Figure 1: The divide-and-conquer algorithm “Small space” devised for clustering data points with space (memory) restrictions. The initial set of points $S$ is divided into $l$ partitions. For each of the $l$ partitions, $O(k)$ medians are found. The $O(lk)$ medians are then considered independently as a set $\chi'$. Then, $k$ medians are found for $\chi'$ and returned as the approximate solution. Here $k = 2$ and $l = 4$. The weights $w_{ij}$ are used to weight the centers found in each intermediate stage with $w_{ij} =$ the number of points assigned to center $j$ in partition $i$, thereby placing more weight on centers that are close to many points.

Summary: Clustering Data Streams: Theory and Practice, by Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani

Main Point: The authors give a short overview of streaming algorithms for clustering, then describe a divide-and-conquer clustering approach which achieves a constant-factor approximation of the k-median problem. Then, they show how to restructure this algorithm for the data stream setting, where only a limited amount of data fits into memory and the $k$ medians must be found in a linear number of passes over the data.

The divide-and-conquer approach is shown in Figure 1. Here, a set $S$ is divided into $l$ partitions $\{\chi_1, \ldots, \chi_l\}$. In the figure, $l = 4$. Then, $O(k)$ medians are found for each partition $\chi_i$, and each median is weighted by the number of points assigned to (closest to) it. Notice that in Figure 1, although more $k = 2$ medians may be found for each $\chi'$. The $O(lk)$ medians are aggregated in the set $\chi'$. $k$ medians are found for $\chi'$, and either (1) the process is repeated, i.e., the algorithm recurses on $\chi'$, or (2) a given number of iterations has passed and these medians are presented as the solution. The authors prove that for $i$ iterations, this algorithm achieves a constant-factor approximation.
to the $k$–Median problem on $S$. More formally, for the cost function $f$ over a set of points $S$ partitioned into $k$ clusters with medians $C_i$, $f(S, C_1, \ldots, C_k)$ is defined as:

$$f(S, C_1, \ldots, C_k) = \sum_{x \in S} \min_{1 \leq i \leq k} d(x, C_i)$$

Thus, the cost of a clustering is the sum of distances between each point and the median $C_i$ closest to that point. The cost of a set of points $S$, cost($S, S$) is defined as the minimum $f(S, C_1, \ldots, C_k)$ over all possible clusterings with $k$ medians. If the medians are required to be contained in a set $Q$, then cost($S, Q$) denotes the smallest possible cost attainable with this requirement for the $k$ medians.

An important result stated in the paper follows from a series of theorems relating cost($S, S$) to cost($S, Q$) as follows:

**Theorem 1** Given an instance $(S, k)$ of $k$-Median, cost($S, S$) $\leq 2$cost($S, Q$) for any $Q$.

**Theorem 2** Consider an arbitrary partition of a set $S$ of $n$ points into $\chi_1, \ldots, \chi_l$. Then, $\sum_{i=1}^l$cost($\chi_i, \chi_i$) $\leq 2$cost($S, S$).

**Theorem 3** If $C = \sum_{i=1}^l$cost($\chi_i, \chi_i$) and $C^* = $cost($S, S$) $= \sum_{i=1}^l$cost($\chi_i, S$), then there exists a solution of cost at most $2(C + C^*)$ to the new weighted instance $\chi'$.

Recall here that $\chi'$ is the set of $O(lk)$ medians that were calculated for the $l$ partitions of $S$ in the first step of the divide-and-conquer algorithm.

**Theorem 4** The algorithm small-space has an approximation factor of $2c(1 + b) + 2b$, where a $b$-approximation is used in the initial clustering phase where $O(k)$ centers (medians) are found for each of the $l$ partitions, and a $c$-approximation is used to find the $k$ medians in $\chi'$.

Thus, the small-space algorithm illustrated in Figure 1 is guaranteed to find a set of centers $C_i$ for which cost($f, C_1, \ldots, C_k$) is at most $(2c(1+b) + 2b)C^*$, where $C^*$ is the optimal (minimum) cost($S, S$). These results are used to generalize the small-space algorithm into a divide-and-conquer approach, which follows the same procedure used by small-space, but can keep iterating over the centers found in each partition. In other words, instead of using $\chi'$ to find the final $k$ centers, we can call small-space again on $\chi'$, dividing it into $l$ partitions, finding $O(k)$ centers for each partition, each center weighted by the number of points assigned to it, and so on, until the desired number of iterations is reached. The authors prove that this divide-and-conquer approach will also give a constant-factor approximation to the $k$-Median problem:

**Theorem 5** We can solve the $k$-Median problem on a data stream in $O(n^{1+\epsilon})$ time and $\Theta(n^\epsilon)$ space up to a factor $2^{O(\frac{1}{\epsilon})}$.
Next, the algorithm is modified further to speed up the running time. In this new version, sampling is used. Assuming we desire $k$ clusters, the algorithm is described by the following steps:

1. Draw a sample of size $s = \sqrt{nk}$
2. Find $k$ medians from these $s$
3. Assign each of the $n$ original points to their closest median
4. Collect the $\frac{n}{s}$ points with the largest assignment distance
5. Find $k$ medians among these $\frac{n}{s}$ points.

After step 5, there are $2k$ medians. Theorem 7 says:

**Theorem 6** The above algorithm gives an $O(1)$ approximation with $2k$ medians with constant probability.

The above algorithm also requires $O(nk)$ time and space. The authors use the sampling-based algorithm to develop a one-pass, $O(nk)$-time algorithm that requires only $O(n^\epsilon)$ space. This algorithm is described by the following steps, where $M$ is the size of available memory:

1. Input the first $O(M/k)$ points and use the randomized algorithm to cluster this set into $2k$ intermediate points
2. Use a local search algorithm to cluster the $O(M)$ intermediate medians of level $i$ into $2k$ medians in level $i + 1$
3. Cluster the final $O(M)$ medians into $k$ medians.

The algorithm above leads to the following theorem, if we assume that in step (3) we use a primal dual algorithm to cluster the $O(M)$ medians into $k$ medians:

**Theorem 7** The $k$-median problem has a constant-factor approximation algorithm running in time $O(nk \log n)$ in one pass over the data set, using $n^\epsilon$ memory for small $k$ ($k << M$).

The authors show that any constant-factor deterministic approximation algorithm for the $k$-medians problem requires $\Omega(nk)$ time, and extend this result to randomized algorithms as well. Thus, any $k$-Median algorithm requires $\Omega(nk)$ time to achieve a constant factor approximation.

In Section 4, the authors discuss further improvements by varying the number $k$ of intermediate medians found at each level of the divide-and-conquer algorithm. To do this, they use a facility-location based algorithm to find $O(k)$ facilities at each intermediate stage. The facility location algorithm uses a regularization term to penalize the addition of too many new facilities, but allows for the formation of new facilities with probability proportional to the corresponding point’s distance from the closest of the current set of facilities. This
facility location algorithm is used to seed the initial $O(k)$ centers in the divide-and-conquer algorithm. Then, a gain function is used to perform a local search which attempts to minimize the cost function by adding a facility and reassigning points to their closest facility center. If the number of facilities (aka centers aka medians) after this stage is between $k$ and $2k$, then return this solution to the final clustering stage which finds $k$ medians. Otherwise, repeat the process, revising the regularization parameter and the initial set of facilities based on the previous iteration’s result. This facility-location based, divide-and-conquer algorithm is the final complete version of the streaming algorithm presented.

Section 5 is an empirical evaluation of the divide-and-conquer, facility-location based algorithm. This algorithm, called LSEARCH, is compared to the BIRCH algorithm as well as $k$-means on various synthetic datasets. All synthetic clusters are spherical in shape and lie in a grid, either at regular intervals, at regular intervals with a random small perturbation added to the cluster center, or randomly spaced inside the grid. As the asymmetry of cluster locations increases from regular to random layout, LSEARCH seems widen the gap between its good performance and the results of $k$-means. On high-dimensional datasets, where the “curse of dimensionality” affects the $k$-means algorithm, LSEARCH again showed significantly better results. However, for all these datasets the $k$-means algorithm was faster than LSEARCH. In a real data application, where the goal was to classify TCP dump data into either a normal class or one of 4 network intrusion classes, the divide-and-conquer streaming algorithm again showed higher quality but slower runtimes than BIRCH.