

On Scaling Up Balanced Clustering Algorithms*

Arindam Banerjee and Joydeep Ghosh †

1 Introduction

The past few years have witnessed a growing interest in clustering algorithms that are suitable for data-mining problems [15, 14, 9]. Clustering algorithms for data-mining problems must be extremely scalable. In addition, several data mining applications demand that the clusters obtained be balanced, i.e., be of approximately the same size or importance. There are several notable approaches that address the scalability issue. Some approaches try to build the clusters dynamically by maintaining sufficient statistics and other summarized information in main memory while minimizing the number of database scans involved. For example, Bradley *et al.* [4, 5] propose out-of-core methods that scan the database once to form a summarized model (for instance, the size, sum and sum-squared values of potential cluster, and well as a small number of unallocated data-points) in main memory. Subsequent refinement based on this summarized information is then restricted to main memory operations without resorting to further disk scans. Another method with a similar flavor [24] compresses the data objects into many small subclusters using modified index trees and performs clustering with these subclusters. A different approach is to subsample the original data before applying the actual clustering algorithms [6, 12]. Ways of effectively sampling large datasets have also been proposed [20]. A recent work [8] suggests using less number of points in each step of an iterative relocation optimization algorithm like k-means as long as the model produced does not differ significantly from the one that would be obtained with full data.

Several of these methods are linear in the number of data-points N as well as the number of clusters k and hence scale very well. However their “sequential cluster building” nature prevents a global view of the emergent clustering that is needed for obtaining well-formed as well as reasonably balanced clusters. Even the

*This work was supported in part by Intel, and the NSF under grant ECS-9000353

†Dept. of Electrical and Computer Engineering, University of Texas at Austin, TX 78712

venerable k-means does not have any explicit way to guarantee that there is at least a certain minimum number of points per cluster, though it has been shown that k-means has an implicit way of preventing highly skewed clusters [17]. Experiments with k-means show that it quite often generates some clusters that are empty or extremely small, specially when the data is in high dimensional (> 100) space. This is undesirable in a lot of practical scenarios where clusters should be of comparable sizes to be actually useful and actionable. For example, a direct marketing campaign often starts with segmenting customers into groups of roughly equal size or equal estimated revenue generation, (based on, say, market basket analysis, or purchasing behavior at a web site), so that the same number of sales teams (or marketing dollars) can be allocated to each segment. In large retail chains, one often desires product categories/groupings of comparable importance, since subsequent procedures such as shelf/floor space allocation and product placement are influenced by the objective of allocating resources proportional to revenue or gross margins associated with the product groups. In clustering of a large corpus of documents to generate topic hierarchies, by avoiding the generation of hierarchies that are highly skewed, with uneven depth in different parts of the hierarchy “tree” or having widely varying number of documents at the leaf nodes, balancing greatly facilitates navigation.

There are a few existing approaches for obtaining balanced clusters. First, an agglomerative clustering method can be readily adapted so that once a cluster reaches a certain size in the bottom-up agglomeration process, it can be removed from further consideration. However, this may significantly impact cluster quality. Moreover, agglomerative clustering methods have $O(N^2)$ complexity and hence does not scale well. A recent approach to obtain balanced clusters is to convert the clustering problem into a graph partitioning problem [22]. A weighted graph is constructed whose vertices are the data-points. An edge connecting any two vertices has a weight equal to a suitable similarity measure between the corresponding data-points. The choice of the similarity measure quite often depends on the problem domain, e.g., Jaccard coefficient for market-baskets, normalized dot products for text, etc. The resultant graph can be partitioned by efficient “min-cut” algorithms that also incorporate a soft balancing constraint [16]. Though this approach gives very good results, $O(N^2)$ computation is required to compute the similarity matrix. Another recent approach is to iterate over k-means but do the cluster assignment by solving a minimum cost flow problem satisfying constraints [3] on the cluster sizes. This approach is $O(N^3)$ and thus has even poorer scaling properties.

In this paper, we address the issue of developing scalable clustering algorithms that satisfy balancing constraints on the cluster sizes. A $O(kN \log N)$ scheme is presented for clustering N data-points into k clusters so that each cluster has at least m points for some given $m \leq \frac{N}{k}$. The proposed scheme can be broken down into three steps - sampling, soft-balanced clustering and populating the clusters while keeping the balance. We address each of the steps separately and show that the three-step process gives a very general methodology for scaling up balanced clustering algorithms. The rest of the paper is organized as follows. A brief overview of the three steps is presented in section 2. The detailed algorithms and analysis of the three steps are provided in section 3. In section 4 we present experimental results

on high-dimensional text clustering problems. The paper concludes in section 5 with a discussion on the proposed scheme and future work.

2 Overview

First, we present the basic model and assumptions based on which the algorithms will be proposed. Let $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ be a set of data-points that needs to be clustered. We assume that there are k underlying clusters and \mathcal{X} has approximately equal number of data points from each cluster. To be more precise, the number of points from each cluster is assumed to be at least m for some $m \leq \frac{N}{k}$. The three steps in the proposed scheme are as follows:

Step 1 *Sampling of the given data:* The problem with sampling is that one may end up sampling all the points from the same cluster so that the sampled data is not suitable for clustering. We show that this kind of a scenario is less likely under our balancing assumptions and compute the number of samples we must draw from the original data in order to get a good representation of each of the clusters in the sampled set with high probability.

Step 2 *Clustering of the sampled data:* Any known balanced clustering algorithm that fits the problem domain can be used in this stage. The algorithm is to be run on the sampled set. Since the size of the sampled set is much less than that of the original data, one can use slightly involved algorithms as well, without much blow-up in complexity. We propose a simple soft-balanced variant of the k-means algorithm for this stage that computes the clusters by iterative relocation but softly takes into account the balancing constraints. A more rigorous balancing requirement can be fulfilled by a graph-based algorithm [16, 3].

Step 3 *Populating the clusters:* The clusters generated in the second step is to be populated with the data-points that were not sampled in the first step. For this step, we propose a less greedy approach motivated by the stable marriage problem [13] that takes into account the balancing constraints, followed by some re-adjustments. Note that this technique of populating the clusters can be applied irrespective of what clustering algorithm was used in the second step, as long as there is a way to represent the clusters.

This is a very general framework for scaling up balanced clustering algorithms, irrespective of the domain from which the data has been drawn and the clustering algorithm that is used as long as they satisfy the assumptions.

3 Algorithms and Analysis

For the basic analysis, we assume that the data-points are d -dimensional vectors and there exists a function $f : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ that gives the “distance” between two points in \mathbb{R}^d . In this section, we present and analyze the three steps of our methodology in detail.

3.1 Sampling

First, we examine the question - given the set \mathcal{X} having k underlying clusters of equal size, what is the number n of samples that need to be drawn from \mathcal{X} so that there are at least s points from each of the k clusters with high probability. Let X be a random variable for the number of samples to be drawn from \mathcal{X} to get at least s points from each cluster. $\mathbf{E}[X]$ can be computed by an analysis using the theory of recurrent events and renewals [10] - a simpler version of this analysis can be found in the so-called Coupon Collector's problem [19], or, equivalently in the computation of the cover time of a random walk on a complete graph. With a similar analysis in this case, we get

$$\mathbf{E}[X] \leq sk \ln k + O(sk) \quad (1)$$

Motivated by this result, we present the following theorem that shows that if we draw $n = csk \ln k \approx c\mathbf{E}[X]$ samples from \mathcal{X} , then we will get at least s samples from each cluster with high probability.

Theorem 1 *Given a set \mathcal{X} of N data-points having k underlying clusters of equal size, the probability of getting at least s samples from each cluster is more than $(1 - \frac{1}{k^a})$ if we draw $csk \ln k$ samples uniformly at random from \mathcal{X} , given $c \geq \frac{1}{\ln k}$ and $k^{c - \frac{a+1}{s}} \geq 4c \ln k$.*

This result can be proved using the union bound and Poisson approximations. The proof is not presented due to lack of space. To help understand the result, consider the case where $k = 10$ and we want at least $s = 50$ points from each of the clusters. Table 1 shows the total number of points that need to be sampled for different levels of confidence. Also note that if the k underlying point clusters are of equal size and

d	1	2	3	4	5
Confidence, $100(1 - \frac{1}{k^d})\%$	90.000	99.000	99.900	99.990	99.999
Number of Samples, n	1160	1200	1239	1277	1315

Table 1. *Number of samples required to achieve a given confidence level for $k=10$ and $s=50$*

$csk \ln k$ points are sampled uniformly at random, the expected number of points from each cluster is $cs \ln k$. Thus the underlying cluster structure is expected to be preserved in this smaller sampled set and the chances of wide variations from this behavior is very small. For example, for the 99.99% confidence level in Table 1, the average number of samples per cluster is 127, which is only 2.5 times the minimum cluster size we can get. The above analysis holds good when the size of all the underlying clusters in the full dataset are equal and given by $\frac{N}{k}$. In this perfectly balanced case, the size of the smallest clusters m is actually $\frac{N}{k}$. A similar analysis continues to hold good for m less than but of the same order of $\frac{N}{k}$. However, for $m \ll \frac{N}{k}$, more involved density-biased sampling [20] should be used for getting approximately equal number of samples from each of the clusters.

3.2 Balanced Clustering

This step involves clustering the set of n sampled points, \mathcal{X}_s , in a balanced fashion. Though it has been shown [17] that the well known k-means algorithm has some implicit balancing properties, it may give skewed clusters at times. We propose a small modification to the standard k-means algorithm that takes into account balancing as an explicit soft constraint. The suggested change is motivated by the ideas behind frequency sensitive competitive learning (FSCL) [21, 1], an online algorithm where the distance of a point from a particular cluster at any point during the execution of the algorithm is weighted by the number of points already assigned to that cluster. Thus, the effective distance to a cluster having more number of points increases, making the allocation of future points to such a cluster less likely. Convergence properties of FSCL has been studied in detail [11] and the results support the use of FSCL for applications such as clustering, design of radial basis functions, vector-quantization codebooks generation etc. In what follows, we introduce the proposed modification from a theoretical perspective and then give an algorithm `fsk-means` (frequency sensitive k-means) for performing the clustering that can be considered as a principled variant of FSCL.

First, we take a closer look at exactly what objective the k-means algorithm (locally) minimizes [18, 17]. Let us assume that the points have been generated from one of a mixture of k Gaussians with unknown parameters $\Theta = \{\theta_h : (\mu_h, \Sigma_h), h = 1, \dots, k\}$. Then, the probability of occurrence of a particular data-point x due to the h -th Gaussian is given by $p_h(x|\theta_h) = \Pr(x|N(\mu_h, \Sigma_h))$. The basic idea is to estimate the parameters, under certain assumptions, so that the modeling uncertainty of the data in terms of which Gaussian generated which of the data points is minimized. For k-means, it is assumed that $\Sigma_h = \mathbb{I}$, the identity matrix, and all the Gaussians have equal priors. Then, the clustering algorithm tries to minimize the modeling uncertainty in terms of the entropy of the data-points given by

$$\begin{aligned} H_{\Theta}(\mathcal{X}_s) &= E_{x \in \mathcal{X}_s}[-\log p_{C(x)}(x|\theta_{C(x)})] = E_{x \in \mathcal{X}_s}[\log \left((2\pi)^{d/2} e^{\frac{1}{2}\|x - \mu_{C(x)}\|^2} \right)] \\ &= c_1 E_{x \in \mathcal{X}_s}[\|x - \mu_{C(x)}\|^2] + c_2 \end{aligned} \quad (2)$$

where $C(x) \in \{1, \dots, k\}$ is the *cluster label* assigned to the data-point x and c_1, c_2 are constants. So, the objective function of k-means is $\mathcal{J} = \sum_{h=1}^k \sum_{x \in C_h} \|x - \mu_h\|^2$. The change we propose is to set $\Sigma_h = \frac{1}{n_h} \mathbb{I}$, where n_h is the number of points assigned to the h -th cluster at any point in the algorithm. So, the generating Gaussians are still isotropic but with diminishing width as more points are assigned to it. Hence, the likelihood of a point belonging to a cluster to which a large number of points have already been assigned is less in this setup than to another cluster whose mean is at the same Euclidean distance but has less number of points. This frequency sensitive stretching of the mixture components result in clusters that are more balanced.¹ Next, we present the objective function for this setup. Following Eqn. 2

¹unless the inherent structure of the data is wildly skewed.

and incorporating the modification of the covariance, we get

$$\begin{aligned} H_{\Theta}(\mathcal{X}_s) &= E_{x \in \mathcal{X}_s} [\log((2\pi)^{d/2} n_{C(x)}^{-1/2} \exp(\frac{1}{2} n_{C(x)} \|x - \mu_{C(x)}\|^2))] \\ &= c_1 E_{x \in \mathcal{X}_s} [n_{C(x)} \|x - \mu_{C(x)}\|^2 - \log(n_{C(x)})] + c_2 \end{aligned} \quad (3)$$

where c_1, c_2 are constants. So, the objective function for the frequency sensitive k-means is given by

$$\mathcal{J} = \sum_{h=1}^k \sum_{x \in C_h} (n_h \|x - \mu_h\|^2 - \log(n_h)) \quad (4)$$

Finally, we present the algorithm `fsk-means` that attempts to minimize the above objective using the standard iterative relocation approach.

Algorithm `fsk-means`

Input: A set \mathcal{X}_s of n data-points and the number of clusters k .

Output: A disjoint partitioning of \mathcal{X}_s into k subsets $C_h, h = 1, \dots, k$ that attempts to minimize the objective in Eqn. 4.

Method: 1. Set iteration count $t \leftarrow 0$. Choose k points (not necessarily in S) as the cluster means $\mu_h^{(0)}$ and set $n_h^{(0)} \leftarrow \frac{n}{k}, h = 1, \dots, k$.

2. Repeat until *convergence*

2a. Assign each data-point x to the cluster $C_{h^*}^{(t)}$ where

$$h^* = \arg \min_h (n_h \|x - \mu_h\|^2 - \log(n_h))$$

2b. Assign $n_h^{(t+1)} \leftarrow |C_h^{(t)}|$.

2c. Assign $\mu_h^{(t+1)} = \frac{1}{n_h^{(t+1)}} \sum_{x \in C_h^{(t)}} x$.

2d. Set $t \leftarrow (t + 1)$.

It is to be noted that the above method provably gives the local minima of the objective in Eqn. 4 under very strong assumptions. However, as we shall see, this method performs extremely well in practice and also gives a good optimization of the k-means objective. A plausible reason for this is that the balancing effort prevents it from converging to a bad local minima and it behaves very similar to traditional simulated annealing algorithms. Moreover, it shares FSCL's property of being less sensitive to initial conditions [1] as compared to k-means.

3.3 Populating the Clusters

After clustering an n point sample from the original data-set \mathcal{X} , the remaining $(N - n)$ points need to be assigned to the clusters, satisfying the balancing constraint. In this section, an algorithm for doing this is presented.

Let $\mu_1, \mu_2, \dots, \mu_k$ be the means of the k clusters C_1, C_2, \dots, C_k and let n_h be the number of points in cluster C_h after the completion of `fsk-means`. Clearly $\sum_{h=1}^k n_h = n$. Let v_1, v_2, \dots, v_{N-n} be the $(N - n)$ points that were not sampled and hence still need to be assigned to the clusters. Now, if the assignment of points is to be done under perfect balancing constraints, a set of exactly $(\frac{N}{k} - n_h)$ points

needs to be assigned to cluster C_h so that each cluster finally has exactly $\frac{N}{k}$ points. A more practical scenario is to ensure that each cluster has at least $m = b \cdot \frac{N}{k}$ points, where $0 < b \leq 1$ is the *balancing fraction*. In this case, at least $(b \cdot \frac{N}{k} - n_h)$ points need to be assigned to cluster C_h . We shall present the algorithm for this practical fractional balancing case. In fact, the exact balancing constraint is a special case of the fractional balancing when the balancing fraction $b = 1$.

Ideally, each point should be assigned to the nearest mean so that $\forall v_i$, $f(v_i, \mu_{C(v_i)}) \leq f(v_i, \mu_h), \forall h \neq C(v_i)$ where $C(v_i) \in \{1, 2, \dots, k\}$ represents the cluster to which v_i is assigned. However, this need not satisfy the balancing constraints. If $S_h = \{v_i | C(v_i) = h\}$ be the set of points that were assigned to C_h , then balancing requires $|S_h| \geq (b \cdot \frac{N}{k} - n_h)$. Our scheme for populating the clusters has two parts. First, each cluster is given the best possible $(b \cdot \frac{N}{k} - n_h)$ points from the entire collection of $(N - n)$ points; after the completion of this part, $\sum_{h=1}^k (b \cdot \frac{N}{k} - n_h) = bN - n$ points get assigned to clusters. Then, in the second part, the remaining $(N - (bN - n) - n) = N(1 - b)$ points are assigned to their nearest clusters greedily followed by some re-adjustments, satisfying the balancing constraints all along.

The first part gives the following guarantee after completion: $|S_h| = (b \cdot \frac{N}{k} - n_h)$, and $\forall v_i$ that has been assigned to a cluster, either $f(v_i, \mu_{C(v_i)}) \leq f(v_i, \mu_h), \forall h \neq C(v_i)$, or, if $\exists h' f(v_i, \mu_{C(v_i)}) > f(v_i, \mu_{h'})$, then $\forall v_{i'} \in S_{h'}, f(v_{i'}, \mu_{h'}) \leq f(v_i, \mu_{h'})$. Any assignment that satisfies the above conditions is said to be a stable assignment. Thus, if a point cannot get into a cluster whose mean is nearer to it than the mean of its own cluster, then that other cluster must have already got its required $(b \cdot \frac{N}{k} - n_h)$ points, each of which is nearer to that mean than the point under consideration. The algorithm we present is motivated by the proposal algorithm for the stable marriage problem [13, 19]. It is in fact a polygamous version of the classical stable marriage problem [2] and will be called the `poly-stable` algorithm.

Polygamous Stable Marriage

In this subsection, we give an outline of the `poly-stable` algorithm. First, the distance $f(v_i, \mu_h)$ between every unassigned point $v_i, i = 1, \dots, (N - n)$ and every cluster $C_h, h = 1, \dots, k$ is computed. For each cluster C_h , all the $(N - n)$ points are sorted in increasing order of their distance with μ_h . Let the ordered list of points for cluster C_h be denoted by $\Pi^{(h)}$. Let m_h denote the number of points yet to be assigned to cluster C_h at any stage of the algorithm. Initially $m_h = (b \cdot \frac{N}{k} - n_h)$. This part of the algorithm will terminate when $m_h = 0, \forall h$. The basic idea behind the algorithm is “cluster proposes, point disposes”. In the first step, each of the k clusters proposes to its nearest $(b \cdot \frac{N}{k} - n_h)$ points. Every point which has been proposed, gets temporarily assigned to the nearest among its proposing clusters and rejects the proposal of all its other proposing clusters. Note that if a point has received only one proposal, it gets temporarily assigned to the only cluster which proposed it and there is no rejection involved. For each cluster m_h is computed. Next, the following process is repeated for $h = 1 \dots k$ until $m_h = 0, \forall h$. If $m_h \neq 0$, cluster C_h proposes the next m_h points from its list $\Pi^{(h)}$ that have not already

rejected its proposal. Each of the proposed points accepts the proposal either if it is currently unassigned or if the proposing cluster is nearer than the cluster $C_{h'}$ to which it is currently assigned. In the later case, the point rejects its old cluster $C_{h'}$ and the cluster $C_{h'}$ loses a point so that $m_{h'}$ goes up by 1. This process is repeated until $m_h = 0, \forall h$ and the algorithm terminates.

Now we show that this algorithm indeed satisfies all the required conditions and hence ends up in a stable assignment. Before giving the actual proof, we take a closer look at what exactly happens when an assignment is unstable. Let $(v_i \rightarrow C_h)$ denote the fact that the point v_i has been assigned to the cluster C_h . An assignment is unstable if there exist at least two assignments $(v_{i_1} \rightarrow C_{h_1})$ and $(v_{i_2} \rightarrow C_{h_2})$, $h_1 \neq h_2$, such that v_{i_1} is nearer to C_{h_2} than C_{h_1} and C_{h_2} is nearer to v_{h_1} than v_{h_2} . The point-cluster pair (v_{i_1}, C_{h_2}) is said to be dissatisfied under such an assignment. An assignment in which there are no dissatisfied point-cluster pairs is a stable assignment and satisfies the constraints. Next, we show that there will no dissatisfied point-cluster pair after the algorithm terminates.

Claim 1 *poly-stable gives a stable assignment.*

Proof. If possible, let $(v_{i_1} \rightarrow C_{h_1})$ and $(v_{i_2} \rightarrow C_{h_2})$ be two assignments such that (v_{i_1}, C_{h_2}) is a dissatisfied point-cluster pair. In this case, since v_{i_1} is nearer to C_{h_2} than v_{i_2} , C_{h_2} must have proposed v_{i_1} before v_{i_2} . Since v_{i_1} is not assigned to C_{h_2} , v_{i_1} must have either rejected the proposal of C_{h_2} meaning it was currently assigned to a cluster which was nearer than C_{h_2} , or accepted the proposal initially only to reject it later meaning it got a proposal from a cluster nearer than C_{h_2} . Since a point only improves its assignments over time, the cluster C_{h_1} to which v_{i_1} is finally assigned must be nearer to it than the cluster C_{h_2} which it rejected. Hence v_{i_1} cannot be dissatisfied which contradicts the initial assumption. So, the final assignment is indeed stable. ■

Next we look at the total number of proposals that are made before the algorithm terminates. After a cluster proposes a point for the first time, there are three possible ways this particular point-cluster pair can behave during the rest of the algorithm: (i) the point immediately rejects the proposal in which case the cluster never proposes it again; (ii) the point accepts it temporarily and rejects it later - the cluster obviously does not propose it during the acceptance period and never proposes it after the rejection; (iii) the point accepts the proposal and stays in that cluster till the algorithm terminates - obviously the cluster does not propose it again during this time. Hence, each cluster proposes each point at most once and so the maximum number of proposals possible is $k \times (bN - n)$. So, the complexity from the proposal part is $O(k(bN - n))$. Before starting the proposals, the sorted lists of distances of all the points from each of the clusters have to be computed, which has a complexity of $O(k(N - n) \log(N - n))$. So, the total complexity of this part is $O(k((N - n)(\log(N - n) + b) - n(1 - b)))$.

Free Point Assignment

The remaining $N(1 - b)$ still unassigned points can now be greedily allocated to their nearest means. However, a closer look at the assignment process tells us that a slight re-adjustment of points after the greedy assignment actually gives a better clustering. After the stable assignment, each cluster C_h will have a set U_h of unhappy points each of which was nearer to some other cluster but got stuck in C_h in order to satisfy the balancing constraints. From the $v \in U_h$, a sorted list L_h in increasing order according to $(f(v, \mu_{h'}) - f(v, \mu_h))$, where $h' = \arg \min_g f(v, \mu_g)$, is generated. Clearly $h' \neq h$, $f(v, \mu_{h'}) < f(v, \mu_h)$, and $C_{h'}$ is the cluster v wants to be in. Now the $N(1 - b)$ still unassigned points are greedily assigned to their nearest clusters. We compute $|S_h|, \forall h$ and note that $(|S_h| - b \cdot \frac{N}{k} + n_h)$ points from U_h can be re-assigned to their desired clusters without violating the balancing constraint. This results in a much less number of unhappy points. Next, we go one step further and allow the re-adjustments to ripple - in the sense that the re-assigned points are also allowed to release some unhappy points from their destination cluster until no further points can be re-assigned. This is similar in flavor to the random heuristic for the pivot movement problem in [23]. Following this idea, one can recompute the means and carry on the re-adjustment rippling while always satisfying the balancing constraints (see [23]). This iterative re-adjustment rippling is not done in the current work and experimental results are presented based only on the re-adjustments rippling due to the newly assigned points.

3.4 Overall Complexity

Finally, we present a brief discussion on the complexity of the above mentioned scheme. First, the sampling of the n points has a complexity of $O(n)$. The `fsk-means` algorithm has a complexity of $O(tkn)$ where t is the number of iterations. As shown earlier, the `poly-stable` algorithm has a complexity of $O(k((N - n)(\log(N - n) + b) - n(1 - b)))$. The number of re-adjustments due to rippling can at most be equal to the number of unhappy points that is upper-bounded by $(bN - n)$. Note that the sorting of the unhappy points for re-adjustments really does not need a separate sorting operation since the sorting results obtained before running `poly-stable` can be used along with some selection operations. Hence, the free point assignment part has a complexity of $O(bN - n)$. So, the total complexity of the scheme is $O(n + tkn + k((N - n)(\log(N - n) + b) - n(1 - b)) + (bN - n)) = O(kN \log N)$. Note that this is better than the $O(N^2)$ complexity of graph-based algorithms that can provide some balancing. It is worse than the $O(kN)$ complexity of the iterative greedy relocation algorithms such as `k-means`, but such algorithms cannot guarantee the satisfaction of the balancing constraints.

3.5 Database Considerations

In this subsection, we briefly discuss out-of-core related computational issues that are important when the data is stored in a database residing in secondary memory. First, the sampling of n points uniformly at random is an expensive operation

in such situations. However, one can perform cluster sampling, wherein uniform sampling is done only within each page. Since the pages are accessed sequentially from disk and not revisited, the sample of size n is obtained over a single database scan. For step 2, there is no problem since even for very high confidence levels, the values of n obtained from Theorem 1 are very small as compared to the typical main memory size. Note that `fsk-means` requires only $O(k + n)$ storage. In fact, even if we needed n^2 storage for this step in order to store the similarity matrix for a more involved graph-based balanced clustering, there should be no problem. The third step may force out-of-core operations if N is so large that it is not possible to retain the distances of each of the unassigned data points to each of the cluster centers in main memory. However, one can simply load the remaining data in batches of size M at a time into main memory (where M is small enough to be accommodated into main memory), allocating each batch to the clusters in a balanced way before loading the next batch. Note that this still needs only one database scan, and the computation requirements is reduced to $O(kN \log M)$. The compromise in quality is expected to be small, but needs to be quantified via extensive simulation. One can of course instead do a greedy assignment of the remaining points to their nearest clusters, reflecting a further computation-quality tradeoff.

4 Experimental Results

In this section, experimental results on the proposed ideas are presented. First we present results on two high-dimensional text datasets. We used the 20-newsgroups dataset² and the Yahoo news dataset³(K1) for the empirical performance analysis. The 20-newsgroup dataset is a collection of 20,000 messages, collected from 20 different usenet newsgroups. One thousand messages from each of the twenty newsgroups were chosen at random and partitioned by newsgroup name. The headers for each of the messages were chopped off so that they do not bias the results. The toolkit MC [7] was used for creating the vector space model for the text documents and a total of 26099 words were used. Thus, each message is represented as a (sparse) vector in a 26099 dimensional space. The Yahoo news K-series dataset is a collection of 2340 Yahoo news articles belonging one of 20 different Yahoo categories. The K1 set actually gives the vector space model having 21839 words and hence the data-points reside in a 21839 dimensional space.

First, we take a look at the performance of the `fsk-means` algorithm as compared to the standard `k-means`, without any sampling. For a particular value of number of clusters k , we run both the algorithms on a data-set. Note that spherical versions of both the algorithms were used since the spherical version tends to perform much better for large document collections [7]. In the spherical versions, each document is normalized to a unit vector and so are the representative means after each iteration. The objective functions were modified appropriately. The spherical

²<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html>

³<ftp://ftp.cs.umn.edu/users/boley/PDDPdata/>

k-means objective [7] is given by

$$\mathcal{J}_{\text{k-means}} = \frac{1}{N} \sum_{h=1}^k \sum_{x \in C_h} x^T \mu_h, \quad (5)$$

and represents the average cosine similarity of every document with its cluster representative. The cosine of the angle between two document vectors is a popular measure of similarity between text documents in the IR community. For **fsk-means**, the spherical objective is given by

$$\mathcal{J}_{\text{fsk-means}} = \frac{1}{N} \sum_{h=1}^k \sum_{x \in C_h} (\eta_h x^T \mu_h + \log \eta_h) \quad (6)$$

where $\eta_h = \frac{n/k}{n_h}$. Effectively, the scaled up distances $n_h \|x - \mu_h\|^2$ have been replaced by scaled down similarities of the form $\frac{1}{n_h} x^T \mu_h$ with some appropriate normalizations. Since the spherical versions maximize similarity, as opposed to minimizing distances, the aim to maximize the objective function and the plots are to be read accordingly. The initial k means for both the algorithms were generated by computing the mean of the entire data and making k small random perturbations to this mean [7]. In Fig. 1(a) the objective function value given by Eqn. 5 for both the algorithms [after the convergence on their individual objectives] on the 20-newsgroups data is presented. The results are averaged over 10 runs on each value of k . It is interesting to note that both the approaches give approximately the same value for the objective $\mathcal{J}_{\text{k-means}}$ upto around $k \approx 15$ after which **fsk-means** performs significantly better than normal k-means. Similar results on the Yahoo dataset is presented in Fig. 1(b). In Fig. 1(c), the variance of the number of documents per cluster on the 20-newsgroups data is presented for both the algorithms. Again, for lower values of k , there is not much difference between their performance; as k becomes larger, **fsk-means** performs better than k-means in the sense that there is less variation in the cluster sizes. Similar results on the Yahoo dataset are presented in Fig. 1(d). As a special and interesting case, we compare both the algorithms based on the smallest sized cluster they generate in different runs [Fig. 1(e)] on the 20-newsgroups data. The results are plotted as a ratio of the smallest cluster size to the expected cluster size. Note that the minimum cluster size that is plotted in Fig. 1(e) is the mean of the smallest cluster sizes over the 10 runs. k-means starts misbehaving again from around $k \approx 15$ and for $k > 25$ it seems to always generate some empty clusters. This is a well known property of the k-means algorithm, specially in very high dimensional spaces. The **fsk-means** algorithm does not suffer from this due to its frequency sensitivity. Again, similar results on the Yahoo dataset are presented in Fig. 1(f). Thus, **fsk-means** gives better clusters as well as better balanced clusters than the normal k-means even when evaluated using the latter's objective [Eqn. 5]. Having said that, we repeat that any good soft-balanced clustering algorithm [16, 22, 3] should suffice for this stage of the methodology. Note that **fsk-means** seems to perform better for the 20-newsgroups data. The primary reason for this is that the natural clusters in the Yahoo dataset does not satisfy

the balancing assumptions and the cluster sizes actually range from 9 to 494. Even with this amount of skew, `fsk-means` gives comparable and at times significantly better results than `k-means`.

Next, we present results of running the whole scheme with sampling on the entire dataset. All these sampling experiments that we present have been performed on the 20-newsgroups data just because it has a larger number of data-points. In Fig. 2, we compare the performance of three different versions of our scheme on the 20-newsgroups data. The versions differ in the way the clusters are populated after running `fsk-means` on the sampled data. The first version, called `greedy fsk-means` or `gfsk-means`, populates the cluster greedily so that each un-sampled point goes to its nearest cluster. Clearly, this version cannot respect the balancing constraints. The second version, called `normal fsk-means` or `nfsk-means`, first runs `poly-stable` for satisfying the balancing constraints followed by greedy assignment of the remaining $N(1 - b)$ points. The third version, called `rippling fsk-means` or `rfsk-means`, runs similar to `nfsk-means` upto the greedy assignment part after which a rippling of the re-adjustments is done as described in subsection 3.3. All results are presented for $k = 20$ clusters. Fig. 2 shows the objective function values obtained for (a) 1000, (b) 2000, (c) 5000, and (d) 10000 initial samples. For lower values of the balancing fraction b , all the three schemes perform similarly. For larger values of b a difference starts to show up. Since the performance `gfsk-means` does not depend on b at all, it achieves similar objective function values for the entire range of b . However, performance of the other two schemes starts degrading for higher values of b since they have to satisfy the balancing constraints. There are three things that needs to be noted. First, the performance does not degrade much if the number of points that were initially sampled is large - for example, all the three schemes perform comparably when initially 10000 points were sampled [Fig 2(d)]. Secondly, `rfsk-means` performs better than `nfsk-means`. The reason for this is intuitively clear - since `rfsk-means` reduces the number of unhappy points significantly by re-adjustment rippling, the clusters are much more compact for this scheme, resulting in a better value of the objective. Lastly, the objective values achieved by our scheme for number of samples $n \geq 2000$ and moderate values of b is comparable to the performance of both `k-means` and `fsk-means` run on the whole data (see objective for $k=20$, Fig. 1(a)). Figs. 2(e) and 2(f) show the standard deviation of the cluster sizes for 2000 and 5000 initial number of samples respectively. This shows that since `gfsk-means` cannot respect the balancing constraints, the cluster sizes produced by this scheme are widely skewed. The other two schemes have steadily decreasing cluster size variation that becomes zero when the balancing fraction is one (perfect balancing). Note that the variation of cluster sizes for `rfsk-means` is more than that of `nfsk-means` even though both of them satisfy the balancing requirements. This is again due to the re-adjustment rippling done by `rfsk-means` that results in a better objective value. To summarize, the `rfsk-means` scheme gives a pretty good balance between objective optimization and constraint satisfaction over a wide range of balancing fractions and initial sample values.

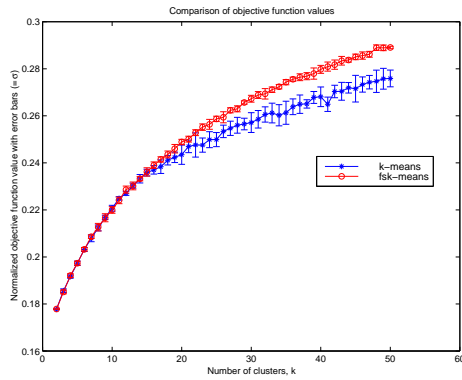
To experiment with artificial data, for a given dimension d , and number of clusters k , we generated k centroids uniformly at random from $[-0.5, 0.5]^d$. Cen-

troids that were very close to other centroids were rejected and replaced by new ones, till cluster centers had some minimum separation. Then, around each centroid, N/k points were obtained from a zero-mean Gaussian with covariance $s\delta \times \mathbb{I}_d$, where \mathbb{I}_d is the d -dimensional identity matrix, δ is the Euclidean distance to the nearest centroid, and s is a fraction that controls the amount of overlap between the nearest clusters. Note that such data should be very favorable to (regular) k-means since the generating function follows assumptions behind the k-means objective function. When clusters were very well separated (s small), our method produced virtually identical results on cluster quality, and occasionally better results on balance, even for fairly drastic sampling rates. Essentially such problems are easy, and results bootstrapped from a small data sample are hardly different from those obtained from the full data. For higher levels of overlap, our methods shows superior balancing properties than k-means. Result plots are not shown for lack of space.

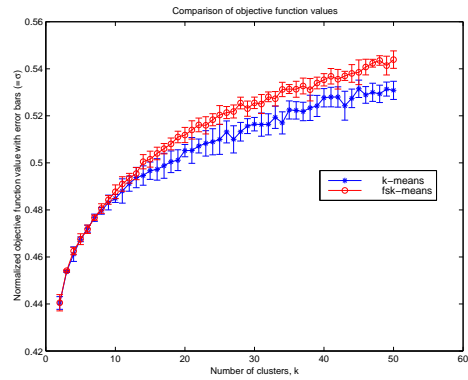
5 Concluding Remarks

The proposed framework for scaling up balanced clustering algorithms is fairly broad in that it accommodates a wide range of balanced clustering algorithms for grouping the sampled data, as well as a range of increasingly sophisticated but more computationally demanding for the final step of populating the initial clusters. Moreover, the guarantees provided by sampling, as stated in Theorem 1 and exemplified in Table 1, are independent of the size of the original population, so the computational demands for the first and second stages are not affected by the size of the database. In terms of results, `fsk-means` performs better than k-means, for both regular and spherical versions, even when the algorithms are evaluated using the k-means cost function. In addition, it provides better balancing and less sensitivity to initial conditions. More interestingly, for the Yahoo! data, where the natural clusters are quite imbalanced, `fsk-means` is still able to provide a better result even though it is forcing a balancing condition on the data. For the overall scheme, at the end of step 3 we obtain fairly balanced clusters with little extra effort, and the method seems quite robust.

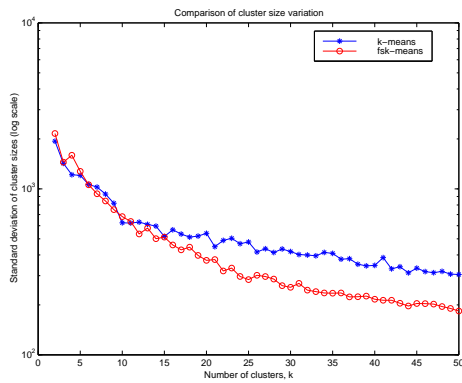
A clustering algorithm that works best for all types of data is yet to be found, and it will be extremely naive to expect that our method is the exception. Experimentation on a wider variety of data sets need to be done to identify weaknesses. In particular, if the natural clusters are highly imbalanced, the method may break. This may be alleviated by applying more involved density-biased sampling techniques [20] during step 1, so that the assumptions and requirements on step 2 remain unchanged. More sophisticated graph-based balanced partitioning algorithms may be used for step 2 [22]. Since this step applies only to the sampled set, the complexity of this stage really does not show up as a big factor in the overall complexity. In contrast, for the final step, more involved techniques such as linear assignment are not attractive, since we want this step to be near linearly scalable. We are currently experimenting on a larger variety of artificial and real data sets to get a better understanding of the performance-runtime trade-offs in this step.



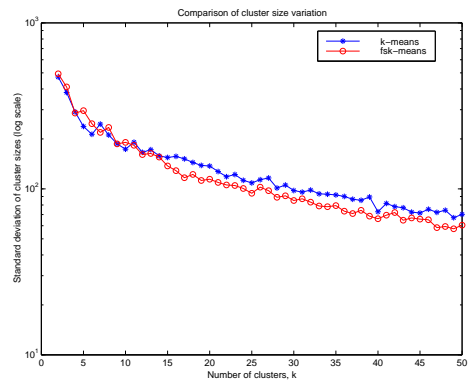
(a)



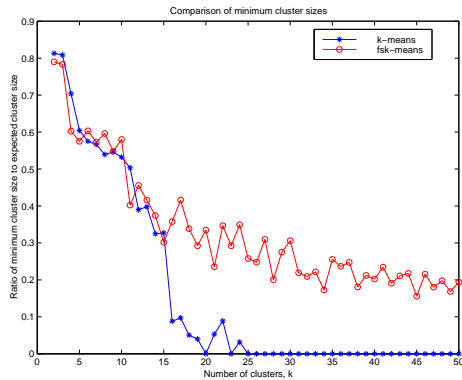
(b)



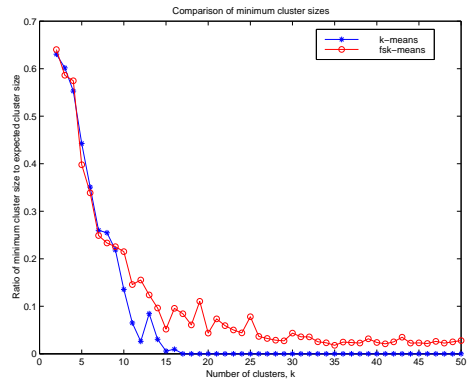
(c)



(d)

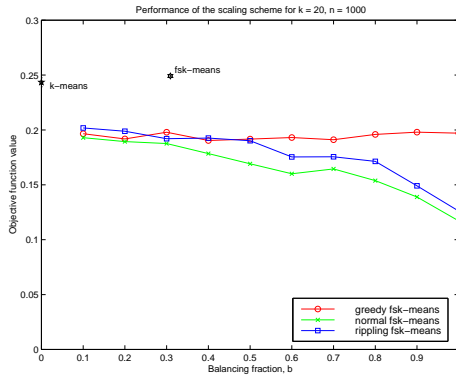


(e)

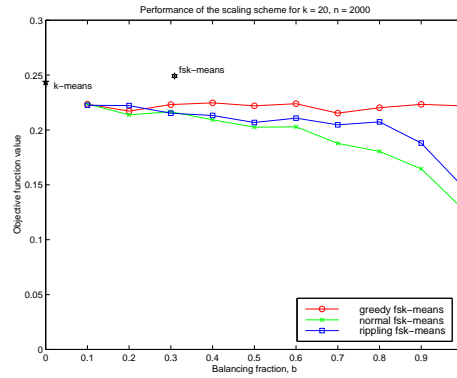


(f)

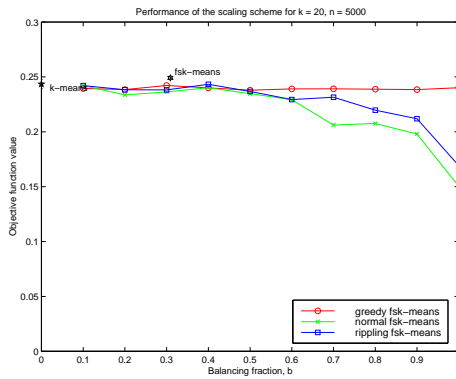
Figure 1. Comparison of k -means and fsk-means (without sampling): (a) the normalized spherical k -means objective function values, (c) the variance in cluster sizes, and (e) the minimum cluster size for the 20-newsgroups data, computed over 10 runs of each algorithm. The corresponding plots on the Yahoo news(K1) dataset are in (b), (d), (f) respectively.



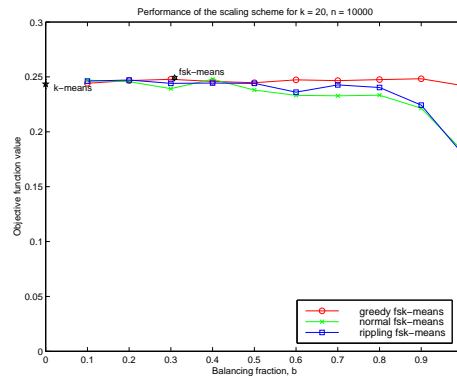
(a)



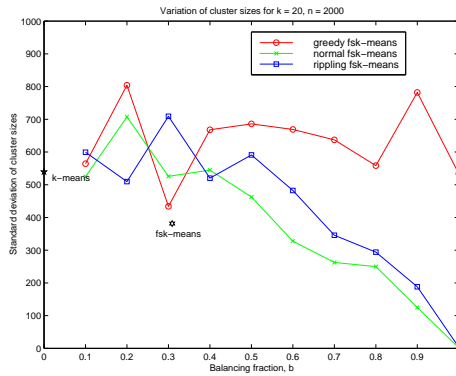
(b)



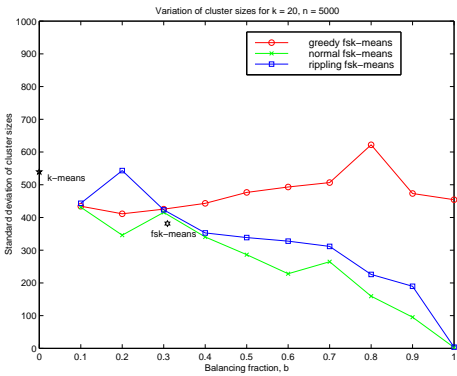
(c)



(d)



(e)



(f)

Figure 2. Comparisons of the variants on fsk-means in terms of objective function values for $k = 20$, and $n =$ (a) 1000, (b) 2000, (c) 5000, and (d) 10000, over a range of values of b . Comparisons of the variations in cluster sizes for $n =$ (e) 2000, and (f) 5000, are also shown. Corresponding results of k -means and fsk-means on the full data are also marked on the plots.

Bibliography

- [1] S. C. AHALT, A. K. KRISHNAMURTHY, P. CHEN, AND D. E. MELTON, *Competitive learning algorithms for vector quantization*, Neural Networks, 3 (1990), pp. 277–290.
- [2] M. BAIYOU AND M. BALINSKI, *Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry)*, Discrete Applied Mathematics, 101 (2000), pp. 1–12.
- [3] P. S. BRADLEY, K. P. BENNETT, AND A. DEMIRIZ, *Constrained k-means clustering*, tech. rep., Microsoft Research, May 2000.
- [4] P. S. BRADLEY, U. M. FAYYAD, AND C. REINA, *Scaling clustering algorithms to large databases*, in Knowledge Discovery and Data Mining, 1998, pp. 9–15.
- [5] ———, *Scaling EM (expectation-maximization) clustering to large databases*, tech. rep., Microsoft Research, 1998.
- [6] D. R. CUTTING, D. R. KARGER, J. O. PEDERSEN, AND J. W. TUKEY, *Scatter/gather: A cluster-based approach to browsing large document collections*, in Proc. 15th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1992, pp. 318–329.
- [7] I. S. DHILLON, J. FAN, AND Y. GUAN, *Efficient clustering of very large document collections*, in Data Mining for Scientific and Engineering Applications, V. K. R. Grossman, C. Kamath and R. Namburu, eds., Kluwer Academic Publishers, 2001.
- [8] P. DOMINGOS AND G. HULTON, *A general method for scaling up machine learning algorithms and its application to clustering*, in Proc. 18th Intl. Conf. Machine Learning, 2001, pp. 106–113.
- [9] D. FASULO, *An analysis of recent work on clustering*, tech. rep., University of Washington, Seattle, 1999.
- [10] W. FELLER, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, 1967.

- [11] A. S. GALANOPOULOS, R. L. MOSES, AND S. C. AHALT, *Diffusion approximation of frequency sensitive competitive learning*, IEEE Trans. Neural Networks, 8 (1997), pp. 1026–1030.
- [12] S. GUHA, R. RASTOGI, AND K. SHIM, *Cure: An efficient clustering algorithm for large databases*, in Proc. ACM SIGMOD Intl. Conf. on Management of Data, New York, 1998, ACM, pp. 73–84.
- [13] D. GUSFIELD AND R. W. IRVING, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Cambridge, MA, 1989.
- [14] J. HAN, M. KAMBER, AND A. K. H. TUNG, *Spatial clustering methods in data mining: A survey*, in Geographic Data Mining and Knowledge Discovery, Taylor and Francis, 2001.
- [15] A. K. JAIN, M. N. MURTY, AND P. J. FLYNN, *Data clustering: a review*, ACM Computing Surveys, 31 (1999), pp. 264–323.
- [16] G. KARYPIS AND V. KUMAR, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on Scientific Computing, 20 (1998), pp. 359–392.
- [17] M. KEARNS, Y. MANSOUR, AND A. NG, *An information-theoretic analysis of hard and soft assignment methods for clustering*, in 13th Annual Conf. on Uncertainty in Artificial Intelligence (UAI97), 1997.
- [18] T. M. MITCHELL, *Machine Learning*, McGraw-Hill Intl, 1997.
- [19] R. MOTWANI AND P. RAGHAVAN, *Randomized Algorithms*, Cambridge University Press, 1995.
- [20] C. R. PALMER AND C. FALOUTSOS, *Density biased sampling: An improved method for data mining and clustering*, tech. rep., Carnegie Mellon University, May 1999.
- [21] P. SCHEUNDERS AND S. D. BACKER, *High-dimensional clustering using frequency sensitive competitive learning*, Pattern Recognition, 32 (1999), pp. 193–202.
- [22] A. STREHL AND J. GHOSH, *A scalable approach to balanced, high-dimensional clustering of market-baskets*, in Proc 17th Intl Conf on High Performance Computing (HiPC 2000), Springer LNCS, December 2000, pp. 525–536.
- [23] A. K. H. TUNG, R. T. NG, L. V. S. LAKSMANAN, AND J. HAN, *Constraint-based clustering in large databses*, in Proc. Intl. Conf. on Database Theory (ICDT'01), Jan 2001.
- [24] T. ZHANG, R. RAMAKRISHNAN, AND M. LIVNY, *BIRCH: an efficient data clustering method for very large databases*, in Proc. ACM SIGMOD International Conference on Management of Data, 1996, pp. 103–114.