A Sublinear Time Approximation Scheme for Clustering in Metric Spaces

PIOTR INDYK *
STANFORD UNIVERSITY
indyk@cs.stanford.edu

Abstract

The metric 2-clustering *problem is defined as follows: given a metric (X, d), partition X into two sets S_1 and S_2 in order to minimize the value of*

$$\sum_i \sum_{\{u,v\} \subset S_i} d(u, v)$$

In this paper we show an approximation scheme for this problem.

1 Introduction

In this paper we consider the following k-*clustering* problem: given a weighted graph $G = (X, d)$ on N vertices, where $d(\cdot, \cdot)$ is a weight function, partition X into k sets $S_1 \dots S_k$ such that the value of

$$\sum_i \sum_{\{u,v\} \subset S_i} d(u, v)$$

is minimized. This problem was first formally posed by Sahni and Gonzalez [7]. They observed that the problem is NP-complete (for $k \geq 2$) and by reduction from k-coloring showed that it cannot be approximated up to any constant (for $k \geq 3$). Instead, they proposed a $1/k$-approximation algorithm for the dual version of this problem where the goal is to *maximize* the weight of edges which *do not* belong to any cluster (i.e. k-max cut). Unfortunately, the latter result does not (and cannot) imply any bounded approximation ratio for the k-clustering problem. The hardness of k-clustering (for $k \geq 3$) was later strengthened by Kann et al [3]. The case $k = 2$ is also known as the *minimum edge deletion bipartition problem.* Garg et al [5] gave a $O(\log N)$-approximation algorithm for this case.

A standard way to reduce the complexity of clustering problems is to assume that the weight function d is a metric. For this case Guttman-Beck and Hassin [4] showed a 2-approximation algorithm running in time $N^{O(k)}$ (they also gave slightly improved bounds for the case $k = 2$ assuming $|S_1| = |S_2|$). Recently, de la Vega and Kenyon [2] gave a polynomial time approximation scheme for the dual of metric 2-clustering (i.e. metric max cut). Unfortunately (as they mention) their result does not imply a PTAS for the 2-clustering problem.

In this paper we show a polynomial (in fact, sublinear) time approximation scheme for the 2-clustering problem. We obtain this result by focusing on the case when the de la Vega-Kenyon PTAS does not work, i.e. when the value of max cut is much higher than the value of the 2-clustering. We observe that in this case the clusters must be "well-separated", i.e. the average intracluster distance (for at least one cluster) has to be much smaller than the average intercluster distance. Therefore, we can assign a vertex to one of the two clusters by examining its average distance to a small sample of points from each cluster (we find this sample by exhaustive search). If done correctly, this procedure yields a $(1 + \epsilon)$-approximate solution.

2 Preliminaries

Let (X, d) be a metric space. For any two sets $A, B \subset X$ we define

$$d(A, B) = \sum_{a \in A, b \in B} d(a, b).$$

We use $d(A)$ to denote $d(A, A)/2$ (notice that this is equal to the sum of pairwise distances of points in A). Also, we will write $d(u, B)$ or $d(B, u)$ instead of $d(\{u\}, B)$ or $d(B, \{u\})$. Moreover, we define

$$\tilde{d}(A, B) = \frac{d(A, B)}{|A| \cdot |B|}$$

*Supported by Stanford Graduate Fellowship and NSF Grant IIS-9811904

Observe, that $\tilde{d}$ is a metric.

For sets A, B of equal cardinality, we define

$$d_M(A,B) = \min_{\pi:A \overset{1:1}{\to} B} \sum_{a\in A} d(a,\pi(a))$$

and $\tilde{d}_M(A,B) = \tilde{d}(A,B)/|A|$. Again, notice that both d_M and $\tilde{d}_M$ satisfy metric properties.

For any $\alpha \in [0,1]$, $u \in X$ and $A \subset X$, we define $d_\alpha(u,A)$ to be the sum of the $\alpha|A|$ smallest values of $\{d(u,a)|a \in A\}$. Notice, that $d_1(u,A) = d(u,A)$. We also define $\tilde{d}_\alpha(u,A) = d_\alpha(u,A)/(\alpha|A|)$.

The algorithms which we give in this paper are randomized and producing an $(1+\epsilon)$-approximate solutions *with high probability*, i.e. with probability $1 - 1/N^a$, where the exponent a can be made arbitrarily large by increasing the running time of the algorithm. For simplicity in the rest of the paper we will skip the exact dependence of the probability of success on the running time of the algorithms.

3 The algorithms

Our result is obtained by running three algorithms in parallel: *MAXCUT*, *BC* and *UC*. In the course of describing the algorithms we will use (S_1, S_2) to denote a (possibly one of many) clustering with minimal value. We will assume that $|S_1| = m$ and $|S_2| = n$ are given to us, as otherwise we use all N possible combinations and increase the running time by a factor of N (it is possible to reduce this factor to $O(\log_{1+\epsilon} N)$ but we skip the description here).

The algorithms are as follows:

MAXCUT: this is the algorithm of [2] for $(1+\epsilon)$-approximate MAX-CUT. This algorithm will be useful for us in case when the value of the maximum cut is at most $c = O(1)$ larger than the value of minimum 2-clustering. This is due to the fact that in this case any $(1+\epsilon)$-approximate solution obtained by MAXCUT is also a $(1+O(\epsilon))$-approximate solution for the 2-clustering problem. Thus we only need to give an algorithm for the case when the cut/clustering ratio exceeds c.

BC (for Balanced Clustering): we will use this algorithm when the balance ratio of the optimal 2-clustering defined as

$$\mathrm{b}(S_1,S_2) = \max\left(\frac{|S_1|}{|S_2|}, \frac{|S_2|}{|S_1|}\right)$$

is smaller than $\rho = O(1)$ and the maximum cut/minimum 2-clustering ratio is greater than c. In this case the algorithm proceeds as follows. Firstly, it chooses set T of $t = O(\rho \log n)$ points uniformly at random (with replacement) from X. Then, it guesses $T_1 \subset S_1 \cap T$ and $T_2 \subset S_2 \cap T$, such that $|T_1| = |T_2| = \lambda = O(\log n)$; this can be implemented deterministically by exploring all $2^t = n^{O(1)}$ possibilities. Then, it checks for each point $u \in X - T_1 - T_2$ if

$$|S_1| d_{1-\alpha}(u, T_1) \leq |S_2| d_{1-\alpha}(u, T_2)$$

for α chosen later. If the above inequality holds, then u is added to set R_1; otherwise we add it to R_2. The pair $(R_1 \cup T_1, R_2 \cup T_2)$ is returned as a solution.

The intuition behind this algorithm is that if the maximum cut is much larger than the 2-clustering cost, the clusters should be "well-separated". Thus a "typical" point u should be either much closer to S_1 than to S_2 or much closer to S_2 than S_1. Therefore the above "randomized comparison" scheme should determine S_1 and S_2 without much error, if T_1 and T_2 "reflect accurately" S_1 and S_2.

UC (for Unbalanced Clustering): we will use this algorithm when $\mathrm{bc}(S_1,S_2) > \rho$. Assume $|S_1| > \rho|S_2|$. In this case, we can easily obtain in polynomial time a set T of λ random points from S_1. Then we sort all points $u \in X - T$ by $d_{1-\alpha}(u,T)$ (in ascending order); the value of α is again determined later. The first $|S_1| - \lambda$ points from the list are added (together with T) to R_1, the remaining points are added to R_2. The algorithm outputs (R_1, R_2) as a solution.

4 The analysis of BC

We will start from relating the outcome of the (randomized) comparison of $d(u,T_1)$ and $d(u,T_2)$ to a certain (deterministic) property of u. More specifically, we will show the following Lemma.

Lemma 1 *Consider any $u \in S_1$. If for every set $S_2' \subset S_2$ such that $|S_2'| \geq (1-2\alpha)|S_2|$ we have $d(u,S_2') \geq d(u,S_1)$, then with high probability we have*

$$n \cdot d_{1-\alpha}(u,T_2) > m \cdot d_{1-\alpha}(u,T_1)$$

Before we prove this lemma, we mention that in the following we use its contraposition, i.e. if $n \cdot d_{1-\alpha}(u,T_2) \leq m \cdot d_{1-\alpha}(u,T_1)$, then we will assume that there exists a set S_2^u of cardinality at least $(1-2\alpha)|S_2|$ (for simplicity we will assume $|S_2^u| = (1-2\alpha)|S_2|$) such that $d(u,S_2^u) < d(u,S_1)$. Using this assumption we will prove the correctness of BC.

Proof: The proof uses a well-known idea that by removing some fraction of largest elements from the sample significantly increases the probability of correct estimation. Therefore, only a sketch of the proof is given.

Without loss of generality we can consider S_2' which contains smallest $(1-2\alpha)n$ elements from S_2. Moreover, we can assume that $d(u,S_2') = d(u,S_1)$ and both are equal to 1. Finally, we will assume that the largest $2\alpha n$ elements of S_2 are all equal; notice that they are upper bounded by $O(1/(\alpha m))$. It is easy to verify that for λ large enough we have a significant gap in the expected values of $d_{1-\alpha}(u,T_1)$ and $d_{1-\alpha}(u,T_2)$. More specifically, one can easily verify that:

- $\frac{m}{\lambda}E[d_{1-\alpha}(u,T_1)] \leq 1$
- $\frac{n}{\lambda}E[d_{1-\alpha}(u,T_2)] \geq 1+\alpha/2$

In the following we want to convert the expectation bounds into bounds holding with high probability. This is easy for T_2, as (from our assumptions) all elements of S_2 are small, i.e. $O(1/(\alpha n))$, therefore we can apply standard tail inequalities. To apply a similar method for S_1, assume that T_1 is chosen in the following manner: firstly, we choose an element q which has rank $(1-\alpha)m$ in T_1 (according to a proper distribution), and then we will choose upper- and lower-ranked elements of T_1 independently. Observe, that q is very likely to be close to the $(1-\alpha)$th quantile of S_1 with high probability when $\lambda = \Omega(\log n/\alpha)$. Assuming this holds, we know that the value of q is $O(1/(\alpha m))$ (i.e. small), so we can apply tail inequalities to T_1 as well. By fairly standard calculations we obtain that $n\cdot d_{1-\alpha}(u,T_2) > m\cdot d_{1-\alpha}(u,T_1)$ with high probability if $\lambda = \Omega(\log n/\alpha^4)$. □

Now we can proceed with the actual proof. We will upper bound the additional cost incurred by assigning $u \in S_1$ to R_2; the opposite case can be handled in the same way. From the above Lemma, we can assume that for every $u \in S_1$ which has been included in R_2 (i.e. such that $n\cdot d_{1-\alpha}(u,T_2) \leq m\cdot d_{1-\alpha} d(u,T_1)$) there exists a set S_2^u of cardinality $(1-2\alpha)|S_2|$ such that

$$d(u,S_2^u) \quad < \quad d(u,S_1) \tag{1}$$

Thus we need only to bound $d(u,S_2-S_2^u)$. We will be only interested in u's such that

$$d(u,S_1)(1+\epsilon) \quad \leq \quad d(u,S_2) \tag{2}$$

as otherwise the difference in costs can be easily bounded. From (1) and (2) we obtain that

$$d(u,S_2-S_2^u) \geq \epsilon d(u,S_1) \geq \epsilon d(u,S_2^u)$$

which can be rewritten as

$$\tilde{d}(u,S_2-S_2^u) \quad \geq \quad \frac{\epsilon(1-2\alpha)}{2\alpha}\tilde{d}(u,S_2^u)$$

By triangle inequality we have

$$\begin{aligned}\tilde{d}(S_2^u,S_2-S_2^u) \quad &\geq \quad \tilde{d}(u,S_2-S_2^u) - \tilde{d}(u,S_2^u)\\ &\geq \quad (1-\frac{2\alpha}{\epsilon(1-2\alpha)})\tilde{d}(u,S_2-S_2^u)\end{aligned}$$

The above gives a bound for $\tilde{d}(u,S_2-S_2^u)$. In the following we show that the number of such us is also not very large if (as we assume) $d(S_1,S_2) \geq c(d(S_1)+d(S_2))$. Firstly, observe that

$$\begin{aligned}&\tilde{d}(S_1,S_2)\\ \leq\ &\tilde{d}(S_1,u)+\tilde{d}(u,S_2)\\ =\ &\tilde{d}(S_1,u)+(1-2\alpha)\tilde{d}(u,S_2)+2\alpha\tilde{d}(u,S_2-S_2^u)\\ \leq\ &\tilde{d}(S_1,u)+(1-2\alpha)\tilde{d}(u,S_2)+2\alpha(\tilde{d}(u,S_2^u)+\\ &\tilde{d}(S_2^u,S_2-S_2^u))\\ =\ &\tilde{d}(S_1,u)+\tilde{d}(u,S_2^u)+2\alpha\tilde{d}(S_2^u,S_2-S_2^u)\end{aligned}$$

We can rewrite it as:

$$\begin{aligned}&d(S_1,S_2)\\ =\ &|S_2|d(S_1,u)+\frac{|S_1|}{1-2\alpha}d(u,S_2^u)+\\ &2\alpha\frac{|S_1|}{|S_2|(1-2\alpha)2\alpha}d(S_2^u,S_2-S_2^u)\\ \leq\ &|S_2|d(S_1,u)+\frac{|S_1|}{1-2\alpha}d(u,S_2^u)+\frac{\rho}{1-2\alpha}d(S_2)\end{aligned}$$

Therefore

$$\begin{aligned}&c(d(S_1)+d(S_2))\\ \leq\ &|S_2|d(S_1,u)+\frac{|S_1|}{1-2\alpha}d(u,S_2^u)+\frac{\rho}{1-2\alpha}d(S_2)\end{aligned}$$

or alternatively

$$\begin{aligned}&(c-\frac{\rho}{1-2\alpha})(d(S_1)+d(S_2))\\ \leq\ &|S_2|d(S_1,u)+\frac{|S_1|}{1-2\alpha}d(u,S_2^u)\\ \leq\ &(|S_2|+\frac{|S_1|}{1-2\alpha})d(u,S_1)\end{aligned}$$

where the last step follows from the fact that $d(S_1,u) \geq d(u,S_2^u)$.

The upper bound for the number of u satisfying the above inequality can be now obtained as follows. Assume that this number is equal to $\gamma|S_1|$. Notice that $\sum_{u\in S_1} d(u,S_1) \le d(S_1)$. By plugging in the lower bound for $d(u,S_1)$ we get

$$1 \ge \gamma(c - \frac{\rho}{1-2\alpha})\frac{1}{\rho + \frac{1}{1-2\alpha}}$$

Therefore

$$\gamma \le \frac{\rho + \frac{1}{1-2\alpha}}{c - \frac{\rho}{1-2\alpha}}$$

Denote the set of u's as above by U. We can bound the total cost difference by

$$\begin{aligned}
& \sum_{u\in U} d(u, S_2 - S_2^u) \\
\le & \sum_{u\in U} 2\alpha n \tilde{d}(u, S_2 - S_2^u) \\
\le & 2\alpha n \frac{1}{1-\frac{2\alpha}{\epsilon(1-2\alpha)}} \tilde{d}(S_2^u, S_2 - S_2^u) \\
\le & 2\alpha n \frac{1}{1-\frac{2\alpha}{\epsilon(1-2\alpha)}} \frac{d(S_2^u, S_2 - S_2^u)}{2\alpha n(1-2\alpha)n} \\
\le & \gamma \frac{\rho}{1-\frac{2\alpha}{\epsilon(1-2\alpha)}(1-2\alpha)} d(S_2) = A d(S_2)
\end{aligned}$$

Observe, that the factor A becomes smaller than ϵ when we set $c = \Omega(\frac{\rho^2}{\epsilon})$ and $\alpha = O(\epsilon)$, for sufficient constants.

5 The analysis of UC

As in the previous section, we start from stating Lemma which characterizes the properties we need from the random sampling process. As the proof is similar to the previous one, we skip it here.

Lemma 2 *Consider a pair $u \in S_1$ and $v \in S_2$. If for every set $S' \subset S_1$ such that $|S'| \ge (1-2\alpha)|S_1|$ we have $d(v,S') \ge (1+\alpha)d(u,S')$, then with high probability $d_{1-\alpha}(v,T_1) \ge d_{1-\alpha}(u,T_1)$, for $\lambda = \Theta(\log n/\alpha^4)$.*

From the Lemma we can assume (with high probability) that if $d_{1-\alpha}(v,S_1) < d_{1-\alpha}(u,S_1)$, then there exists a set S' of size at least $(1-2\alpha)|S_1|$ such that $d(v,S') < (1+\alpha)d(u,S')$.

From the way the algorithm works, it follows that for every $u \in S_1$ included by mistake to R_2 there is $v \in S_2$ included to R_1. Let U denote the set of mistaken u's and let V denotes the set of mistaken v's. Moreover, let f be any bijection from U to V. For each pair u, v such that $v = f(u)$ we denote the set S' as above by S^u (we will also use the notation S^v). Also, we will overload slightly the definitions of d and $\tilde{d}$ by using $d(U,S^u)$ to denote $\sum_{u\in U}\sum_{p\in S^u} d(u,p)$ and defining $\tilde{d}(U,S^u) = d(U,S^u)/\sum_{u\in U}|S_u|$. Similarly, we define $d(U, S_1 - S^u)$ [1].

In order to prove the correctness of the algorithm we will bound the differences $d(V,S_1) - d(U,S_1)$ and $d(U,S_2) - d(V,S_2)$. We start from the first one. Observe, that it is sufficient to bound $d(V,S_1 - S^v) - d(U,S_1 - S^u)$ (since $d(V,S^v) \le (1+\alpha)d(U,S^u)$ and we will make sure that $\alpha \le \epsilon$). We can focus only on the case $d(V,S_1-S^v) \ge \epsilon d(U,S^u)$, which can be written as

$$\tilde{d}(V,S_1 - S^v) \ge \frac{\epsilon(1-\alpha)}{\alpha}\tilde{d}(U,S^u).$$

To bound the RHS, we observe that (since any two sets S^u and $S^{u'}$ have $\ge (1-4\alpha)m$ size overlap)

$$\begin{aligned}
\tilde{d}_M(U,V) &\le \frac{1}{1-4\alpha}(\tilde{d}(U,S^u) + \tilde{d}(V,S^v)) \\
&\le \frac{2+\alpha}{1-4\alpha}\tilde{d}(U,S^u)
\end{aligned}$$

Therefore

$$\tilde{d}(V,S_1 - S^v) \le \frac{\epsilon(1-\alpha)}{\alpha}\frac{1-4\alpha}{2+\alpha}\tilde{d}(U,V)$$

We also use the fact

$$\tilde{d}_M(U,V) + \tilde{d}(U,S_1 - S^u) \ge \tilde{d}(V,S_1 - S^u)$$

and therefore

$$\begin{aligned}
& \tilde{d}(U,S_1 - S^u) \\
\ge & \tilde{d}(V,S_1 - S^u) - \tilde{d}_M(U,V) \\
\ge & \tilde{d}(V,S_1 - S^u)(1 - \alpha\frac{2+\alpha}{1-4\alpha}\frac{1}{\epsilon(1-\alpha)}) \\
= & \tilde{d}(V,S_1 - S^u)F
\end{aligned}$$

Thus

$$\begin{aligned}
& d(V,S_1 - S^u) - d(U,S_1 - S^u) \\
\le & (1/F - 1)d(U,S_1 - S^u) \\
\le & (1/F - 1)d(S_1)
\end{aligned}$$

In this way we bounded the first component. Notice that by setting $\alpha = O(\epsilon^3)$ we make the value of $1/F - 1$ smaller than ϵ.

To bound $d(U,S_2) - d(V,S_2)$, we write it as $|V|n(\tilde{d}(U,S_2) - \tilde{d}(V,S_2))$, and use the inequality

$$\tilde{d}_M(U,V) + \tilde{d}(V,S_2) \ge \tilde{d}(U,S_1)$$

[1] We feel that the abuse of notation is partially justified by the fact for all u_1, u_2 the sets S^{u_1} and S^{u_2} have significant overlap (i.e. at least $(1-4\alpha)$) and therefore we can almost treat them as equal sets.

obtaining

$$\begin{aligned} & |V|n(\tilde{d}(U,S_2)-\tilde{d}(V,S_2)) \\ \leq\ & |V|n\tilde{d}_M(U,V) \\ \leq\ & \frac{2+\alpha}{1-4\alpha}|V|n\tilde{d}(U,S^u) \\ \leq\ & \frac{(2+\alpha)n}{(1-4\alpha)(1-\alpha)m}d(U,S^u) \\ \leq\ & 1/\rho\frac{(2+\alpha)}{(1-4\alpha)(1-\alpha)}d(S_1)=Bd(S_1) \end{aligned}$$

Observe that by setting $\rho=\Omega(1/\epsilon)$ we make B smaller than ϵ.

By combining the above algorithms and using the fact that the algorithm in [2] runs in $O(n^2\exp(1/\epsilon))$ time (which makes it negligible compared to the running times of BC and UC), we show the following Theorem.

Theorem 1 *There is a randomized $(1+\epsilon)$-approximation algorithm for 2-clustering running in time $n^{O(1/\epsilon^5)}$.*

6 Sublinear time algorithm

In this section we sketch the description of how to improve the running time of the above algorithm to $O(\log^{1/\epsilon^{O(1)}} n^{1+\gamma})$, for any $\gamma>0$. Since the size of the input (the description of the metric) is $\Theta(n^2)$, the algorithm runs in time *sublinear* in the input size.

Firstly, we observe that the running time of UC is essentially bounded by the time needed to collect a sample of t points from the larger cluster. By using random sampling we can perform this task in time roughly $O((1+1/\rho)^t)$. By making ρ sufficiently large we can make the running time $n^{1+\gamma}$, for any $\gamma>0$.

The main time bottleneck is the time needed for exhaustive partitioning of the set T in the BC procedure. Below we show that that subroutine can be actually replaced by recursive 2-clustering of T (say into C_1 and C_2) and choosing $T_1\subset C_1$ and $T_2\subset C_2$. Although we are no longer guaranteed that $T_i\subset S_i$, we show that the new T_i's are good enough for the purpose of our algorithm.

Specifically, we use the following lemma from [6] (presented here in a slightly different form):

Lemma 3 *Let (S_1,S_2) and (S_1',S_2') be two partitions of the metric space over S. There exists a constant B such that for any A and any $\beta<1/B$ if $d(S_1)+d(S_2)\leq\beta A/Bd(S_1,S_2)$ and $d(S_1')+d(S_2')\leq A/B(d(S_1)+d(S_2))$, then (S_1,S_2) and (S_1',S_2') differ on at most βn points.*

In other words, the lemma says that we the two clusters S_1 and S_2 are very separated, then any clustering with cost close to the optimal differs very little from (S_1,S_2). We use this fact as follows. Select a sample R of r points. It can be easily shown that (with constant probability) R can be split into R_1 and R_2 such that the ratio $\frac{d(R_1,R_2)}{d(R_1)+d(R_2)}$ is comparable (up to a constant) to the ratio $\frac{d(S_1,S_2)}{d(S_1)+d(S_2)}\geq\rho$. Assuming that ρ and R are large enough, we know (from the above Lemma) that we can find $T_1'\subset R_1$ and $T_2'\subset R_2$ such that $|T_1'|=|T_2'|=t$ and $|T_1'-S_1|+|T_2'-S_2|\leq\beta t$ (or we can swap R_1 with R_2). It turns out that T_1' and T_2' are almost as good as T_1 and T_2 obtained by exhaustive search. More specifically, it is easy to see that for any two equal size sets A',A, if $\max(|A-A'|,|A'-A|)\leq\beta|A|$ then for any α and u

$$d_{1-\alpha-\beta}(u,A)\leq d_{1-\alpha}(u,A')\leq d_{1-\alpha+\beta}(u,A)$$

Therefore, we can prove a version of Lemma 1 where

$$n\cdot d_{1-\alpha}(u,T_2)\leq m\cdot d_{1-\alpha}d(u,T_1)$$

is replaced by

$$n\cdot d_{1-3\alpha/2}(u,T_2')\leq m\cdot d_{1-\alpha/2}d(u,T_1')$$

by taking $\beta\leq\alpha/2$. This means that if we replace T_1 and T_2 in the BC algorithm by T_1' and T_2', the algorithm is still correct.

Finally, we can improve the running time of the MAXCUT procedure of [2] by using the techniques of [1]. Specifically, the first step of the algorithm of [2] can be described as choosing a (small) random subset of the metric space s.t. each point u is chosen with probability proportional to $\sum(u)=\sum_v d(u,v)$; computing the latter quantities is the time bottleneck of the algorithm. By using the randomized comparator of [1], the values of $\sum(u)$ can be approximately computed for al u in time $n\text{poly}(\log n)$ (instead of $\Omega(n^2)$).

References

[1] P. Indyk, "Sublinear time algorithms for metric space problems", *STOC'99*.

[2] W. F. de la Vega, C. Kenyon, "A randomized approximation scheme for Metric MAX-CUT", *FOCS'98*, pp. 468 - 471.

[3] V. Kann, S. Khanna, J. Lagergren and A. Panconesi, "On the Hardness of Max k-Cut and its Dual", Proc. *5th Israel Symposium on Theory and Computing Systems (ISTCS)*, 1996, pp. 61-67.

[4] N. Guttman-Beck, R. Hassin, "Approximation algorithms for min-sum p clustering", *Discrete Applied Mathematics*.

[5] N. Garg, V.V. Vazirani, M. Yannakakis, "Approximate max-flow min-(multi)-cut theorems and their applications", *SICOMP* (25), 1995, pp. 235-251.

[6] L. Schulman, "Clustering for Edge-Cost Minimization", manuscript.

[7] S. Sahni, T. Gonzalez, "P-complete Approximation Problems", *JACM* (23), 1976, pp. 555-566.