Approximation Schemes for Clustering Problems (extended abstract)

W. Fernandez de la Vega [*] Marek Karpinski [†] Claire Kenyon [‡] Yuval Rabani [§]

ABSTRACT

Let k be a fixed integer. We consider the problem of partitioning an input set of points endowed with a distance function into k clusters. We give polynomial time approximation schemes for the following three clustering problems: Metric k-Clustering, ℓ_2^2 k-Clustering, and ℓ_2^2 k-Median. In the k-Clustering problem, the objective is to minimize the sum of all intra-cluster distances. In the k-Median problem, the goal is to minimize the sum of distances from points in a cluster to the (best choice of) cluster center. In metric instances, the input distance function is a metric. In ℓ_2^2 instances, 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). For the first two problems, our results are the first polynomial time approximation schemes. For the third problem, the running time of our algorithms is a vast improvement over previous work.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations

*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 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

General Terms

Algorithms,Theory

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 [8, 12, 30, 16], or interpretation of experimental data in molecular biology [29].

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 to solve exactly even for $k = 2$ (using arguments similar to those in [14, 13]).

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. The running time is $O(f(k,\epsilon)n^{3k})$ for the metric case, and $n^{O(k/\epsilon^2)}$ for the ℓ_2^2 case. Our algorithms can be modified to handle variants which exclude outliers. The details are omitted from this extended abstract.

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, our algorithms can be modified easily to derive polynomial time approximation schemes for other objective functions, such as the k-Center problem. (Similar results for k-Center were known previously [1, 6].)

Related work. The k-Clustering problem was proposed by Sahni and Gonzalez [27] in the setting of arbitrary weighted graphs. Unfortunately, only poor approximation guarantees are possible [23, 17]. Guttman-Beck and Hassin [20] initiated the study of the problem in metrics. Schulman [28] gave probabilistic algorithms for ℓ_2^2 k-Clustering. (Thus he also handled other interesting cases of metrics that embed isometrically into this distance space, such as Euclidean metrics or L^1 metrics.) 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. The running time is linear if $d = o(\log n/\log\log n)$ and otherwise the running time is $n^{O(\log\log n)}$. Thus our results improve and extend Schulman's result, giving a true polynomial time approximation scheme for arbitrary dimension.

Earlier, Fernandez de la Vega and Kenyon [14] presented a polynomial time approximation scheme for Metric Max Cut, an objective function that is the complement of Metric 2-Clustering. Indyk [21] later used this algorithm to derive a polynomial time approximation scheme for the latter problem. Thus our results extend Indyk's result to the case of arbitrary fixed k. Bartal, Charikar, and Raz [7] gave a polynomial time approximation algorithm with polylogarithmic performance guarantees for Metric k-Clustering where k is arbitrary (i.e., part of the input).

As mentioned above, instances of k-Median in finite metrics with fixed k are trivially solvable in polynomial time. (For arbitrary k, the problem is APX-hard [19] and has elicited much work and progress [5, 11, 22, 10].) 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 [15], who gave a 2-approximation algorithm. Ostrovsky and Rabani [26] gave a polynomial time approximation scheme for this case and other geometric settings. Our results improve significantly the running time for the ℓ_2^2 case. Recently and independently of our work, Bădoiu, Har-Peled, and Indyk [6] gave a polynomial time approximation scheme for the Euclidean case with much improved running time. (The running time of their algorithm is similar to ours. Their paper includes other results on other clustering objectives.) Their algorithm and analysis are in some respects similar to our algorithm (though it handles a different distance function).

It is interesting to note that both Schulman's algorithm for k-Clustering and the algorithm of Fernandez de la Vega and Kenyon for Mertic Max Cut use a similar idea of sampling data points at random from a biased distribution that depends on the pairwise distances. In recent research on clustering problems, sampling has been the core idea in the design of provably good algorithms for various objective functions. Examples include [3, 2, 25].

Comments and notation. 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. Instances of points in $\mathbb{R}^d$ are usually computationally hard if d is part of the input.[1]

For simplicity, we omit the ceiling notation from expressions such as $\lceil 1/\epsilon\rceil$. Our proofs can be modified trivially to handle the rounding error. 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 $\delta(X, X)$. We use $C_1^*, C_2^*, \ldots, C_k^*$ to denote a clustering of V of minimum cost c^* (depending, of course, on the objective function being discussed).

2. METRIC K-CLUSTERING

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

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

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

Let $I_j = (\epsilon^{j+1}, \epsilon^j]$. Let $n_1 \ge n_2 \ge \cdots \ge n_k$ be the cluster sizes. Let $j_0 \le k^2$ 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 $n_i \ge \epsilon^{j_0} n_1$ and *small* if $n_i < \epsilon^{j_0+1} n_1$. In our proofs, the following quantities will come up frequently as upper or lower bounds to various cluster sizes: $M = n_1 = \max\{n_i\}$, $m = \min\{n_i \mid i \text{ large}\}$, $s = \max\{n_i \mid i \text{ small}\}$. Notice 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 3. $s/m \le \epsilon^2 \cdot m/M$. □

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

Our algorithm uses random sampling. In fact, we will use just one sample point per cluster. For the algorithm to work, those sample points must be representative in the following sense: Let C be a set of points. An element c of C is said to be *representative* of C iff $\delta(c, C) \le 2\delta(C)/|C|$. (As usual, we can always run the algorithm several times to boost up its success probability.) The representatives satisfy a few handy properties described in the lemmas below.

Lemma 4. Let c be a representative point of cluster C. Then, for every $x \in V$, we have: $|\delta(x, C) - |C|\delta(x, c)| \le 2\delta(C)/|C|$.

Proof sketch: Apply Proposition 1. □

Lemma 5. 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.

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

Proof: We are trying to estimate the probability of the conjunction of independent events, which is the product of the probabilities. The largest cluster has size $n_1 \geq n/k$. Since C_i is large, it has size at least $\epsilon_{j_0} n_1 \geq n\epsilon_{j_0}/k$. With probability at least ϵ_{j_0}/k, point c_i is in C_i. Conditioning on that happening, c_i is a random uniform element of C_i, so on average, $\delta(c_i, C_i)$ equals $\delta(C_i)/|C_i|$. By Markov's inequality, the probability that it is representative is then at least $1/2$. Overall, the probability that c_i is representative of C_i is at least $\epsilon_{j_0}/(2k)$. $\square$

Lemma 6. Let C_i^* and C_j^* be two large clusters in an optimal solution, and let c_i, c_j be their representatives. Assume that C_i^* and C_j^* are close. Then: $\delta(c_i, c_j) \leq 2(M/m)OPT/(m^2\epsilon)$.

Proof: By a variant of Proposition 1, we have:

$$n_i n_j \delta(c_i, c_j) \leq n_j \delta(c_i, C_i^*) + \delta(C_i^*, C_j^*) + n_i \delta(C_j^*, c_j).$$

Since the points are representatives, this implies:

$$\delta(c_i, c_j) \leq \frac{\delta(C_i^*)}{n_i^2} + \frac{\delta(C_i^*, C_j^*)}{n_i n_j} + \frac{\delta(C_j^*)}{n_j^2}.$$

Since the two clusters are close to each other,

$$\delta(C_i^*, C_j^*) \leq \frac{M}{m\epsilon}(\delta(C_i^*) + \delta(C_j^*)).$$

Thus:

$$\delta(c_i, c_j) \leq \frac{OPT}{m^2}(1 + \frac{M}{m\epsilon}). \qquad \square$$

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

We are now ready to describe and analyze the k-Clustering algorithm. We present a randomized version of the algorithm. Derandomizing it is straightforward. Fix $\epsilon > 0$. Our algorithm consists of taking the best of all partitions that are generated by the code titled "the metric k-clustering algorithm" at the end of the paper.

[2]Theorem 7 is an easy extension of the Max Cut approximation scheme of [14]: 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 [18, 4]) apply.

Theorem 8. For any fixed positive integer k, the Metric k-Clustering algorithm is a polynomial time approximation scheme. The running time of the algorithm is $O(f(k,\epsilon)\cdot n^{3k})$, where $f(k,\epsilon)$ is of the order of $\exp((1/\epsilon)^{k^2})$.

The running time analysis can be proved by inspection of the algorithm. The rest of this section will be devoted to analyzing the cost of the clustering constructed by the algorithm. 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 $C_i = C_i^* + \cup_j F(j,i) - \cup_j F(i,j)$ for i large, and let $C_i = C_i^*$ for i small.

Proposition 9. $\sum_i \delta(C_i) \leq \sum_i \delta(C_i^*)(1 + 80k^3\epsilon)$.

To prove this Proposition, we need the following lemma.

Lemma 10. $\delta(F(j,i), C_i^*) - \delta(F(j,i), C_j^*) \leq \frac{2}{m}(\delta(C_i^*) + \delta(C_j^*))|F(j,i)|$.

Proof: Let $x \in F(j,i)$. By Lemma 4, we have

$$\delta(x, C_i^*) \leq n_i \delta(x, c_i) + 2\frac{\delta(C_i^*)}{n_i}.$$

By the choice of the algorithm, $n_i \delta(x, c_i) \leq n_j \delta(x, c_j)$. By Lemma 4 again, we have

$$n_j \delta(x, c_j) \leq \delta(x, C_j^*) + 2\frac{\delta(C_j^*)}{n_j}.$$

Thus

$$\delta(x, C_i^*) \leq \delta(x, C_j^*) + \frac{2}{m}(\delta(C_i^*) + \delta(C_j^*)).$$

Summing over $x \in F(j,i)$ concludes the proof of the lemma. $\square$

To be able to use Lemma 10, we need an upper bound on $|F(j,i)|$.

Lemma 11. $|F(j,i)| \leq \frac{8}{1-8\epsilon} m\epsilon$.

Proof: Let $F = F(j,i)$ for shorthand. Since i and j are in different groups, C_i^* and C_j^* are far from each other, so

$$\delta(C_i^* \cup C_j^*) > \beta(\delta(C_i^*) + \delta(C_j^*)). \tag{1}$$

Consider $x \in F$. By Proposition 1, we have

$$\delta(C_i^* \cup C_j^*) \leq 2\delta(x, C_i^* \cup C_j^*)|C_i^* \cup C_j^*|.$$

Summing over $x \in F$, we get

$$|F|\delta(C_i^* \cup C_j^*) \leq 4M(\delta(F, C_i^*) + \delta(F, C_j^*)).$$

We now use the result of Lemma 10.

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

Since $F \subset C_j^*$, we have $\delta(F, C_j^*) \leq \delta(C_j^*)$. Combining with Equation 1 and factoring in $|F|$ gives

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

We conclude that $|F| \leq \frac{8M}{\beta - 8M/m}$, and it only remains to replace β by its value to get the statement of the Lemma. $\square$

Plugging the result of Lemma 11 into Lemma 10 (for the first inequality), and using Proposition 1 followed by Lemma 11 (for the next two inequalities), yields the following Corollary.

Corollary 12.

1. $\delta(F(j,i), C_i^*) - \delta(F(j,i), C_j^*) \le \epsilon \frac{16}{1-8\epsilon} c^*$;
2. $\delta(F(j,i)) \le \epsilon \frac{16}{1-8\epsilon} c^*$;
3. $\delta(F(i,j), F(i,j')) \le \epsilon \frac{16}{1-8\epsilon} c^*$. $\square$

Lemma 13. $\delta(F(j,i), F(j',i)) \le \epsilon \frac{16}{1-8\epsilon} c^*$.

Proof: By Proposition 1, we have $|C_i^*|\delta(F(j,i), F(j',i)) \le |F(j',i)|\delta(F(j,i), C_i^*) + |F(j,i)|\delta(F(j',i), C_i^*)$. By Lemma 11, this yields

$$\delta(F(j,i), F(j',i)) \le \frac{8}{1-8\epsilon}\epsilon(\delta(F(j,i), C_i^*) + \delta(F(j',i), C_i^*)).$$

By the first statement of Corollary 12, this can be replaced by $\delta(F(j,i), F(j',i)) \le \frac{8}{1-8\epsilon}\epsilon(\delta(F(j,i), C_j^*) + \delta(F(j',i), C_{j'}^*) + \epsilon\frac{32}{1-8\epsilon}c^*)$. Since $F(j,i) \subset C_j^*$ and $F(j',i) \subset C_{j'}^*$, we have: $\delta(F(j,i), C_j^*) + \delta(F(j',i), C_{j'}^*) \le c^*$, hence the lemma. $\square$

Proof of Proposition 9: We write: $\sum_i \delta(C_i) = \sum_i \delta(C_i^* + \cup_j F(j,i) - \cup_j F(i,j)) = \sum_i \delta(C_i^*) + [\sum_{i,j} \delta(C_i^*, F(j,i)) - \sum_{i,j} \delta(C_i^*, F(i,j))] + \sum_i \delta(\cup_j F(j,i) - \cup_j F(i,j))$. We exchange the roles of i and j on the right hand side to bound the brackedted quantity using the first statement of Corollary 12. We use bilinearity of $\delta(\cdot,\cdot)$ and appeal to the rest of the corollary to bound the other terms. This gives the bound of the proposition. $\square$

Before we can continue modifying the clustering, we need to prove that C_a is not too different from C_a^*. The following lemma is an easy consequence of Lemma 11.

Lemma 14. $||C_a| - |C_a^*|| \le \frac{8k}{1-8\epsilon}\epsilon|C_a^*|$. $\square$

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.

Lemma 15. $\sum \delta(v, v') \le (2 + 6k\epsilon^2 + 2k^2\epsilon)\frac{c^*}{m}$.

Proof: Let a be a large cluster and $v \in \mathrm{Out}(a)$, and let v' be the element which is paired with v, and let G denote a's group. Why did v' end up in G rather than v ? Because there is some large cluster b also in group G, such that

$$\delta(v', c_b) = f(v') < f(v) \le \delta(v, c_a).$$

This yields

$$\delta(v, v') \le \delta(v, c_a) + \delta(c_a, c_b) + \delta(c_b, v') \le 2\delta(v, c_a) + \delta(c_a, c_b).$$

Since a and b are in the same group, there is a chain of at most k clusters connecting them, such that consecutive clusters along the chain are close. By Lemma 6, this implies

$$\delta(c_a, c_b) \le k\frac{2M}{m\epsilon}\frac{c^*}{m^2}.$$

By Proposition 1, we have:

$$\delta(v, c_a) \le \frac{\delta(v, C_a) + \delta(C_a, c_a)}{|C_a|}.$$

By the choice of the algorithm, we have

$$\delta(c_a, C_a) \le \delta(c_a, C_a^*)|C_a|/|C_a^*| \le 2(1 + \frac{8k}{1-8\epsilon}\epsilon)\frac{\delta(C_a^*)}{|C_a^*|} \le 3\frac{c^*}{m}.$$

Hence

$$\delta(v, v') \le 2\frac{\delta(v, C_a)}{m} + 6\frac{c^*}{m^2} + k\frac{2M}{m\epsilon}\frac{c^*}{m^2}.$$

Summing and realizing that the number of terms is at most the sum of the cardinalities of the small clusters, which is at most ks, we get

$$\sum \delta(v, v') \le (2 + 6k\frac{s}{m} + k^2\frac{2M}{m\epsilon}\frac{s}{m})\frac{c^*}{m}.$$

Now, remember Fact 3:

$$\sum \delta(v, v') \le (2 + 6k\epsilon^2 + 2k^2\epsilon)\frac{c^*}{m}. \qquad \square$$

Equipped with this Lemma, we are now ready to attack the analysis of the clustering (C_i').

Lemma 16. For every small i, $\delta(C_i') \le \delta(C_i) + 3k(2 + 6k\epsilon^2 + 2k^2\epsilon)\epsilon^2 c^*$.

Proof: Let b be a small cluster. Let $C_b' = C_b + P(b) - M(b)$. By bilinearity, we can write $\delta(C_b') = \delta(C_b) + [\delta(C_b, P(b)) - \delta(C_b, M(b))] + [\delta(P(b)) - \delta(P(b), M(b))] + [\delta(M(b)) - \delta(M(b), P(b))]$. Since $\delta(u, v) - \delta(u, v') \le \delta(v, v')$, it is easy to see that $\delta(P(b)) - \delta(P(b), M(b)) \le |P(b)| \sum \delta(v, v') \le ks(2 + 6k\epsilon^2 + 2k^2\epsilon)\frac{c^*}{m} \le k(2 + 6k\epsilon^2 + 2k^2\epsilon)\epsilon^2 c^*$. Similarly, $\delta(M(b)) - \delta(M(b), P(b)) \le k(2 + 6k\epsilon^2 + 2k^2\epsilon)\epsilon^2 c^*$. Now, let $v \in P(b)$ and v' paired with v. We write with Proposition 1 $\delta(v, C_b) \le |C_b|\delta(v, v') + \delta(v', C_b)$. Summing, we get

$$\delta(P(b), C_b) \le ks\sum_v \delta(v, v') + \delta(M(b), C_b).$$

We apply Lemma 15 to yield $\delta(P(b), C_b) - \delta(M(b), C_b) \le ks(2 + 6k\epsilon^2 + 2k^2\epsilon)\frac{c^*}{m} \le k(2 + 6k\epsilon^2 + 2k^2\epsilon)\epsilon^2 c^*$. Summing our various inequalities gives the lemma. $\square$

The only thing left to do is analyze the modifications to the large clusters.

Lemma 17. For every large a, $\delta(C_a') \le \delta(C_a) + (6k\epsilon^2 + 2k^2\epsilon)c^*$.

Proof: We use the same notations as in the proof of Lemma 15. Similarly to the pervious Lemma we can easily get

$\delta(C'_a) \leq \delta(C_a) + [\delta(C_a, P(a)) - \delta(C_a, M(a))] + 2k(3+2k^2\epsilon)\epsilon^2 c^*.$

Now, recall that $\delta(c_a, C_a) \leq 3c^*/m$. By Proposition 1,

$$\delta(v', C_a) \leq \delta(v', c_a)|C_a| + 3\frac{c^*}{m}.$$

$$\delta(v', c_a) \leq \delta(v', c_b) + \delta(c_b, c_a) \leq \delta(v, c_a) + k\frac{2M}{m\epsilon}\frac{c^*}{m^2}.$$

Hence

$$\delta(v', C_a) \leq |C_a|\delta(v, c_a) + k\frac{2M}{m\epsilon}\frac{c^*}{m} + 3\frac{c^*}{m}.$$

Now,

$$|C_a|\delta(v, c_a) \leq \delta(v, C_a) + \delta(C_a, c_a) \leq \delta(v, C_a) + 3\frac{c^*}{m}.$$

Replacing and summing over $v' \in P(a)$, and remembering that $|P(a) = |M(a)| \leq ks$, we obtain

$$\begin{aligned} \delta(P(a), C_a) &\leq \delta(M(a), C_a) + (6 + k\frac{2M}{m\epsilon})\frac{c^*}{m}ks \\ &\leq \delta(M(a), C_a) + (6k\frac{s}{m} + 2\frac{k^2}{\epsilon}\frac{M}{m}\frac{s}{m})c^* \\ &\leq \delta(M(a), C_a) + (6k\epsilon^2 + 2k^2\epsilon)c^*. \quad \square \end{aligned}$$

Finally, we need to analyze the use of Max-h-Cut in step 6 of the algorithm. We will present the analysis as if the group was perfect, i.e. consisted of the clusters C_i^*. (It is easy to see that the proof also goes through when replacing the C_i^* by C'_i, at the cost of some bookkeeping of the small errors introduced at every step of the calculation.) In the groups of large clusters, we can prove that c^* is $\Omega(\sum_{V\times V}\delta(x,y))$ as follows.

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 have:

$$\begin{aligned} \delta(C_i^*, C_j^*) &\leq n_j\delta(C_i^*, c_i) + n_i n_j \delta(c_i, c_j) + n_i\delta(c_j, C_j^*) \\ &\leq M2\frac{\delta(C_i^*)}{m} + M^2 k\frac{2M}{m\epsilon}\frac{c}{m^2} + M2\frac{\delta(C_j^*)}{m}. \end{aligned}$$

Summing over the k^2 terms gives

$$W \leq 4\frac{M}{m}kc + \frac{2k^3}{\epsilon}(\frac{M}{m})^3 c \leq 3\frac{k^3}{\epsilon}(1/\epsilon_{j_0})^3 c.$$

Run the PTAS for Max-h-Cut with error parameter

$$\epsilon' = \frac{\epsilon\epsilon_{j_0}^3}{3k^3}\epsilon.$$

The error is then at most $\epsilon' W \leq \epsilon c$.

Overall, the algorithm produces a cut of value at most $(1 + O(k^4\epsilon + k^2\epsilon^2))c^*$. Assuming that $\epsilon < 1/k$, this is $(1 + O(k^2\epsilon^2))c^*$. $\square$

3. ℓ_2^2 K-CLUSTERING

In this section and the next section, $\delta(x,y) = \|x-y\|_2^2$. 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 18. For every finite $X \subset \mathbb{R}^d$, $\delta(X) = |X|\delta(\overline{X}, X)$. $\square$

Proposition 19. Let Y be a multi-subset of $\mathbb{R}^d$. Then $\overline{Y}$ minimizes $\delta(Y, z)$ over z. In other words,

$$\overline{Y} = \arg\min_{z\in\mathbb{R}^d}\{\delta(Y,z)\}. \quad \square$$

Proposition 20. For every $x, y, z \in \mathbb{R}^d$, $\delta(x,z) \leq \delta(x,y) + \delta(y,z) + 2\sqrt{\delta(x,y)\cdot\delta(y,z)}$. $\square$

Proposition 21. For every $x \in \mathbb{R}^d$, for every multi-subset Y of $\mathbb{R}^d$, we have: $\delta(x, Y) \geq |Y|\delta(x, \overline{Y})$. $\square$

The first part of the following lemma is attributed to Maurey [9]. We denote the diameter of Y by $\mathrm{diam}(Y) = \max_{x,y\in Y}\delta(x,y)$.

Lemma 22. 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

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

Since z^i and z^j are independent, we have $E\left[(x - z^i)\cdot(x - z^j)\right] = \sum_{l=1}^d E\left[(x_l - z_l^i)\right]E\left[(x_l - z_l^j)\right]$ 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:

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

$\square$

Lemma 22 can be used to derive a high-probability result as follows.

Lemma 23. 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: $\delta(\overline{Y}, \overline{Z}) \leq \epsilon\delta(Y, \overline{Y})/|Y|$. $\square$

Our algorithm consists of taking the best of all partitions that are generated by the code titled "the ℓ_2^2 k-Clustering algorithm" at the end of the paper. Our algorithm is motivated by the following bound.

Lemma 24. 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 $\delta(Y, \overline{Z}) \leq (1+\epsilon)\delta(Y, \overline{Y})$.

Proof: By Proposition 20, for every $y \in Y$, $\delta(y, \overline{Z}) \leq \delta(y, \overline{Y}) + \delta(\overline{Y}, \overline{Z}) + 2\sqrt{\delta(y, \overline{Y})\delta(\overline{Y}, \overline{Z})}$. By the Cauchy-Schwarz inequality, $\sum_{y \in Y} \sqrt{\delta(y, \overline{Y})} \leq \sqrt{|Y| \sum_{y \in Y} \delta(y, \overline{Y})}$. Therefore, summing the previous expression over $y \in Y$, we get that $\delta(Y, \overline{Z}) \leq \delta(Y, \overline{Y}) + |Y|\delta(\overline{Y}, \overline{Z}) + 2\sqrt{|Y|\delta(Y, \overline{Y})\delta(\overline{Y}, \overline{Z})}$. Plugging in the bound for $\delta(\overline{Y}, \overline{Z})$ from Lemma 22, we get that $\delta(Y, \overline{Z}) \leq (1 + \frac{\epsilon}{2} + \frac{\epsilon^2}{16})\delta(Y, \overline{Y}) \leq (1+\epsilon)\delta(Y, \overline{Y})$. $\square$

Theorem 25. The ℓ_2^2 k-Clustering algorithm is a polynomial time approximation scheme. Its running time is $n^{O(k/\epsilon^2)}$.

Proof: By Lemma 24 applied to $Y = C_i^*$, for every $i = 1, 2, \ldots, k$, there exists a multi-subset Z_i of C_i^* of size $|Z_i| = 16/\epsilon^2$, such that $\delta(C_i^*, \overline{Z_i}) \leq (1+\epsilon)\delta(C_i^*, \overline{C_i^*})$. Consider the iteration of the algorithm where $A_i = Z_i$ and $n_i = |C_i^*|$ for every $i = 1, 2, \ldots, k$. Let $C_1, C_2, \ldots, C_k$ be the clustering computed by the algorithm in this iteration, and let c be the cost of this clustering. Then,

$$\begin{aligned}
c &= \sum_{i=1}^{k} |C_i| \cdot \sum_{x \in C_i} \delta(x, \overline{C_i}) \\
&\leq \sum_{i=1}^{k} n_i \cdot \sum_{x \in C_i} \delta(x, \overline{A_i}) \\
&\leq \sum_{i=1}^{k} n_i \cdot \sum_{x \in C_i^*} \delta(x, \overline{A_i}) \\
&\leq (1+\epsilon) \cdot \sum_{i=1}^{k} |C_i^*| \cdot \delta(C_i^*, \overline{C_i^*}) \\
&= (1+\epsilon) \cdot c^*.
\end{aligned}$$

The performance guarantee follows because the algorithm finds a partition whose cost is at least as good as c.

As for the running time of the algorithm, there are less than n^k possible representations of n as a sum $n_1 + n_2 + \cdots + n_k$. There are less than n^{16k/ϵ^2} possible choices for $\mathcal{A}$. Computing a minimum cost assignment to clusters can be done using a minimum cost perfect matching algorithm in time $O(n^3 \log n)$. $\square$

4. ℓ_2^2 *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 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 by the code titled "the ℓ_2^2 k-Median algorithm" at the end of the paper. We will proceed with the analysis of the algorithm. Consider the iteration of the algorithm where all the guesses are correct. For all $t = 1, 2, \ldots, T$, let a_t denote the index of the first and largest cluster in the t^{th} group (so $m_t = n_{a_t}$), and let b_t denote the index of the last and smallest cluster in that group.

Lemma 26. For all $t \in \{1, 2, \ldots, T\}$, the number of points in the smallest and in the largest clusters of group t are not very different: $\left(\frac{\epsilon}{16k}\right)^{2(k-1)} n_{a_t} \leq n_{b_t} \leq n_{a_t}$. $\square$

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

Lemma 27. 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 $\gamma > 0$ (the γ used to define the size of Z) such that with probability at least $1 - \frac{\rho}{k}$, if j is well-represented then we have $|A_j| \geq \frac{\lambda}{\epsilon^4} \ln k$.

Proof sketch: Use Lemma 26 and the definition of being well-represented to bound $|U_{jt}|/|U_t|$ from below, then use standard Chernoff bounds for A_j. $\square$

Lemma 28. For every $\rho > 0$ there exist $\lambda > 0$ and $\gamma > 0$ such that with probability at least $1-\rho$, we have, for all t and for all well-represented j,

$$\left|\delta(U_{jt}, c_j) - \delta(U_{jt}, \overline{U_{jt}})\right| \leq \frac{\epsilon}{8} \cdot \delta(U_{jt}, \overline{U_{jt}}). \tag{2}$$

Proof sketch: Apply Lemma 23 to the sample A_j in U_{jt}, so that for j well-represented and $|A_j|$ large enough, with probability at least $1 - \rho/(3k)$ we have

$$\delta(c_j, \overline{U_{jt}}) \leq \frac{\epsilon^2}{2^{10}} \cdot \delta(U_{jt}, \overline{U_{jt}})/|U_{jt}|. \tag{3}$$

(This defines λ.) Set γ according to Lemma 27 so that if j is well-represented, then A_j is large enough with probability at least $1-\rho/(3k)$. By the proof of Lemma 24, Equation 3 then implies $\left|\delta(U_{jt}, c_j) - \delta(U_{jt}, \overline{U_{jt}})\right| \leq (\epsilon/8) \cdot \delta(U_{jt}, \overline{U_{jt}})$. Summing failure probabilities then concludes the proof. $\square$

In the rest of the analysis we will assume that Equation 2 holds. 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 iff 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 $j_x^* \le b_t$ and j_x^* is poorly represented. Let L_t denote the leftover points of D_t.

We start the analysis with the easiest category, that of regular points.

Lemma 29.

$$\sum_{x \text{ regular}} \delta(x, c_{j_x}) \le \left(1+\frac{\epsilon}{8}\right) \cdot \sum_{j \text{ well-represented}} \delta(C_j^*, \overline{C_j^*}).$$

Proof: Take x a regular point and let t be the group containing j_x^*. Then $x \in U_{j_x^* t}$. Thus the left hand side of the sum ranges over U_{jt}, where j is well-represented. The assignment of x by the algorithm has value $\delta(x, c_{j_x}) \le \delta(x, c_{j_x^*})$ by definition of the algorithm. Thus:

$$\begin{aligned}
\sum_{x \text{ regular}} \delta(x, c_{j_x}) &\le \sum_{x \text{ regular}} \delta(x, c_{j_x^*}) \\
&\le \sum_{j \text{ well-represented}} \delta(U_{jt}, c_j) \\
&\le (1+\frac{\epsilon}{8}) \sum_{j \text{ well-represented}} \delta(U_{jt}, \overline{U_{jt}}) \\
&\le (1+\frac{\epsilon}{8}) \sum_{j \text{ well-represented}} \delta(U_{jt}, \overline{C_j^*}) \\
&\le (1+\frac{\epsilon}{8}) \sum_{j \text{ well-represented}} \delta(C_j^*, \overline{C_j^*}).
\end{aligned}$$

□

We now deal with the category of premature points. The proof of this lemma crucially uses the specific feature of the algorithm according to which one keeps assigning unsufficiently many points to the clusters under consideration. Thus this is one of the key points in the analysis.

Lemma 30.

$$\sum_{x \text{ premature}} \delta(x, c_{j_x}) \le \frac{\epsilon}{8} \cdot \sum_{x \text{ not premature}} \delta(x, c_{j_x}).$$

Proof: First note that by definition of premature points, P_t has size at most $|\cup_{j>b_t} C_j^*| \le k m_{t+1}$. By definition of the algorithm, the number of points in U_{t+1} is exactly $16k^2 m_{t+1}/\epsilon$. Since $|\cup_{j>b_t} C_j^*|$ has size at most $k m_{t+1}$, in U_{t+1} there must be at least $m_{t+1}(16k^2/\epsilon - k) > m_{t+1} 8k^2/\epsilon$ points which belong to $C_1^* \cup \cdots \cup C_{b_t}^*$ (hence which are not premature). Among those, let S_t denote the $|P_t|$ points such that $\delta(x, c_{j_c^*})$ is smallest.

Since the algorithm chooses a minimum cost assignment and prefers P_t over S_t in doing so during iteration t, we have that $\sum_{P_t} \delta(x, c_{j_x}) \le \sum_{S_t} \delta(x, c_{j_x^*})$. The right hand side is at most

$$\frac{|S_t|}{|U_{t+1} \cap (C_1^* \cup \cdots \cup C_{b_t}^*)|} \sum_{x \in U_{t+1}, x \text{ not premature}} \delta(x, c_{j_x}),$$

and this in turn is at most

$$\frac{\epsilon}{8k} \sum_{x \text{ not premature}} \delta(x, c_{j_x}).$$

Summing over t yields the lemma. □

Finally, we deal with the leftover points.

Lemma 31.

$$\sum_{x \text{ leftover}} \delta(x, c_{j_x}) \le \sum_{j \text{ poorly represented}} \delta(C_j, \overline{C_j^*})$$
$$+O(\epsilon^3) \sum_{y \text{ premature}} \delta(y, c_{j_y})$$
$$+2\sqrt{\sum_{j \text{ poorly represented}} \delta(C_j, \overline{C_j^*}) O(\epsilon^3) \sum_{y \text{ premature}} \delta(y, c_{j_y})}.$$

Proof sketch: Let C_j^* be a poorly represented cluster and t be its group. By definition of being poorly represented, most of the points of C_j^* were assigned before their turn, i.e., they got assigned to some cluster of index $< a_t$; thus most of the points of C_j^* were premature. Take $x \in C_j^*$, x leftover, and $y \in C_j^*$, y premature. We have:

$$\delta(x, c_{j_x}) \le \delta(x, c_{j_y}) \le \delta(x, y) + \delta(y, c_{j_y}) + 2\sqrt{\delta(x, y)\delta(y, c_{j_y})}.$$

Summing over $x \in C_j^* \cap L$ (leftover) and $y \in C_j^* \cap P$ (premature), and using the Cauchy-Schwartz Inequality, we get:

$$|C_j^* \cap P| \sum_{C_j^* \cap L} \delta(x, c_{j_x}) \le \delta(C_j^*) + |C_j^* \cap L| \sum_{C_j^* \cap P} \delta(y, c_{j_y})$$
$$+2\sqrt{\delta(C_j^*)|C_j^* \cap L| \sum_{C_j^* \cap P} \delta(y, c_{j_y})}.$$

Now, by definition of leftover points, we have

$$|C_j^* \cap L| \le \frac{\epsilon^3}{16^3} n_j \text{ and } |C_j^* \cap P| \ge n_j(1 - \frac{\epsilon^3}{16^3}).$$

Thus, replacing, we have:

$$\sum_{C_j^* \cap L} \delta(x, c_{j_x}) \le (1 + O(\epsilon^3)) \frac{\delta(C_j^*)}{|C_j^*|} + O(\epsilon^3) \sum_{C_j^* \cap P} \delta(y, c_{j_y})$$
$$+2\sqrt{\frac{\delta(C_j^*)}{|C_j^*|} O(\epsilon^3) \sum_{C_j^* \cap P} \delta(y, c_{j_y})}.$$

It only remains to apply Proposition 18, sum over j, and use Cauchy-Schwartz again to deduce the statement of the lemma. □

We are now ready to prove the main theorem of this section.

Theorem 32. With constant probability the ℓ_2^2 k-Median 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 $g(k,\epsilon) = \exp\left(\frac{1}{\epsilon^8} \cdot k^3 \ln k \cdot \left(\ln \frac{1}{\epsilon} + \ln k\right)\right)$.

Proof sketch: Let c denote the cost of the clustering produced by the algorithm in the iteration analyzed above. Clearly, the algorithm outputs a solution of cost at most c. Assume that Equation 2 holds. (We will make sure this happens with constant probability.) We have that $c = \sum_{j=1}^k \sum_{x \in C_j} \delta(x, \overline{C_j}) \le \sum_{j=1}^k \sum_{x \in C_j} \delta(x, c_j) = \sum_{x \in X} \delta(x, c_{j_x})$. We separate the sums into three parts R, P, L corresponding to regular, premature and leftover points, apply the three lemmas above to the three parts, and then a short algebraic manipulation yields that the cost is $(1+O(\epsilon))c^*$.

As for the running time of the algorithm, the number of sequences $n_1, n_2, \ldots, n_k$ that the algorithms has to enumerate over is $O\left(\left(\log_{1+\epsilon} n\right)^k\right)$. The total number of sequences of cluster representatives that are enumerated over is at most

$$2^{\left(\frac{1}{\epsilon^4}\cdot k^3 \ln k\cdot\left(\ln\frac{1}{\epsilon}+\ln k\right)\right)}.$$

Computing the augmentation of a partial solution to the next group given the representatives of its clusters requires $O(n)$ distance computations, where the hidden constant depends mildly on k and ϵ. □

5. ACKNOWLEDGEMENTS

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

6. REFERENCES

[1] P.K. Agarwal and C.M. Procopiuc. Exact and approximation algorithms for clustering. In *Proc. of the 9th Ann. ACM-SIAM Symp. on Discrete Algorithms*, January 1998, pages 658–667.

[2] 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.

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

[4] 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.

[5] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proc. of the 30th Ann. ACM Symp. on Theory of Computing*, 1998.

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

[7] Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k-clustering in metric spaces. In *Proc. of the 33rd Ann. ACM Symp. on Theory of Computing*, July 2001, pages 11–20.

[8] 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.

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

[10] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proc. of the 40th Ann. IEEE Symp. on Foundations of Computer Science*, 1999.

[11] M. Charikar, S. Guha, D.B. Shmoys, and É. Tardos. A constant factor approximation algorithm for the k-median problem. In *Proc. of the 31st Ann. ACM Symp. on Theory of Computing*, 1999.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

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

[21] 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.

[22] K. Jain and V.V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proc. of the 40th Ann. IEEE Symp. on Foundations of Computer Science*, 1999.

[23] 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.

[24] 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.

[25] 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.

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

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

[28] 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.

[29] 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.

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

The Metric k-Clustering algorithm

1. By exhaustive search, guess the optimal cluster sizes $n_1 \geq n_2 \geq \cdots \geq n_k$.
2. By exhaustive search, for each pair of large cluster indices i and j, guess whether C_i^* and C_j^* are close or far.
3. Taking the equivalence relation which is the transitive closure of the relation "C_i^* and C_j^* are close", 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 $\epsilon' = \epsilon^2 \epsilon^{3 j_0} / (3k^3)$.
7. Recursively partition the removed elements into the appropriate number of clusters.

The ℓ_2^2 k-Clustering algorithm

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 $i = 1, 2, \ldots, k$.

The ℓ_2^2 k-Median algorithm

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 $\delta(x, c_i)$.