Approximation Schemes for Clustering Problems (Extended Abstract)*

W. Fernandez de la Vega † Marek Karpinski ‡ Claire Kenyon§
Yuval Rabani¶

November 5, 2002

Abstract

Let k be a fixed integer. We give polynomial time approximation schemes (PTASs) for the following three problems: metric k-clustering, i.e., partitioning an input set of n points into k clusters so as to minimize the sum of all intra-cluster distances, for an arbitrary metric space; l_2^2 k-clustering, when the points are in $\mathbb{R}^d$ and the "distance" between two points x, y is measured by $\|x-y\|_2^2$ (notice that $(\mathbb{R}^d, \|\cdot\|_2^2)$ is not a metric space); l_2^2 k-Median, which differs from the l_2^2 k-clustering problem in the definition of the objective function, which is now to minimize the sum of distances from points in a cluster to the (best choice of) cluster center. For the first two problems, these are the first PTASs. For the third problem, our running time is a vast improvement over previous work.

1 Introduction

Problem statement and motivation. The problem of partitioning a data set into a small number of *clusters* of related items has a crucial role in many information retrieval and data analysis applications, such as web search and classification [5, 7, 24, 11], or interpretation of experimental data in molecular biology [23].

*The full version of this paper can be found under the following URL: `http://theory.cs.uni-bonn.de/cs-reports-2002`

†Email:`lalo@lri.lri.fr` LRI, CNRS UMR 8623, Université Paris-Sud, France.

‡Email: `marek@cs.uni-bonn.de`, Dept. of Computer Science, University of Bonn. Research partially supported by DFG grants, PROCOPE grant 31022, and IST grant 14036 (RAND-APX).

§Email:`kenyon@lix.polytechnique.fr`. LIX, CNRS UMR 7650, Ecole Polytechnique, France.

¶Computer Science Department, Technion — IIT, Haifa 32000, Israel. Work at the Technion supported by Israel Science Foundation grant number 386/99, by US-Israel Binational Science Foundation grant number 99-00217, by the European Commission Fifth Framework Programme Thematic Networks contract number IST-2001-32007 (APPOL II), and by the Fund for the Promotion of Research at the Technion. Email: `rabani@cs.technion.ac.il`

We consider a set V of n points endowed with a distance function δ. These points have to be partitioned into a fixed number k of subsets $C_1, C_2, \ldots, C_k$ so as to minimize the cost of the partition, which is defined to be the sum over all clusters of the sum of pairwise distances in a cluster. We call this problem k-Clustering. We also deal with the k-Median and the k-Center problems. In the k-Median problem the cost of a clustering is the sum over all clusters of the sum of distances between cluster points and the best choice for a cluster center. In the k-Center problem, the cost of a clustering is the maximum distance between a point and its cluster center. In the settings that we consider, these optimization problems are NP-hard (by similar arguments as in [9, 8][1]) to solve exactly even for $k = 2$.

Our results. Our algorithms deal with the case that δ is an arbitrary metric. We also handle the non-metric case of "ℓ_2^2 instances", i.e. points in $\mathbb{R}^d$ where the distance between two points x, y is measured by $\delta(x, y) = \|x - y\|_2^2$.

For the metric and for the ℓ_2^2 k-Clustering problem, we present algorithms for every fixed integer k and for every fixed $\epsilon > 0$ that compute a partition into k clusters of cost at most $1 + \epsilon$ times the cost of an optimum partition. Although we do not discuss it in this extended abstract, our algorithms can be modified to handle variants which exclude outliers.

The k-Median problem can be solved optimally in polynomial time for fixed k in finite metrics, because the number of choices for centers is polynomial. However, if the points are located in a larger space, such as $\mathbb{R}^d$, and the centers can be picked from this larger space, the problem may become hard. For ℓ_2^2 instances, we give a randomized algorithm that partitions the input point-set into k clusters of cost at most $1 + \epsilon$ of the optimum cost in probabilistic time $O(g(k, \epsilon)n(\log n)^k)$. Although we do not discuss it in this extended abstract, it can easily be modified to derive polynomial time approximation schemes for other objective functions, such as the k-Center problem.

Related work. The k-Clustering problem was proposed by Sahni and Gonzalez [21] in the setting of arbitrary weighted graphs. Unfortunately, only poor approximation guarantees are possible [17, 12]. Guttman-Beck and Hassin [15] initiated the study of the problem in metrics. Indyk [16] designed a polynomial time approximation scheme for Metric 2-Clustering. Thus our metric results extend Indyk's result to the case of arbitrary fixed k.

Schulman [22] gave probabilistic algorithms for clustering ℓ_2^2 instances. His algorithms find a clustering such that either its cost is within a factor of $1 + \epsilon$ of the optimum cost, or it can be converted into an optimum clustering by changing the assignment of at most an ϵ fraction of the points.

For arbitrary k, the k-Median problem is APX-hard [14]. This is not the case in geometric settings, including the ℓ_2^2 case discussed in this paper. This case was considered by Drineas, Frieze, Kannan, Vempala, and Vinay [10], who gave a 2-approximation algorithm. Ostrovsky and Rabani [20] gave a polynomial time approximation scheme for this case. Bădoiu, Har-Peled, and Indyk [4] gave a polynomial time approximation scheme for points in Euclidean space with much improved running time. Our results, derived independently of [4] but with a similar algorithm, improve significantly the running time for the ℓ_2^2 case.

[1] which is how the authors of [8] got interested in this problem

Notations. The function δ can be given explicitly or implicitly (for example, if $V \subset \mathbb{R}^d$ and δ is derived from a norm on $\mathbb{R}^d$). Our time bounds count arithmetic operations and assume that computing $\delta(x, y)$ is a single operation. The reader may assume that the input is rational to avoid having to deal with unrealistic computational models. Without loss of generality, we omit the ceiling notation from expressions such as $\lceil 1/\epsilon \rceil$.

Let $X, Y \subset V$ and $x \in V$. With a slight abuse of notation, we use $\delta(x, Y)$ to denote $\sum_{y \in Y} \delta(x, y)$, and we use $\delta(X, Y)$ to denote $\sum_{x \in X} \delta(x, Y)$. Notice that $\delta(\cdot, \cdot)$ is a symmetric bilinear form. We use $\delta(X)$ to denote $\delta(X, X)$.

Let $C_1, C_2, \ldots, C_k$ be a partition of V into k disjoint clusters. We use $\mathrm{cost}(C_i)$ to denote the cost of C_i: for k-Clustering, $\mathrm{cost}(C_i) = \delta(C_i)/2$, and for k-Median, $\mathrm{cost}(C_i) = \min_{x \in \mathbb{R}^d} \{\delta(x, C_i)\}$. (In the k-Center problem, $\mathrm{cost}(C_i) = \min_{x \in \mathbb{R}^d} \max_{y \in C_i} \{\delta(x, y)\}$, and $\mathrm{cost}(C_1, C_2, \ldots, C_k) = \max_i \mathrm{cost}(C_i)$.) We use C_1^*, C_2^*, ..., C_k^* to denote a clustering of V of minimum cost c^*.

Instances of points in $\mathbb{R}^d$ are computationally hard if d is part of the input.[2]

Comments. We view the main contribution of this paper as the results rather than the proofs themselves. Can these results be strengthened? Well, it is likely that the metric k-clustering algorithm could be combined with the ideas of the k-median algorithm, to yield a much faster algorithm for metric k-clustering. We feel that the results presented here are robust, in the sense that they could probably be extended to other related problems. However, one limit of the results is that they crucially rely on k being fixed. Doing k-clustering for k large is a completely different problem which requires entirely different tools.

In terms of proof techniques, everything in this paper is elementary and can be proved from first principles. What gave us the momentum to go through lengthy calculations and several erroneous earlier versions of the algorithms? In the metric setting, Lemma 10 was essential in keeping up our optimism. In the ℓ_2^2 setting, Lemma 18 served the same role. Thus they can be seen as the central piece of the construction, and perhaps the one thing to remember from the arguments, for possible future re-use in other settings.

2 A PTAS for Metric Instances

In this section we present our algorithm for clustering metric spaces. Before we describe the algorithm, we give crucial basic properties and some definitions.

Proposition 1. Let $X, Y, Z \subseteq V$. Then $|Z|\delta(X, Y) \leq |X|\delta(Y, Z) + |Y|\delta(Z, X)$.

Corollary 2. Let $C \subseteq V$. For every vertex $v \in C$ we have $\delta(v, C) \geq \delta(C)/(2|C|)$.

Definition 3. Let $I_j = (\epsilon^{j+1}, \epsilon^j]$. Let $n_1 \geq n_2 \geq \cdots \geq n_k$ be the cluster sizes. Let $j_0 \leq k^2$ be the minimum j such that for every i, i', the ratio $n_i/n_{i'}$ is outside the interval I_j. Call a cluster index i *large* if $n_i \geq \epsilon^{j_0} n_1$ and *small* if $n_i < \epsilon^{j_0+1} n_1$.

[2] An exception to this rule is the case of Euclidean distance. The hardness of the problems considered here in the Euclidean case is an open problem.

In our proofs, the following quantities will come up frequently as upper or lower bounds to various cluster sizes, so we use some specific notations for them.

Notation 4.

$$M = n_1 = \max\{n_i\}, \quad m = \min\{n_i \mid i \text{ large}\}, \quad s = \max\{n_i \mid i \text{ small}\}.$$

The advantage of the above definition is that there is a large gap between the sizes of large and of small clusters, much larger than between the sizes of any two large clusters.

Fact 5. $s/m \leq \epsilon^2 \cdot m/M$.

Definition 6. Let $\beta = \epsilon M/m$. We say that two large clusters A and B are *close* if $\delta(A, B) < \beta(\delta(A) + \delta(B))$, and that they are *large* otherwise.

Now, the algorithm uses random sampling to have some rough estimate of the position of the clusters in the metric space. In fact, we will use just one sample point per cluster. For the algorithm to work, those sample points must be representative. (As usual, we can always run the algorithm several times to boost up its success probability). The representatives satisfy a couple of handy properties described in the lemmas below.

Definition 7. Let C be a set of points. An element c of C is said to be *representative* of C if $\delta(c, C) \leq 2\delta(C)/|C|$.

Lemma 8. Consider a partition $(C_1, \ldots, C_k)$ of V such that C_i has size n_i. For each large i, let c_i be a random uniform element of V. Then, with probability at least $(\epsilon^{j_0}/(2k))^k$, we have the following: for every large i, point c_i is a representative element of C_i.

Lemma 9. Let C_i^* and C_j^* be two large clusters, and c_i, c_j their representatives. Assume that C_i^* and C_j^* are close to each other. Then: $\delta(c_i, c_j) \leq 2(M/m)OPT/(m^2\epsilon)$.

Lemma 10. Let c be a representative point of cluster C. Then, for any x in V, we have: $|\delta(x, C) - |C|\delta(x, c)| \leq 2\delta(C)/|C|$.

Our algorithm uses, as a black box, an approximation scheme for Metric Max-k-Cut which is already known in the litterature. The Metric Max-k-Cut problem takes as input a set V of n points from an arbitrary metric space, and outputs a partition of V into k clusters $C_1, C_2, \ldots, C_k$ so as to maximize the total distance between pairs of points in different clusters, $\sum_i \sum_{j>i} \delta(C_i, C_j)$. For any partition into k clusters, the sum of the Max-k-Cut value and of the k-Clustering value is constant and equal to the sum of all distances, thus the same partition is optimal for both objective functions. Unfortunately, from the viewpoint of approximation, which involves controlling the relative error, the two problems are quite different, since in general the optimal k-clustering value could be much smaller than the optimal Max-k-Cut value. However, the Max-k-Cut approximation algorithm is still useful when the clusters are close together.

Theorem 11 ([9]). Let k be a fixed integer. Then there is a polynomial time approximation scheme for Metric Max-k-Cut.[3] The running time is $O(n^2 + nk2^{O(1/\epsilon^3)})$.

[3]Theorem 11 is actually an easy extension of the Max Cut approximation scheme of [9]: The same reduction which is used there for Max Cut also applies to Max-k-Cut, and the resulting weighted dense graph is only a variant of dense graphs in the usual sense, so that the Max-k-Cut approximation schemes for dense graphs (see [13, 3]) apply.

2.1 The k-Clustering Algorithm

We are now ready to describe the k-clustering algorithm. The algorithm presented below is randomized, but making it deterministic is straightforward. Fix $\epsilon > 0$. Our algorithm consists of taking the best of all partitions that are generated as follows.

1. By exhaustive search, guess the optimal cluster sizes $n_1 \geq n_2 \geq \cdots \geq n_k$. Define large and small as in Definition 3.
2. Define far and close as in Definition 6. By exhaustive search, for each pair of large cluster indices i and j, guess whether C_i^* and C_j^* are close to each other.
3. Taking the equivalence relation which is the transitive closure of the relation "C_i^* and C_j^* are close to each other", define a partition of large cluster indices into groups.
4. For each large cluster C_i^*, let c_i be a random uniform element of V. Assign each point $x \in V$ to the group G which minimizes $\min_{i \in G}[n_i \delta(x, c_i)]$.
5. By exhaustive search, for each group G thus constructed, guess $|G \cap S|$, where $S = \cup_{i \text{ small}} C_i^*$ is the union of small clusters. For each x assigned to group G, let $f(x) = \min_{i \in G} \delta(x, c_i)$. Remove from G's assignment the $|G \cap S|$ elements with largest value $f(x)$.
6. Partition each group of large clusters into the appropriate number h of clusters using the PTAS for Max-h-Cut with error parameter $\epsilon' = \epsilon^2 \epsilon^{3j_0}/(3k^3)$.
7. Recursively partition the removed elements into the appropriate number of clusters.

Theorem 12. For any fixed positive integer k, the algorithm presented in Section 2.1 is a PTAS for the Metric k-Clustering problem. The running time of the algorithm is $O(n^{3k} f(k, \epsilon))$, where $f(k, \epsilon)$ has a leading factor of $exp((1/\epsilon)^{k^2})$.

The running time analysis can be proved by inspection of the algorithm. We outline the analysis of the cost of the clustering constructed. The proof is a rather long and technical, sometimes tricky, but not particularly interesting elementary calculation, involving a careful management of the various error terms. The interesting facts have already been spelled out in the beginning of the section.

We first analyze the mistakes made in step 4 of the algorithm. For any two large clusters i and j which belong to different groups, let $F(i, j)$ denote the set of points $x \in C_i^*$ such that $\min_\ell n_\ell \delta(x, c_\ell) = n_j \delta(x, c_j)$. These points, which really should be in i's group, are mistakenly placed by the algorithm in j's group. Let $C_i = C_i^*$ for i small cluster, and

$$C_i = C_i^* + \cup_j F(j, i) - \cup_j F(i, j) \quad \text{for } i \text{ large.}$$

Using the bilinearity of $\delta(\cdot, \cdot)$, one can then develop $\delta(C_i)$, and analyze each term separately, to prove that $\sum_i \delta(C_i) \leq \sum_i \delta(C_i^*)(1 + 80k^3\epsilon)$. One ingredient of the calculation involves showing that $F(j, i)$ has small cardinality. We include that argument for the diligent reader, as an illustrative example of our proofs.

Lemma 13.

$$|F(j,i)| \leq \frac{8}{1-8\epsilon} m\epsilon.$$

Proof: Let $F = F(j,i)$ for shorthand, and let $x \in F$. By Lemma 10, $\delta(x, C_i^*) \leq n_i \delta(x, c_i) + \delta(C_i^*)(2/n_i)$. By the choice of the algorithm, $n_i \delta(x, c_i) \leq n_j \delta(x, c_j)$. By Lemma 10 again, $n_j \delta(x, c_j) \leq \delta(x, C_j^*) + \delta(C_j^*)(2/n_j)$. Thus $\delta(x, C_i^*) \leq \delta(x, C_j^*) + (\delta(C_i^*) + \delta(C_j^*))(2/m)$. Summing over x gives :

$$\delta(F, C_i^*) - \delta(F, C_j^*) \leq \frac{2}{m}(\delta(C_i^*) + \delta(C_j^*))|F|. \tag{1}$$

Now, since i and j are in different groups, C_i^* and C_j^* are far from each other, so

$$\delta(C_i^* \cup C_j^*) > \beta(\delta(C_i^*) + \delta(C_j^*)). \tag{2}$$

Let $x \in F$. By Proposition 1, $\delta(C_i^* \cup C_j^*) \leq 2\delta(x, C_i^* \cup C_j^*)|C_i^* \cup C_j^*|$. Summing over x, we get $|F|\delta(C_i^* \cup C_j^*) \leq 4M(\delta(F, C_i^*) + \delta(F, C_j^*))$. We now use Equation 1:

$$|F|\delta(C_i^* \cup C_j^*) \leq 4M(2\delta(F, C_j^*) + \frac{2}{m}(\delta(C_i^*) + \delta(C_j^*))|F|).$$

Since $F \subset C_j^*$, we have $\delta(F, C_j^*) \leq \delta(C_j^*)$. Combining with Equation 2 and factoring in $|F|$ gives

$$|F|(\delta(C_i^*) + \delta(C_j^*))(\beta - \frac{8M}{m}) \leq 8M\delta(C_j^*).$$

Replacing β by its value and solving in $|F|$ yields the Lemma. □

To analyze the mistakes made in the next step of the algorithm, let (C_i') denote the clustering obtained from (C_i) as follows. Let G denote a group, and for each cluster C_i of G, let $\mathrm{Out}(i)$ denote the elements of C_i which are (mistakenly) removed from G by the algorithm. Let $\mathrm{In}(G)$ denote the elements of S which (mistakenly) get to stay in G. We have:

$$|\mathrm{In}(G)| = \sum_{i \text{ cluster of } G} |\mathrm{Out}(i)|.$$

Thus, we can pair up the vertices of $\cup_i \mathrm{Out}(i)$ in a one-to-one fashion with the vertices of $\mathrm{In}(G)$. For i large, let C_i' denote the elements of C_i which get to stay in G, plus the elements of $\mathrm{In}(G)$ which are paired up with elements of $\mathrm{Out}(i)$. For i small, let C_i' denote the elements of C_i which stay outside the groups, plus the elements paired up with elements of C_i which end up in large groups.

By convention, we will always use (v, v') for elements which are paired, with v denoting the element which goes out of the large cluster and v' the element which goes out of the small cluster. One important piece of our calculation involves proving that

$$\sum \delta(v, v') \leq (2 + 6k\epsilon^2 + 2k^2\epsilon)\frac{OPT}{m}.$$

Once we have that, it does not take too much more work to show that

$$\forall i, \quad \delta(C_i') \leq \delta(C_i) + 3k(2 + 6k\epsilon^2 + 2k^2\epsilon)\epsilon^2 OPT.$$

Finally, we need to analyze the use of Max-h-Cut in step 6 of the algorithm; Consider a group $C_1^* \cup C_2^* \cup \cdots \cup C_h^*$. Let $c = \delta(C_1^*) + \cdots + \delta(C_h^*)$ and $W = \delta(C_1^* \cup \cdots \cup C_h^*) = \sum_{i,j} \delta(C_i^*, C_j^*)$. We can show that: $W \leq 3k^3/\epsilon(1/\epsilon^{j_0})^3 c$. Running the PTAS for Max-h-Cut with error parameter $\epsilon' = \epsilon\epsilon^{3j_0}/(3k^3)$, the error is then at most $\epsilon' W \leq \epsilon c$. Overall, the algorithm produces a cut of value at most $(1 + O(k^4\epsilon + k^2\epsilon^2))OPT$. Assuming that $\epsilon < 1/k$, this is $OPT(1 + O(k^2\epsilon^2))$.

3 Properties of the Square of Euclidean distance

¿From now on, $\delta(x, y) = \|x - y\|_2^2$. We denote by $\mathrm{conv}(X)$ the *convex hull* of $X = \{x^1, x^2, \ldots, x^n\} \subseteq \mathbb{R}^d$. Let $y = \sum_{i=1}^n (q_i/r)x^i$ be a point in $\mathrm{conv}(X)$ which is a rational convex combination of X (so r and q_i are integers). We associate with y a multi-subset Y of X of size r, obtained by taking q_i copies of x^i, for all i. Notice that the center of mass $\overline{Y}$ of Y equals y. The following proposition characterizes the cost of a cluster in terms of the center of mass.

Proposition 14. For every finite $X \subset \mathbb{R}^d$, $\delta(X) = |X|\delta(\overline{X}, X)$.

Proposition 15. Let Y be a multi-subset of $\mathbb{R}^d$. Then $\overline{Y}$ minimizes $\delta(Y, z)$ over z. In other words, $\overline{Y} = \arg\min_{z \in \mathbb{R}^d} \{\delta(Y, z)\}$.

Proposition 16. For every $x, y, z \in \mathbb{R}^d$, $\delta(x, z) \leq \delta(x, y) + \delta(y, z) + 2\sqrt{\delta(x, y) \cdot \delta(y, z)}$.

Proposition 17. For every $x \in \mathbb{R}^d$, for every multi-subset Y of $\mathbb{R}^d$, we have: $\delta(x, Y) \geq |Y|\delta(x, \overline{Y})$.

The first part of the following lemma is attributed to Maurey [6]. We denote the diameter of Y by $\mathrm{diam}(Y) = \max_{x,y \in Y} \delta(x, y)$. Since this is a key lemma, we choose to include its proof in this extended abstract.

Lemma 18. Let $Y \subset \mathbb{R}^d$ and $\epsilon > 0$.

1. **(Maurey)** For every $x \in \mathrm{conv}(Y)$, there exists a multi-subset Z of Y containing $1/\epsilon$ points and whose center of mass is close to x: $\delta(x, \overline{Z}) \leq \epsilon \cdot \mathrm{diam}(Y)$.
2. There exists a multi-subset Z of Y containing $\frac{1}{\epsilon}$ points and whose center of mass is close to the center of mass of Y: $\delta(\overline{Y}, \overline{Z}) \leq \epsilon\delta(Y, \overline{Y})/|Y|$.

Proof: We start with the first assertion. Let $t = 1/\epsilon$ and $x = \sum_{y \in Y} \alpha_y y$, where the α_y's are non-negative and sum up to 1. We use the probabilistic method. Pick a multiset $Z = \{z^1, z^2, \ldots, z^t\}$ at random, where the z^i-s are i.i.d. random variables with $\Pr[z^i = y] = \alpha_y$. Now, it is easy to see that

$$\begin{aligned} E\left[\delta(x, \overline{Z})\right] &= E\left[\frac{1}{t^2} \sum_{i=1}^{t} \sum_{j=1}^{t} (x - z^i) \cdot (x - z^j)\right] \\ &= \frac{1}{t^2} \sum_{i=1}^{t} \left(E\left[\|x - z^i\|_2^2\right] + \sum_{j \neq i} E\left[(x - z^i) \cdot (x - z^j)\right] \right). \end{aligned}$$

Since z^i and z^j are independent, we have $E\left[(x-z^i)\cdot(x-z^j)\right] = \sum_{l=1}^{d} E\left[(x_l - z_l^i)\right] E\left[\left(x_l - z_l^j\right)\right]$ which is 0 by our choice of distribution. Thus,

$$E(\delta(x,\overline{Z})) = \frac{1}{t^2}\sum_{i=1}^{t} E\left[\|x-z^i\|_2^2\right] \leq \frac{1}{t}\mathrm{diam}(Y).$$

Therefore there exists a choice of Z such that $\delta(x,\overline{Z}) \leq \frac{1}{t}\mathrm{diam}(Y)$.

For the second assertion, we start the proof in the same way, with $x = \overline{Y}$, and replace the last part of the calculation by the following slightly finer estimate:

$$\frac{1}{t^2}\sum_i E(\delta(\overline{Y},z^i)) = \frac{1}{t^2}\sum_i\sum_{y\in Y}\frac{1}{|Y|}\delta(\overline{Y},y) = \frac{\delta(\overline{Y},Y)}{t|Y|}. \qquad \square$$

Lemma 18 can be used to derive a high-probability result as follows.

Lemma 19. There exists a constant κ such that the following holds. Let $Y \subset \mathbb{R}^d$ and $\epsilon, \rho > 0$. Let Z be a random multi-subset of Y generated by taking $\kappa \cdot \frac{1}{\epsilon^2} \cdot \log\frac{1}{\rho}$ i.i.d. points distributed uniformly in Y. Then, with probability at least $1-\rho$, we have: $\delta(\overline{Y},\overline{Z}) \leq \epsilon\delta(Y,\overline{Y})/|Y|$.

4 A PTAS for ℓ_2^2 Instances of k-Clustering

Our algorithm consists of taking the best of all partitions that are generated as follows.

1. By exhaustive search, guess the optimal cluster sizes $|C_i| = n_i$. By exhaustive search, consider all possible sequences $A_1, A_2, \ldots, A_k$, where the A_i-s are mutually disjoint multisets, each containing $16/\epsilon^2$ points from V.

2. Compute a minimum cost assignment of points of V to clusters $C_1, C_2, \ldots, C_k$, subject to the conditions that exactly n_i points are assigned to C_i, when the cost of assigning a point x to C_i is $\hat{\delta}(x, C_i) = n_i \cdot \delta(x, \overline{A_i})$, for all $i = 1, 2, \ldots, k$.

Our algorithm is motivated by the following bound.

Lemma 20. Let Y be a multi-subset of V and $1 \geq \epsilon > 0$. Then there exists a multi-subset Z of Y of size $|Z| = 16/\epsilon^2$ such that $\delta(Y,\overline{Z}) \leq (1+\epsilon)\delta(Y,\overline{Y})$.

Theorem 21. The above algorithm is a PTAS for the ℓ_2^2 k-Clustering problem. Its running time is $n^{O(k/\epsilon^2)}$.

5 A PTAS for ℓ_2^2 Instances of k-Median

A simple variant of the above algorithm solves the k-Median case and has similar running time. Here we give a much faster randomized polynomial time approximation scheme for ℓ_2^2 instances of k-Median: the running time of our algorithm, for fixed k, ϵ, and failure probability ρ, is just $O(n(\log n)^{O(1)})$. The approximation scheme consists of taking the best of all partitions that are generated as follows.

1. By exhaustive search, guess an approximation $n_1 \geq n_2 \geq \cdots \geq n_k$ on the sizes of the k clusters, where n_i is the power of $(1+\epsilon)$ larger than and closest to $|C_i^*|$.

2. Partition the k clusters into groups in a greedy fashion: 1 goes into the first group, and for i going from 2 to k, i goes into the current group if $n_i \geq (\epsilon/16k)^2 n_{i-1}$, and into a new group otherwise. Let T be the number of groups and let m_t denote the size of the largest cluster in the t^{th} group. Let $m_{T+1} = 0$.

3. For t going from 1 to T, do the following:

   (a) Let U_t denote the points not yet clustered (initially $U_1 = V$).

   (b) Let Z denote a random uniform sample of U_t, with replacement, of constant size (size $k^{2k}/(16\epsilon)^{2k} \cdot (\ln k)\gamma/\epsilon^6$, where $\gamma > 0$ is a constant).

   (c) By exhaustive search, guess $A_i = Z \cap C_i^*$ for all i in the t^{th} group. Define, for each such cluster C_i, the representative point as $c_i = \overline{A_i}$. (If $A_i = \emptyset$, take an arbitrary point as the representative of C_i.)

   (d) Assign $|U_t| - m_{t+1}16k^2/\epsilon$ points from U_t to the clusters in groups 1 through t, where point x is assigned to a cluster C_i that minimizes $\delta(x, c_i)$.

This completes the specification of the algorithm. We now proceed with its analysis.

Consider the iteration of the algorithm where all the guesses are correct. For all $t \leq T$, let a_t denote the index of the first and largest cluster in the t^{th} group (so that $m_t = n_{a_t}$), and let b_t denote the index of the last and smallest cluster in that group. Consider the situation when the algorithm starts iteration t. For each j in group t, let $U_{jt} = C_j^* \cap U_t$ denote the points which have not yet been classified and which we hope the algorithm will place in cluster j during iteration t.

Definition 22. For $j \in [a_t, b_t]$, we say that C_j^* is a *well-represented* cluster if $|U_{jt}| \geq \epsilon^3/16^3 \cdot n_j$. Otherwise C_j^* is called *poorly represented.*

Lemma 23. Fix a cluster index j and let t be j's group. For every $\rho > 0$ and for every sufficiently large $\lambda > 0$, there exists $\gamma > 0$ such that with probability at least $1 - \frac{\rho}{k}$, if j is a surviving index then we have: $|A_j| \geq \ln k \lambda/\epsilon^4$.

The following Lemma motivates the terminology of well-represented clusters.

Lemma 24. For every $\rho > 0$ there exist $\lambda > 0$ and $\gamma > 0$ such that with probability at least $1 - \rho$, we have:

$$\forall t, \forall j \text{ surviving index}, \quad \left|\delta(U_{jt}, c_j) - \delta(U_{jt}, \overline{U_{jt}})\right| \leq \frac{\epsilon}{8} \cdot \delta(U_{jt}, \overline{U_{jt}}). \tag{3}$$

For $x \in X$, denote by j_x the index of the cluster that x gets assigned to by the algorithm, and denote by j_x^* the index of the cluster that x gets assigned to by the optimal clustering. Let D_t denote the set of points which are assigned during iteration t of the loop in step 3 of the algorithm. Such points can be classified into three categories:

- *regular* points: $x \in D_t$ is regular if its optimal cluster j_x^* has $j_x^* \leq b_t$ and is well-represented.
- *premature* points: $x \in D_t$ is premature if $j_x^* > b_t$, i.e. the optimal cluster of x is too small to be taken into consideration yet. Let P_t denote the premature points in D_t.
- *leftover* points: $x \in D_t$ is leftover if $j_x^* \leq b_t$ and cluster j_x^* is poorly represented. Let L_t denote the premature points of D_t.

We analyze these categories separately and charge the contribution of premature points to regular points, and of leftover points to the rest. This eventually leads to the following theorem.

Theorem 25. With constant probability the above algorithm computes a solution whose cost is within a factor of $1+\epsilon$ of the optimum cost. The running time of the algorithm is $O(g(k,\epsilon) \cdot n \cdot (\log n)^k)$, where $g(k,\epsilon) = \exp\left(\frac{1}{\epsilon^8} \cdot k^3 \ln k \cdot \left(\ln \frac{1}{\epsilon} + \ln k\right)\right)$.

Acknowledgements: We thank Piotr Indyk and Ravi Kannan for stimulating discussions and helpful suggestions.

References

[1] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. In *Proc. of the 41th Ann. IEEE Symp. on Foundations of Computer Science(FOCS)* 2000, 240-250.

[2] N. Alon and B. Sudakov. On two segmentation problems. *Journal of Algorithms*, 33:173–184, 1999.

[3] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *J. Comp. System. Sci.*, 58:193–210, 1999.

[4] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via Core-Sets. *Proc. 34th ACM STOC (2002)*, pages 250-257.

[5] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Proc. of the 6th Int'l World Wide Web Conf.*(WWW), 1997, pages 391–404.

[6] B. Carl and I. Stephani. *Entropy, Compactness and the Approximation of Operators.* Cambridge University Press, 1990.

[7] S. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.

[8] W. Fernandez de la Vega, M. Karpinski, and C. Kenyon. A polynomial time approximation scheme for metric MIN-BISECTION. *ECCC TR02-041*, 2002.

[9] W. Fernandez de la Vega and C. Kenyon. A randomized approximation scheme for metric MAX CUT. In *Proc. of the 39th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, 1998, pages 468-471, also in JCSS 63 (2001). pages 531-541.

[10] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proc. of the 10th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 1999, pages 291–299.

[11] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3):231–262, 1994.

[12] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[13] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. of the ACM*, 45:653–750, 1998.

[14] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proc. of the 9th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, January 1998, 649-657.

[15] N. Guttmann-Beck and R. Hassin. Approximation algorithms for min-sum p-clustering. *Disc. Applied Math.*, 89:125–142, 1998.

[16] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *Proc. of the 40th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, 1999, 154-159.

[17] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi. On the hardness of approximating Max k-Cut and its dual. In *Proc. of the 4th Israeli Symp. on Theory of Computing and Systems (ISTCS)*, 1996. Also in *Chicago Journal of Theoretical Computer Science*, 1997.

[18] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems. In *Proc. of the 30th Ann. ACM Symp. on Theory of Computing (STOC)*, 1998, pages 473–482.

[19] N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. In *Proc. of the 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, January 2001, pages 439–447.

[20] R. Ostrovsky and Y. Rabani. Polynomial time approximation schemes for geometric clustering problems. *J. of the ACM*, 49(2):139–156, March 2002.

[21] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[22] L.J. Schulman. Clustering for edge-cost minimization. In *Proc. of the 32nd Ann. ACM Symp. on Theory of Computing (STOC)*, 2000, pages 547–555.

[23] R. Shamir and R. Sharan. Algorithmic approaches to clustering gene expression data. In T. Jiang, T. Smith, Y. Xu, M.Q. Zhang eds., *Current Topics in Computational Biology*, MIT Press, to appear.

[24] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7:11–32, 1991.