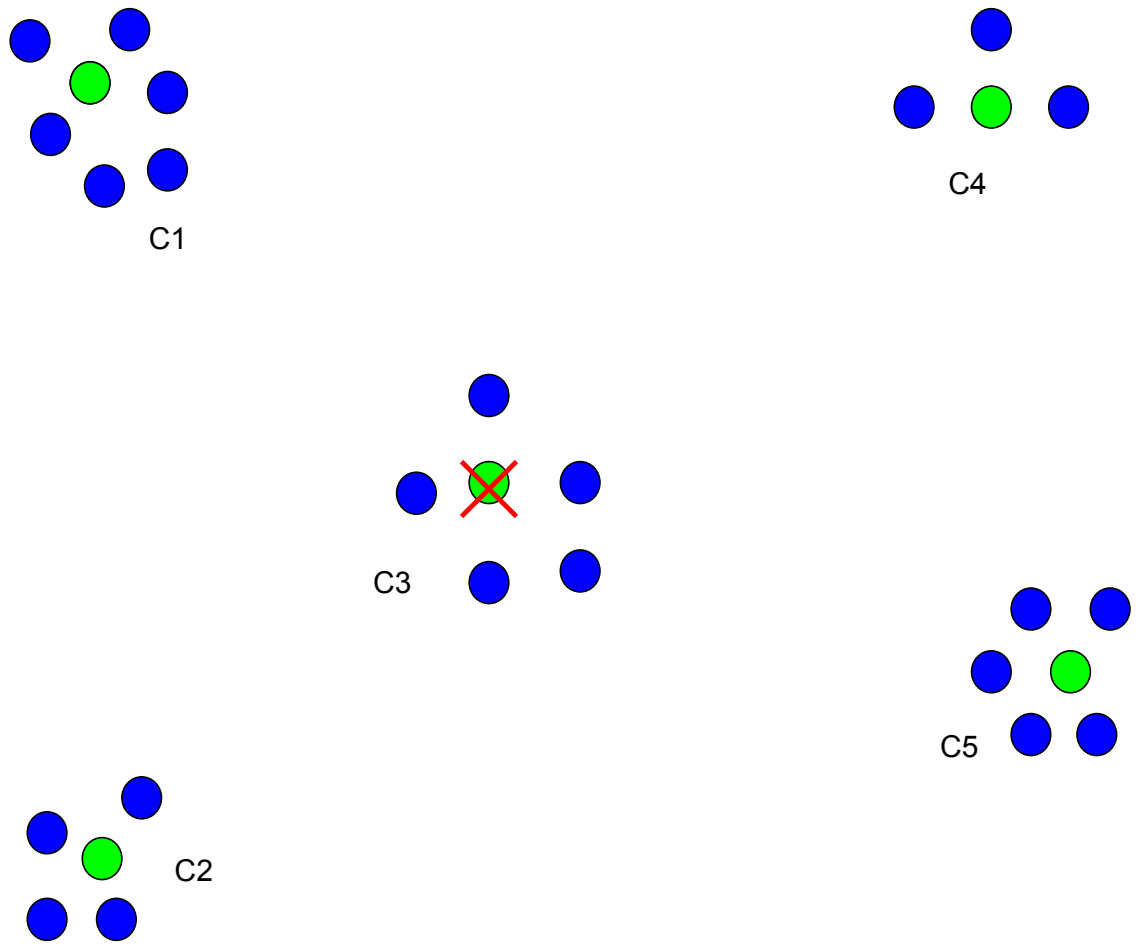
Arbitrary Deletions (given a guess r)



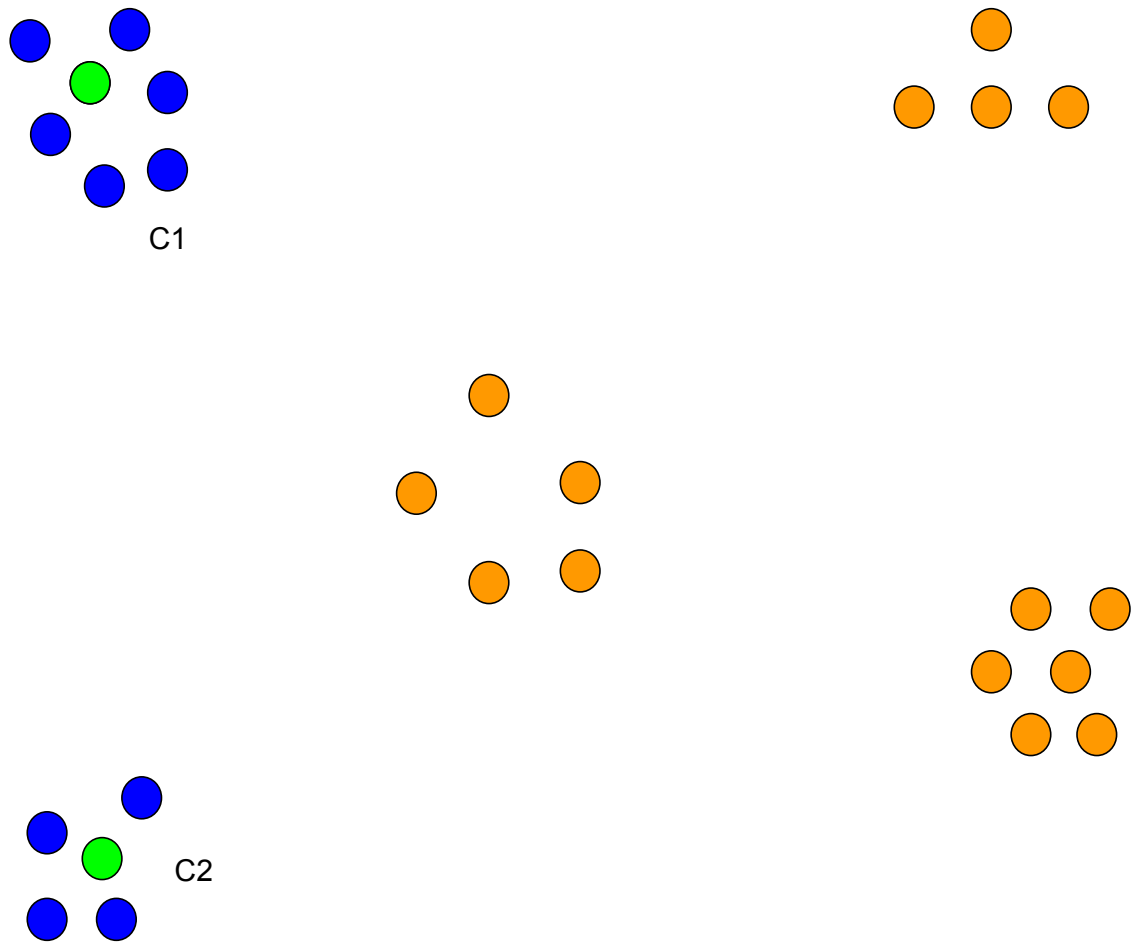
Arbitrary Deletions (given a guess r)



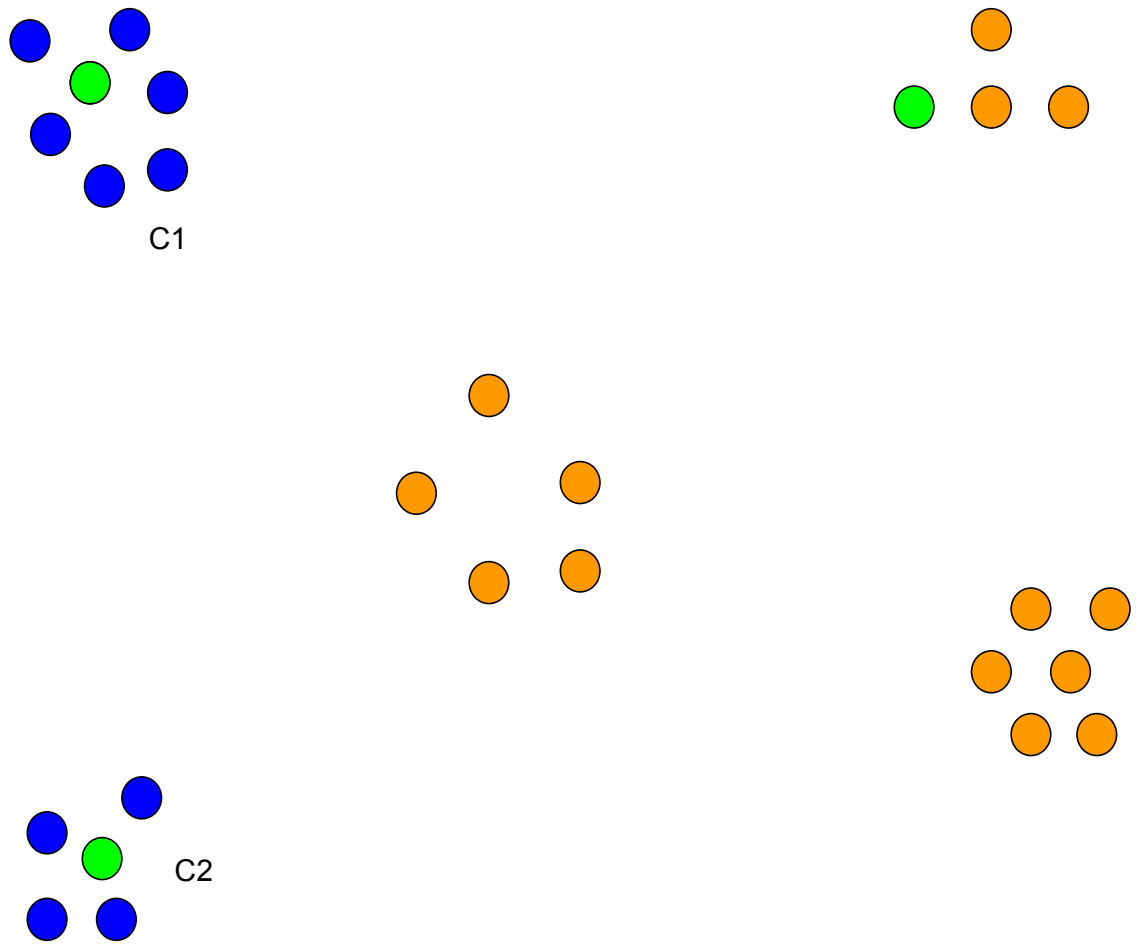
Arbitrary Deletions (given a guess r)



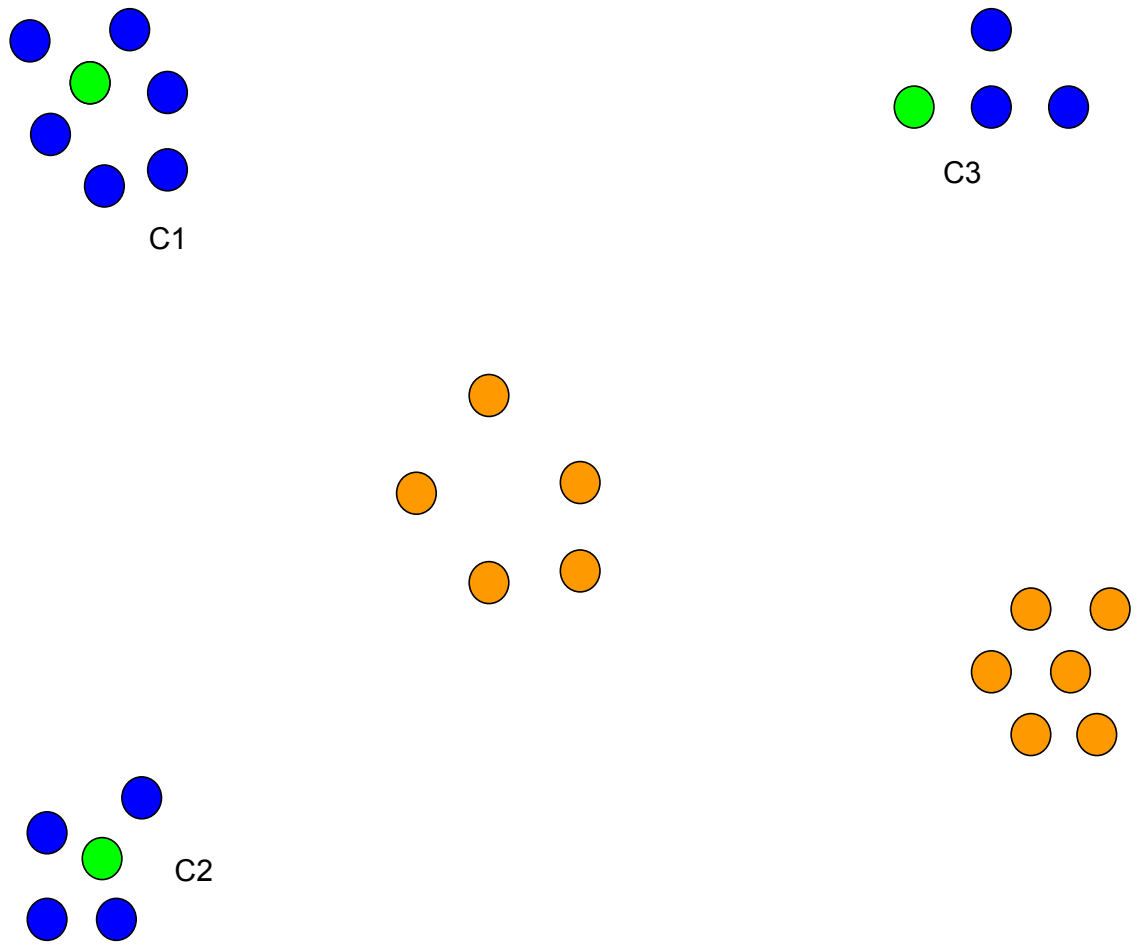
Arbitrary Deletions (given a guess r)



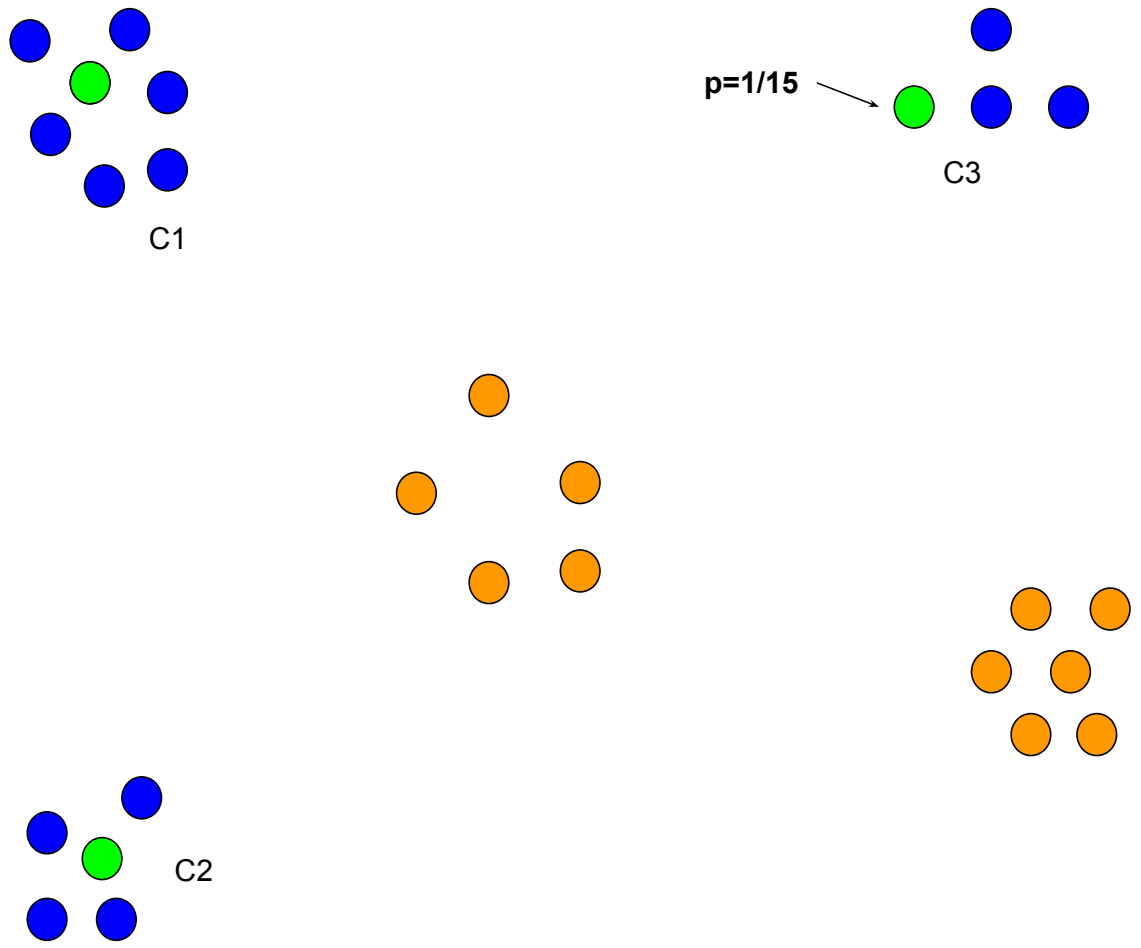
Arbitrary Deletions (given a guess r)



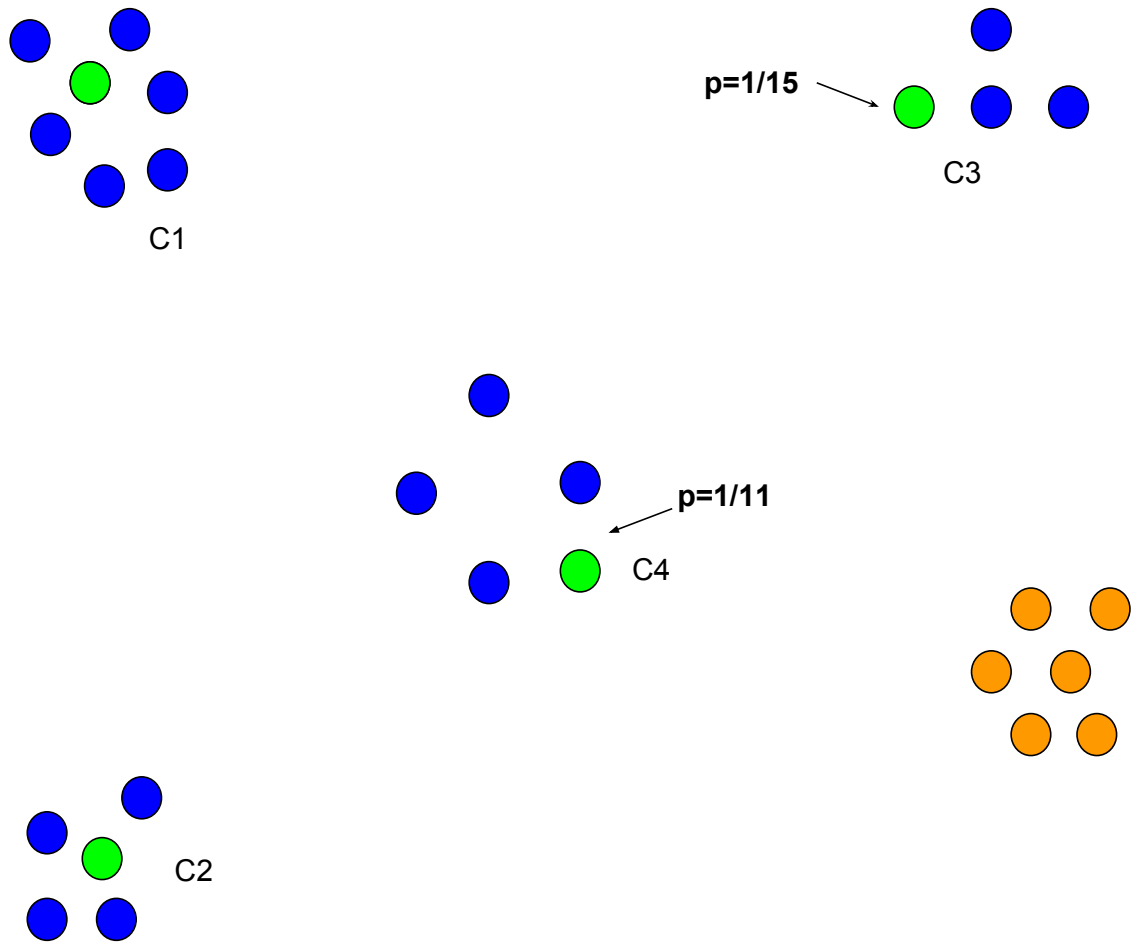
Arbitrary Deletions (given a guess r)



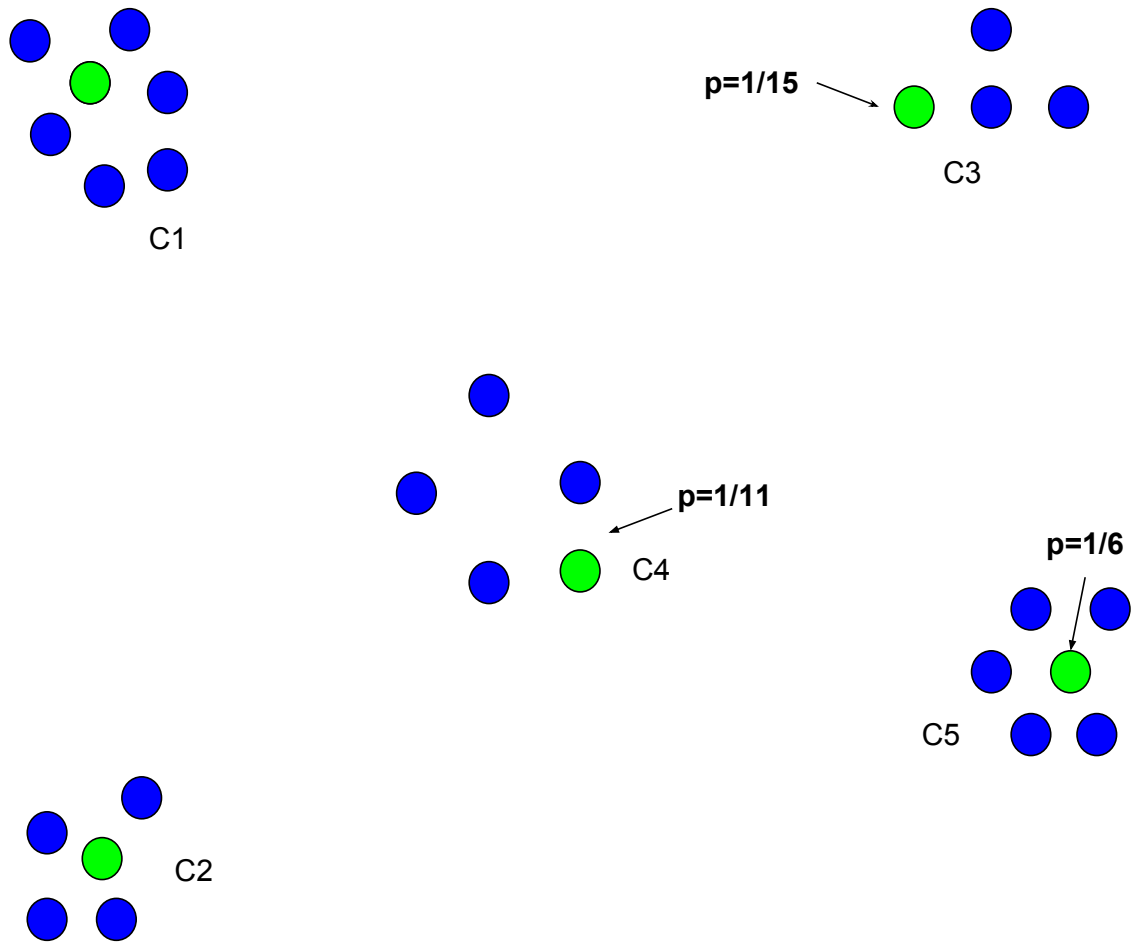
Arbitrary Deletions (given a guess r)



Arbitrary Deletions (given a guess r)



Arbitrary Deletions (given a guess r)



Roadmap

Amortized Cost Analysis

We wish to show that the total number of operations of our algorithm is not "much" larger than the number of update operations.

We refer to the solution computed for each guess r of the optimum solution as *level*. We analyze each level separately.

In our charging scheme, we distribute the cost incurred at each step t between two quantities F^t and Z^t, and bound each of them separately.

Amortized Cost Analysis: charging scheme

Whenever we insert a point or remove a non-center, we let $F^t = Z^t := 0$.

Otherwise, let c_i be the center being removed. The cost is $O(k \cdot \bar{X})$ where $\bar{X} := \bigcup_{j \geq i} C_j \cup U \setminus \{x\}$ which is distributed between F^t and Z^t as follows:

Amortized Cost Analysis: charging scheme

Whenever we insert a point or remove a non-center, we let $F^t = Z^t := 0$.

Otherwise, let c_i be the center being removed. The cost is $O(k \cdot \bar{X})$ where $\bar{X} := \bigcup_{j \geq i} C_j \cup U \setminus \{x\}$ which is distributed between F^t and Z^t as follows:

- Let $X_{\text{new}}(c_i) := \{u \in \bar{X} : u \text{ is inserted after } c_i \text{ is chosen as center}\}$, $F^t := k \cdot |X_{\text{new}}(c_i)|$.

Amortized Cost Analysis: charging scheme

Whenever we insert a point or remove a non-center, we let $F^t = Z^t := 0$.

Otherwise, let c_i be the center being removed. The cost is $O(k \cdot \bar{X})$ where $\bar{X} := \bigcup_{j \geq i} C_j \cup U \setminus \{x\}$ which is distributed between F^t and Z^t as follows:

- Let $X_{\text{new}}(c_i) := \{u \in \bar{X} : u \text{ is inserted after } c_i \text{ is chosen as center}\}$, $F^t := k \cdot |X_{\text{new}}(c_i)|$.
- Let $X_{\text{old}}(c_i) := \{u \in \bar{X} : u \text{ is inserted before } c_i \text{ is chosen as center}\}$, $Z^t := k \cdot |X_{\text{old}}(c_i)|$.

Amortized Cost Analysis: bounding F^t

Lemma 2

Let a_t be the number of insertions up to step t (included). It holds:

$$\sum_{\tau=1}^{t} F^{\tau} \leq a_t \cdot k^2.$$

Proof (sketch).

For every point u there are at most k sets $X_{\text{new}}(c_i)$'s that u belongs to. Those are the points c_i's that were centers when u was inserted. Hence:

$$\sum_{\tau=1}^{t} F^{\tau} \leq k \cdot |X_{\text{new}}(c_i)| \leq a_t \cdot k^2,$$

where c_i denotes the center deleted at step τ. □

Amortized Cost Analysis: bounding Z^t

Lemma 3

For each t, $\mathbb{E}[Z^t] \leq k^2$.

Proof (sketch).

Let $Z^t := \sum_{i=1}^{k} Z_i^t$, where Z_i^t is the cost incurred when the center c_i is deleted (at the beginning of step t). For any t, there is at most one i such that Z_i^t is non-zero.

Let U_i denote the set of points out of which c_i was chosen as center (in step 1 of a re-clustering step).

Dynamic Algorithm, r given (Recall)

Insertions:

- insert x to any cluster whose center is at distance $\leq 2 \cdot r$ from x.
- if not possible and $< k$ clusters, form a new cluster with center x.
- if the previous two fail $\Rightarrow$ **certificate**: $k+1$ points at distance $> 2 \cdot r$.

Deletions:

- if x is not a center, delete x from its cluster.
- otherwise let $C_1, \ldots, C_k$ be the current clusters , U be the *unclustered* points, i.e. $U := X \setminus \cup_{i=1}^{k} C_i$. Let $x = c_i$. Recluster $\widehat{U} := \bigcup_{j=i}^{k} C_j \cup U$:

  At each step $j \in [1, k-i+1]$:

  1. pick one center c_j uniformly at random from $\widehat{U}$;
  2. create a cluster C_j with c_j and all points at distance $\leq 2 \cdot r$ from c_j;
  3. remove C_j from $\widehat{U}$.

Amortized Cost Analysis: bounding Z^t (continue)

Proof (continue).

Observe that c_i was selected as center with prob. $\frac{1}{|U_i|}$ (conditioning on U_i).
Let $\bar{U} \subseteq U_i$ be the points that remain at the beginning of step t. It holds:

$$\mathbb{E}[Z_i^t | U_i] = \frac{1}{|U_i|} \cdot k \cdot |\bar{U}| \leq k.$$

The proof terminates by taking expectation of the random variable $\mathbb{E}[Z_i^t | U_i]$ and summing up over all i's.

Theoretical Results

Theorem 4

For any sequence of $T \in \mathbb{N}$ insert/delete operations, our algorithm:

- *always maintains a $2+\epsilon$-approximation;*
- *expected time: $O(\frac{\log \delta}{\epsilon} \cdot k^2 T)$;*
- *space: $O(\frac{\log \delta}{\epsilon} \cdot N)$;*
- *works in the fully dynamic model;*

N max size of input, $\delta = \frac{d_{\max}}{d_{\min}}$, $\epsilon > 0$.

Clustering membership can be tested in $O(1)$ time, while producing in output all points in the cluster C of a given point requires $O(k + |C|)$ time.

Theoretical Results: High Probability Statement

Theorem 5

For any sequence of $T \in \mathbb{N}$ insert/delete operations, our algorithm:

- *always maintains a $2+\epsilon$-approximation;*
- *expected time: $O(\frac{\log \delta}{\epsilon} \cdot k^2 T)$;*
- *space: $O(\frac{\log \delta}{\epsilon} \cdot N)$;*
- *works in the fully dynamic model;*
- *probability total time exceeds $\Omega(\frac{\log \delta}{\epsilon} \cdot \lambda k^2 T)$ is $\leq \frac{\log \delta}{\epsilon} \cdot k e^{-\Omega(\frac{\lambda T}{N})}$*

N max size of input, $\delta = \frac{d_{\max}}{d_{\min}}$, $\epsilon > 0$, $\lambda \geq 2$.

Clustering membership can be tested in $O(1)$ time, while producing in output all points in the cluster C of a given point requires $O(k + |C|)$ time.

Reducing Memory Requirements

Invariant. For each guess $r \in \Gamma$, compute a clustering with max radius $2 \cdot r$ or a certificate that there is no clustering with max radius r.

Weaker Invariant. For each $r \in \Gamma$, compute a clustering with maximum radius $4 \cdot r$ or a certificate that there is no clustering with max radius r.

Reducing Memory Requirements

Invariant. For each guess $r \in \Gamma$, compute a clustering with max radius $2 \cdot r$ or a certificate that there is no clustering with max radius r.

Weaker Invariant. For each $r \in \Gamma$, compute a clustering with maximum radius $4 \cdot r$ or a certificate that there is no clustering with max radius r.

Let $C_1^r, \ldots, C_k^r, U^r$ be the solution computed for a guess r, we obtain a solution $C_1^{2r}, \ldots, C_k^{2r}, U^{2r}$ for a guess $2 \cdot r$ as follows:

If the centers of C_i^r and C_j^r are at distance $\leq 4 \cdot r$ then merge the two clusters and obtain a cluster $C_i^r \cup C_j^r$ with center c_i and radius at most $8r$, $i < j$.

Reducing Memory Requirements

Invariant. For each guess $r \in \Gamma$, compute a clustering with max radius $2 \cdot r$ or a certificate that there is no clustering with max radius r.

Weaker Invariant. For each $r \in \Gamma$, compute a clustering with maximum radius $4 \cdot r$ or a certificate that there is no clustering with max radius r.

Let $C_1^r, \ldots, C_k^r, U^r$ be the solution computed for a guess r, we obtain a solution $C_1^{2r}, \ldots, C_k^{2r}, U^{2r}$ for a guess $2 \cdot r$ as follows:

If the centers of C_i^r and C_j^r are at distance $\leq 4 \cdot r$ then merge the two clusters and obtain a cluster $C_i^r \cup C_j^r$ with center c_i and radius at most $8r$, $i < j$. Move all points $x \in U^r$ to C_i^{2r}, if $d(x, c_i) \leq 8r$.

Reducing Memory Requirements (Continue)

Given $C_1^r, \ldots, C_k^r, U^r$, we can represent $C_1^{2r}, \ldots, C_k^{2r}, U^{2r}$ by storing only the difference with respect to $C_1^r, \ldots, C_k^r, U^r$ aka *compact representation*.

The algorithm stores explicitly all clusters with guess r, $r < 2d_{\min}$, $r \in \Gamma$. We employ a compact representation for the other clusters.

It requires $O(N \cdot \log \frac{1}{\epsilon} + k \cdot N)$ space, where $N := \max_{t \in [T]} |X^t|$.

Running time worsens by a factor of $\log \delta$ (the randomness in the first levels pay for the other levels).

Comparison: Theory

	approx.	avg. run. time	space	model
FD	$2+\epsilon$	$O(k^2 \cdot \frac{\log(\delta)}{\epsilon})$	$O(N \cdot \frac{\log(\delta)}{\epsilon})$	Fully Dyn.
FD2	$4+\epsilon$	$O(k^2 \cdot \frac{\log^2(\delta)}{\epsilon})$	$O(N \cdot \log \frac{1}{\epsilon} + k \cdot N)$	Fully Dyn.
SW	$6+\epsilon$	$O(k \cdot \frac{\log(\delta)}{\epsilon})$	$O(k \cdot \frac{\log(\delta)}{\epsilon})$	Sliding Window

Table: Our algorithm (FD) vs the sliding-window algorithm (SW)

	model	$x \in C$?	list all y s.t. $x, y \in C$
FD	Fully Dyn.	$O(1)$	$O(\|C\|)$
FD2	Fully Dyn.	$O(\|C\|)$	$O(\|C\|)$
SW	Sliding Window	$O(k)$	$k \cdot N$

Table: Our algorithm (FD) vs the sliding-window algorithm (SW)

Roadmap

Settings

- Machine equipped with two Intel(R) Xeon(R) CPU E5-2660 v3 running at 2.60GHz and with 250Go of DDR3 SDRAM.
- default values: $k = 20$, $\epsilon = 0.1$, size of the slid. window: $60k$ points.

We evaluate running time and quality of the solution.

Code and datasets available at `https://github.com/fe6Bc5R4JvLkFkSeExHM/k-center`.

Datasets

Twitter. 21M tweets (09/09/17-20/10/17) with GPS and timestamps.

Flickr. 47M pictures with GPS and timestamps from The Yahoo Flickr Creative Commons.

Trajectories. 83M $2D$-points from trajectories performed by the 442 taxis in the city of Porto, Portugal. Trajectories are updated every 15 seconds.

Distance Measures

We consider the following distance measures:

- Twitter, Flickr: great circle distance between two GPS coordinates;
- taxi trajectories: symmetric Hausdorff distance defined as the max between $H(T_1, T_2)$ and $H(T_2, T_1)$ with

$$H(T_1, T_2) := \max_{p \in T_1} \min_{q \in T_2} d(p, q),$$

where T_1, T_2 are two trajectories (sets of points), d is a distance.

Dynamic Environments

We consider the following dynamic environments:

- Twitter, Flickr: sliding window each point inserted at time t is removed at time $t + W$, where W is the length of the sliding window.
- taxi trajectories: at each point in time, either a new point is added to one trajectory or a new trajectory is created.

Not clear how to adapt algorithms for sliding window for the latter case.

Comparison

We compare our algorithm (FD) against the algorithm developed in the sliding window model for k-center clustering (SW).

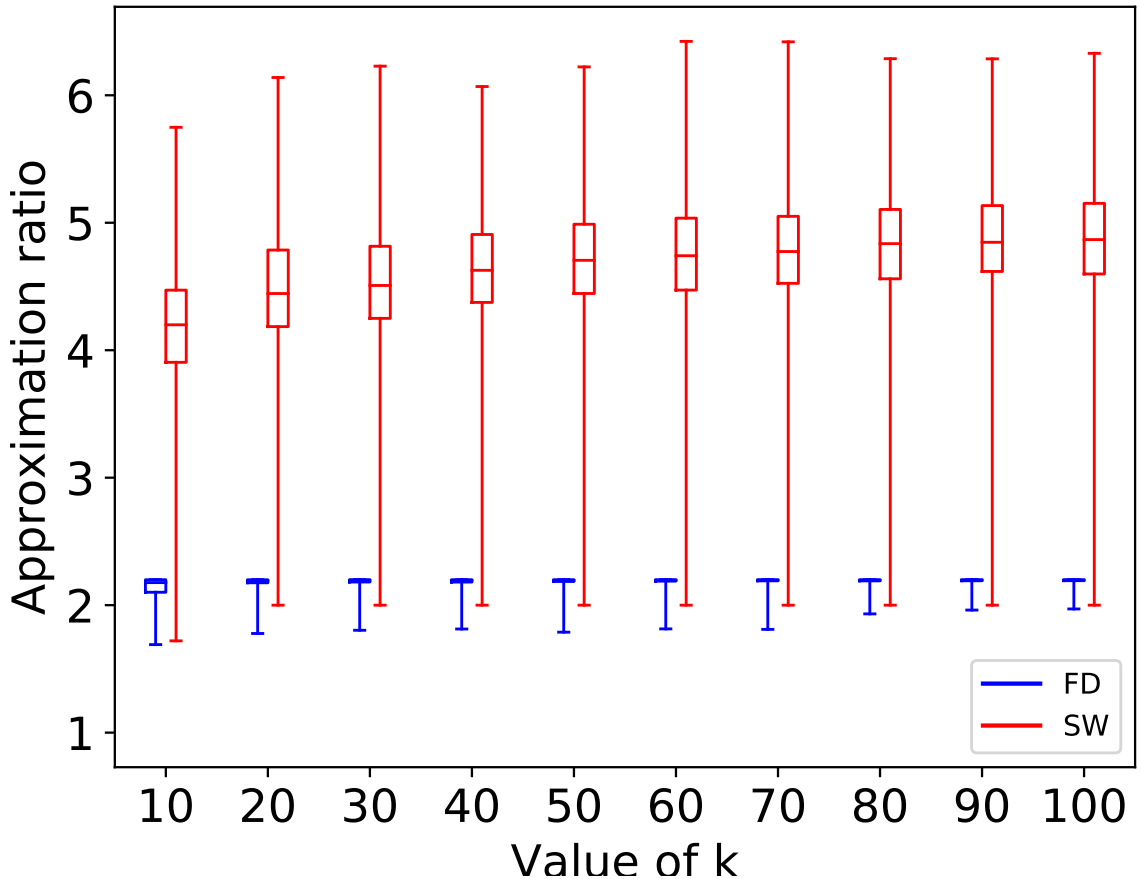
We evaluate:

- approximation ratios (how far from the opt?);
- running time;

Lower Bound on the opt: half the minimum pairwise distance between $k+1$ points (we take the max among all $r \in \Gamma$).

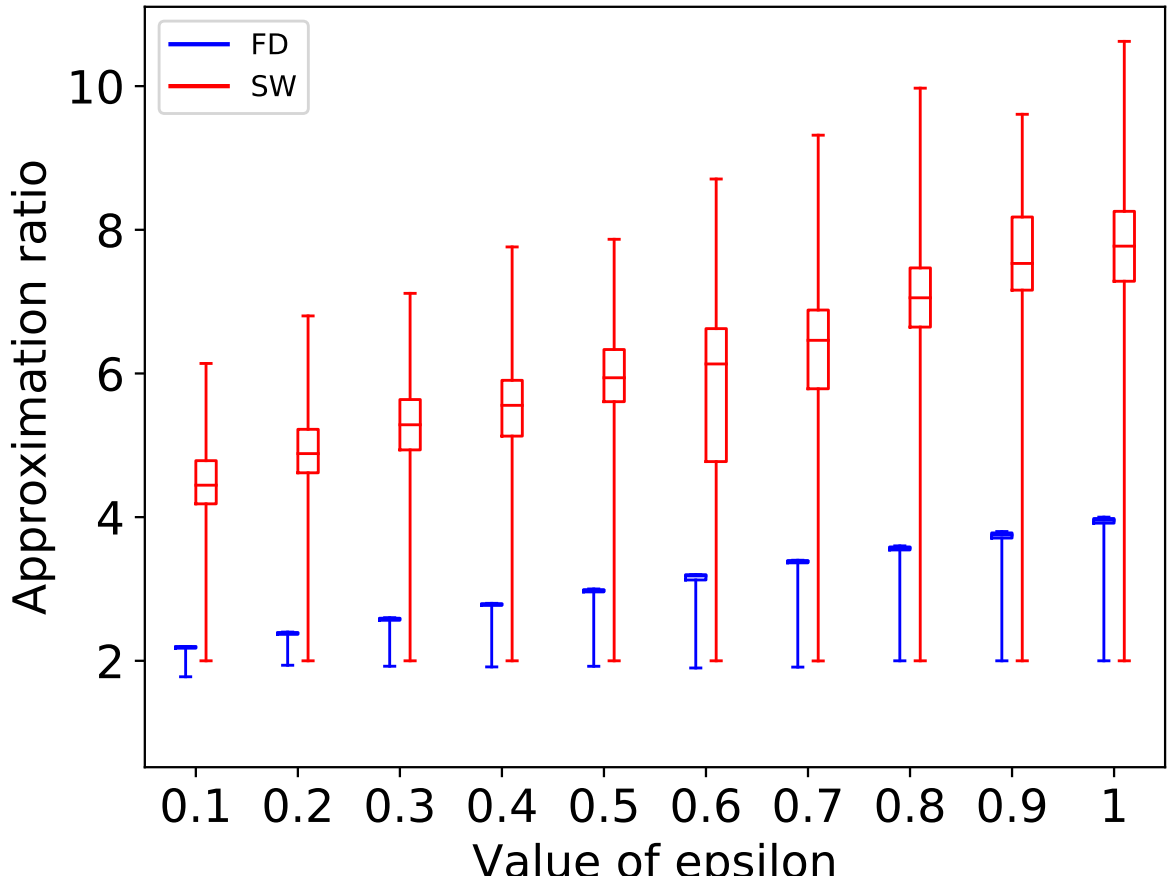
Approximation ratio vs. k

Picture shows the median, first and third quartiles, max and min approximation ratios, when running the algorithm on the whole dataset.

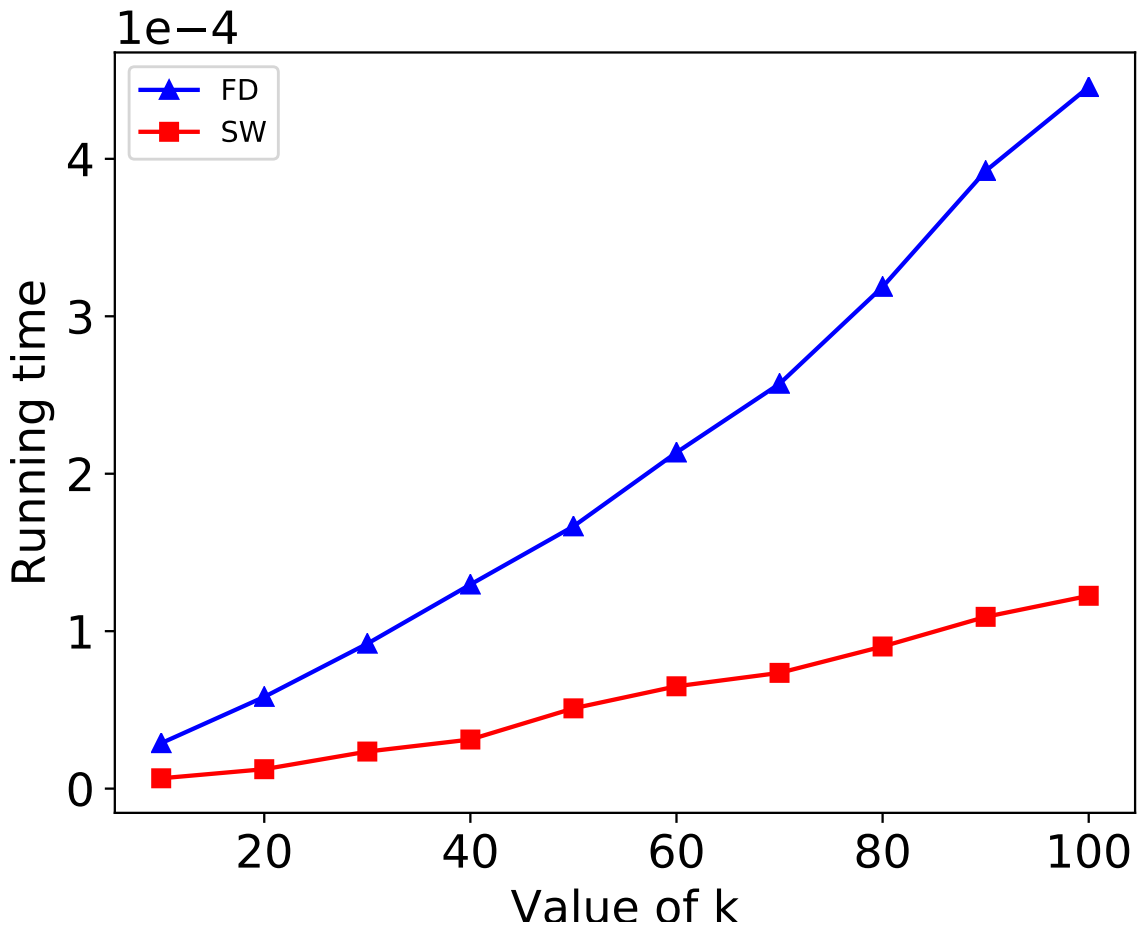


Approximation ratio vs. ϵ

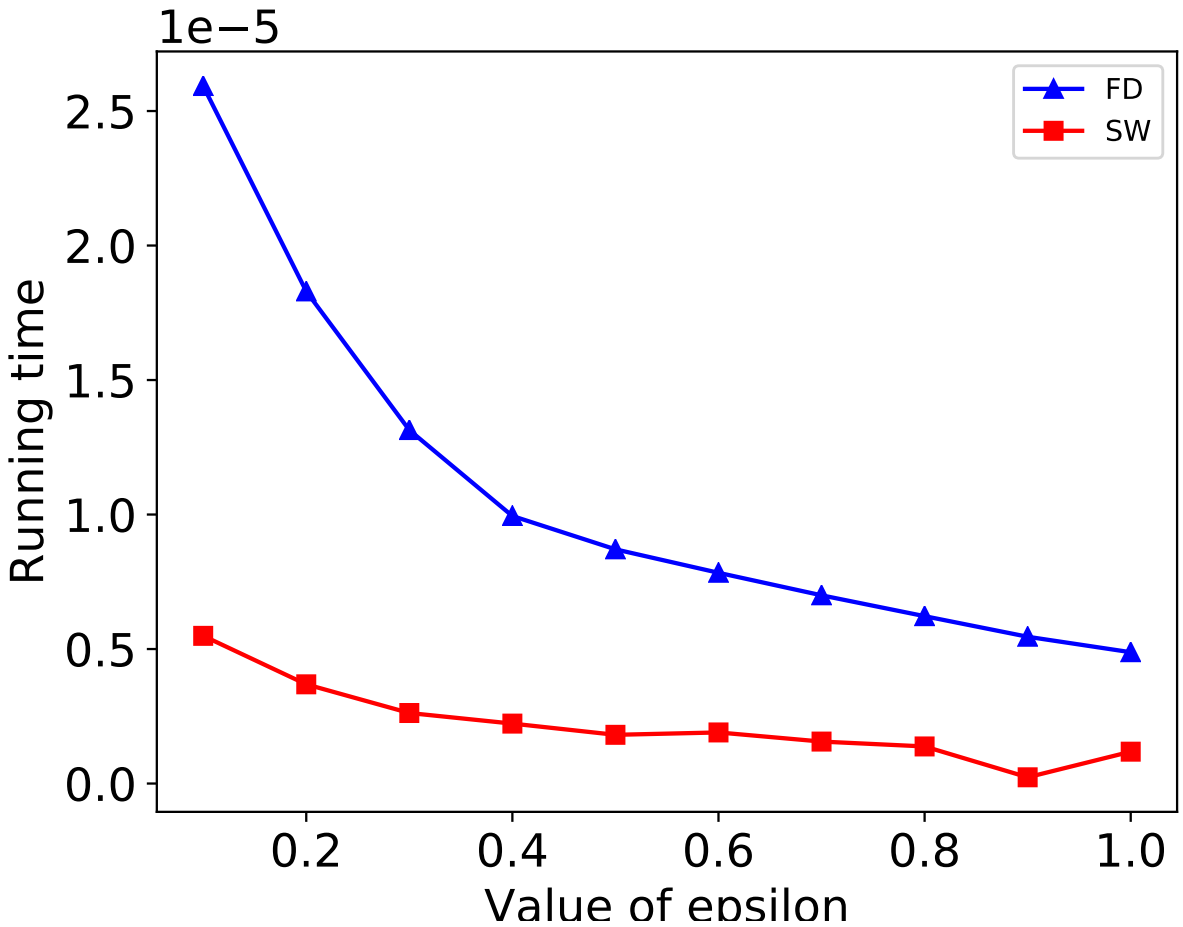
Picture shows the median, first and third quartiles, max and min approximation ratios, when running the algorithm on the whole dataset.



Avg. Running time per update (in secs) vs. k



Avg. Running time per update (in secs) vs. ϵ



Number of runs (out of 100) vs. avg. running time

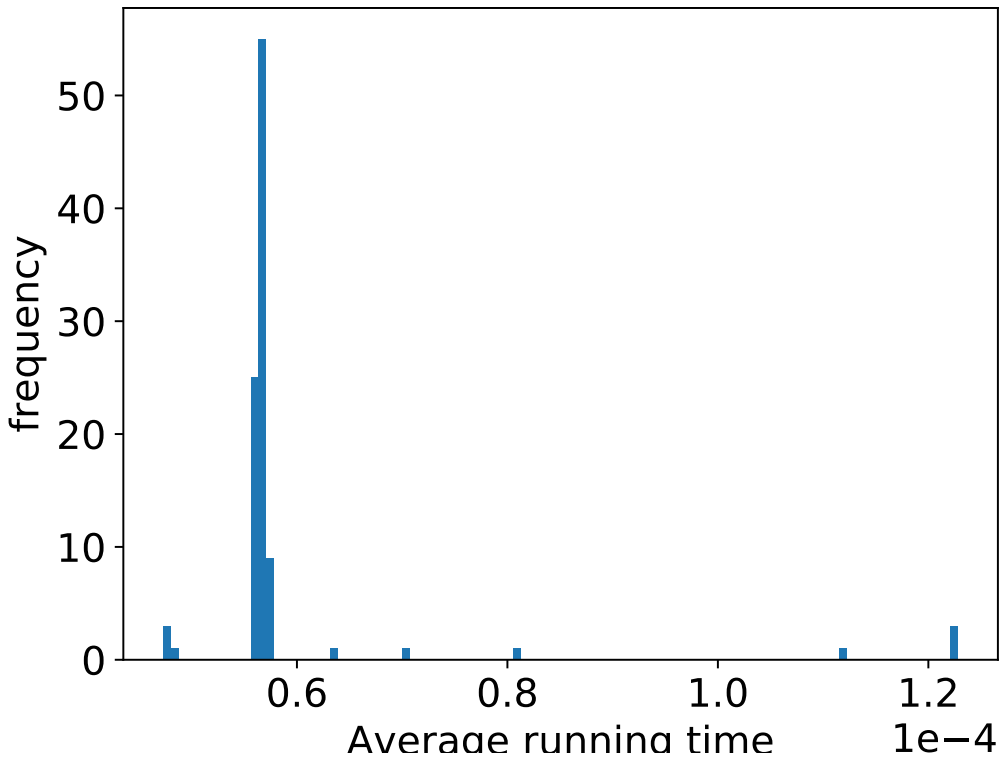


Figure: concentration of measure experiments

Trajectories

Recall:

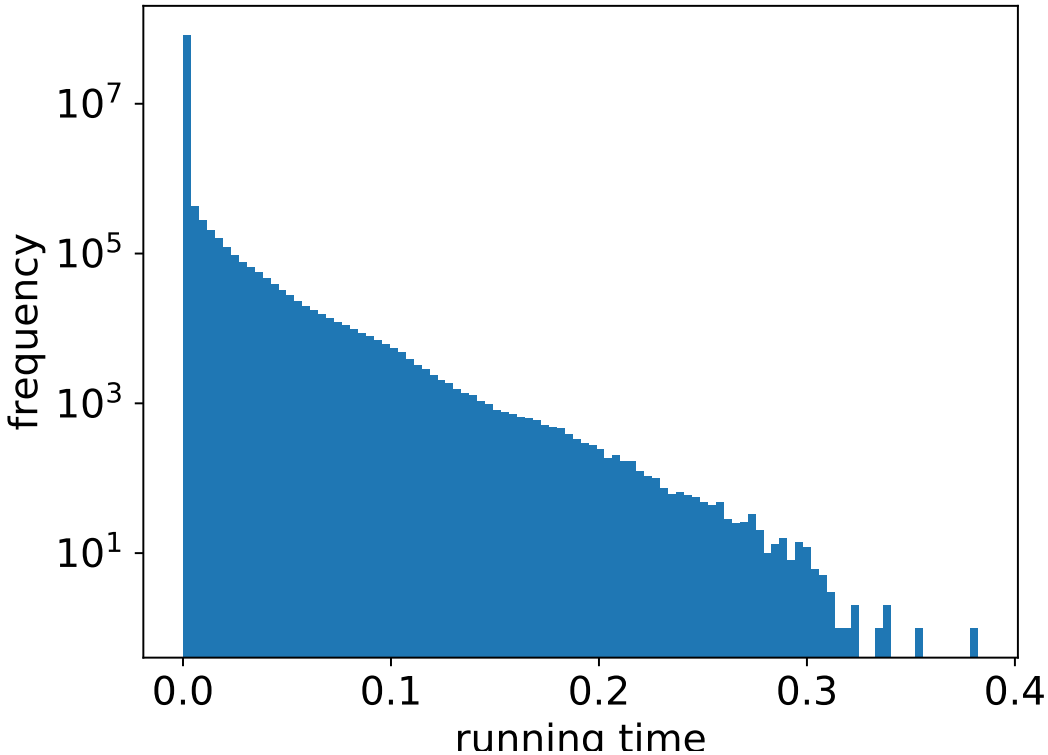
- use Hausdorff distance to estimate distances between trajectories.
- Trajectories updated every 15 seconds in arbitrary order $\Rightarrow$ not trivial to adapt SW algorithms.

More expensive $\Rightarrow$ we use a multicore implementation (30 cores) parallelized over the r's in Γ.

Approx. ratio and Running time

Approx. ratios: median: 2.05, first quartile: 2.03, third quartile: 2.13.

Histogram of the running time of update operations (one single run), $80M$ operations require less than 0.004 secs.



Roadmap

Conclusions

First fully dynamic algorithm for k-center clustering with provable guarantees and efficient in practice. Poly-logarithmic update cost.

Close to the best we can do:

- approx guarantee better than 2 is Np-Hard;
- less than $N^{\frac{1}{3}}$ memory not possible for any 4-approx.

Experimental evaluation:

- Running time per operation $\approx 10^{-4}$ secs most of the times
- approx. ratio is around 2 or better.
- outperforms state-of-the-art: smaller radius, fully dynamic (but more space, slightly less efficient)

Future Work

k-center Clustering:

- Sublinear space? can we match the lower bound?
- adapt other variants of k-center in a fully dynamic environment?

Other directions:

- Can other clustering algorithms (k-means++, etc.) be adapted into a fully dynamic environment?
- Other data mining/machine learning algorithms?

Future Work

k-center Clustering:

- Sublinear space? can we match the lower bound?
- adapt other variants of k-center in a fully dynamic environment?

Other directions:

- Can other clustering algorithms (k-means++, etc.) be adapted into a fully dynamic environment?
- Other data mining/machine learning algorithms?

Questions?