Approximating min-max k-clustering

Asaf Levin*

July 24, 2007

Abstract

We consider the problems of set partitioning into k clusters with minimum total cost and minimum of the maximum cost of a cluster. The cost function is given by an oracle, and we assume that it satisfies some natural structural constraints. That is, we assume that the cost function is monotone, the cost of a singleton is zero, and we assume that for all $S \cap S' \neq \emptyset$ the following holds $c(S) + c(S') \geq c(S \cup S')$. For this problem we present a $(2k-1)$-approximation algorithm for $k \geq 3$, a 2-approximation algorithm for $k = 2$, and we also show a lower bound of k on the performance guarantee of any polynomial-time algorithm.

We then consider special cases of this problem arising in vehicle routing problems, and present improved results.

1 Introduction

We are given a ground set E with a cost function, c, defined over all subsets of E. Assume that c satisfies the following properties:

1. $c(\{i\}) = 0$ for all $i \in E$.
2. $c(S) \geq c(S')$ for all $S' \subseteq S$. That is, c is a monotone cost function.
3. If $S \cap S' \neq \emptyset$, then $c(S) + c(S') \geq c(S \cup S')$.

The k MIN-MAX problem is to find a partition of E into k (disjoint) subsets $S_1, S_2, \ldots S_k$ so that $\max_i c(S_i)$ is minimized. The k MIN-SUM problem is to find a partition of E into k (disjoint) subsets $S_1, S_2, \ldots S_k$ so that $\sum_{i=1}^{k} c(S_i)$ is minimized.

We assume that c is given as an oracle that evaluates the value of c for a given subset of E in $O(1)$ time, and we are interested in polynomial-time (in size of E) algorithms.

The sets $S_1, S_2, \ldots, S_k$ (of a feasible solution) are called the *blocks of the partition* or *clusters*. For a partition $\mathcal{P}$ we denote its cost by $c(\mathcal{P})$ (where the definition of the cost of the solution depends on the problem we consider). For a partition of E into $E_1, E_2, \ldots, E_k$ a *refinement* is a partition of E into k' subsets $E'_1, E'_2, \ldots, E'_{k'}$ such that for all $1 \leq i \leq k'$ there exists $j(i)$ such that $E'_i \subseteq E_{j(i)}$.

In recent years the study of clustering problems has increased dramatically. In most problems that were investigated in the literature the goal is either to minimize the maximum cost of a cluster or to minimize the sum of costs of the clusters. This leaves the definition of a *cost of a cluster* different for different problems. To obtain a polynomial representation of the problem, the cost function is usually given in a compact form. In this paper we assume that the cost function satisfies Properties 1, 2 and 3, and we investigate the optimization problems that result from such cost structure. The assumption that the cost function is given by an

*Department of Statistics, The Hebrew University, Jerusalem 91905, Israel. **E-mail:** levinas@mscc.huji.ac.il

oracle replaces the compact form of the cost function (that is usually assumed) and is particularly interesting in cases in which the cost of a cluster is evaluated by solving an NP-hard problem or performing a simulation study of some engineering problem (so in these cases it is unclear how to obtain a compact representation of the cost function). For example, in the well-known vehicle routing problem a cluster is a subset of the vertex set to be served by a common vehicle, and the cost of a cluster is the optimal solution of the traveling salesperson problem over this vertex set. Given such a heavy computational procedure to evaluate the cost of a cluster, it is clear that we are interested in trying to minimize the number of such procedure calls.

The k CLUSTERING PROBLEM (kCluster) is defined as follows: Given a complete undirected graph $G = (V, A)$ where the edges are endowed with a metric length function $\ell : A \rightarrow R^+$, the goal is to partition the vertex set into k sets $V_1, V_2, \ldots, V_k$ so as to minimize the following objective function $\max_i \max_{u,v \in V_i} \ell(u, v)$. We next note that the kCluster is a special case of the k min-max problem. To see this last claim note that we can define a cost function $c(S)$, where $S \subseteq V$, to be $\max_{u,v \in S} \ell(v, u)$. To see that the resulting problem is an instance of the k min-max problem, we need to show that c satisfies the requested properties. Properties 1 and 2 are trivially satisfied. To see that Property 3 is also satisfied, consider a pair of non-disjoint sets S_1, S_2, and assume that $v \in S_1 \cap S_2$. Fix arbitrary vertices $u, w \in S_1 \cup S_2$. If $u, w \in S_1$ or $u, w \in S_2$, then clearly $\ell(u, w) \leq c(S_1) + c(S_2)$. If $u \in S_1$ and $w \in S_2$, then by the triangle inequality $\ell(u, w) \leq \ell(u, v) + \ell(v, w) \leq c(S_1) + c(S_2)$. Therefore, for all pairs $u, w \in S_1 \cup S_2$ we conclude that $\ell(u, w) \leq c(S_1) + c(S_2)$. Property 3 follows by taking the maximum over all pairs u and w. Gonzales [2] and Hochbaum and Shmoys [5] showed that the kCluster problem has a 2-approximation algorithm. Hsu and Nemhauser [6] and Hochbaum [4] showed that kCluster cannot be approximated within a factor better than 2 (assuming $P \neq NP$). Gonzales [2] showed that the same lower bound on the approximability of the kCluster applies also to the special case where the input is restricted to points in 3-dimensional Euclidean space. Feder and Greene [1] showed that the kCluster problem where the input is restricted to be a set of points in 2-dimensional Euclidean space is not approximable within 1.969. As we noted above the k min-max problem generalizes the kCluster, and therefore it is NP-hard for all $k \geq 3$.

We then consider special cases of this problem arising in vehicle routing problems, and present improved results. In particular we consider the following problem. The MIN-MAX RURAL POSTMEN COVER PROBLEM is defined as follows. The input is a complete graph $G = (N, E)$, a length function $l : E \rightarrow R_+$, a subset $E' \subseteq E$, and an integer number $k > 0$. The output is a set of k paths $P_1, \ldots, P_k$, such that $E' \subseteq \cup P_i$. The goal is to minimize the maximum length of a path in the solution, that is, to minimize $\max_i l(P_i)$. We note that this problem is a special case of the min-max k-clustering problem. However, using the special structure of this problem, we are able to present a 7-approximation algorithm for this problem (so the approximation ratio is independent of k).

Paper outline. In Section 2 we provide an approximation algorithm for the k min-max problem whose approximation ratio is 2 for $k = 2$ and $((k-1) \cdot \alpha + 1)$ for all $k \geq 3$ where $\alpha = 2$ denotes the approximation ratio of an approximation algorithm for the kCluster problem, and we conclude this section by showing that any polynomial-time algorithm cannot guarantee an approximation ratio that is better than k for the k min-max problem.

2 The k min-max problem

In this section, we first provide an approximation algorithm and analyze its performance guarantee. Our approximation algorithm is a 2-approximation if $k = 2$, and a $((k-1)\alpha + 1)$-approximation algorithm where α is the approximation ratio of any approximation algorithm for the kCluster problem. Afterwards, we will show a lower bound of k on the performance guarantee of any polynomial-time algorithm for the k min-max problem.

Recall that for $k = 2$ there is a polynomial-time algorithm that solves (optimally) the kCluster problem.

This algorithm is based on guessing the optimal cost (that is one out of $O(|A|)$ values that are known in advance), and then defining an expensive edge to be an edge whose cost is greater than the current guess. The current guess is not smaller than the optimal cost if and only if the graph of the expensive edges is bipartite. This results an $O(|A| \log |A|)$ time algorithm for the 2Cluster problem (by using a binary search on the optimal cost guess value). In this case where $k = 2$ we denote $\alpha = 1$. For higher values of k, one can show [6, 4] a similar reduction from k-coloring of a simple graph that shows that approximating the kCluster within a factor $2 - \varepsilon$ is NP-hard (for all $\varepsilon > 0$). That is, given an input graph $G_c = (V_c, E_c)$ for the k-coloring problem we let $G = (V_c, E)$ to be a complete graph with edge cost $c(\{i, j\}) = 1$ if and only if $(i, j) \in E_c$ and otherwise $c(\{i, j\}) = 2$. Then the kCluster instance has a solution with cost one if and only if G_c is k-colorable, and otherwise the optimal solution to the kCluster instance costs two. For the kCluster problem Gonzales [2] and Hochbaum and Shmoys [5] provided 2-approximation algorithms (for all values of k). Using the results of [2, 5], when $k \geq 3$ we denote $\alpha = 2$. So α is either 1 or 2 depending on the value of k.

Algorithm 1

1. *Compute a cost function c' defined as $c'(S) = \max_{i,j \in S}\{c(\{i, j\})\}$.*
2. *Using an α-approximation algorithm for the kCluster problem, find a partition of E into k subsets such that its cost with respect to c' is minimized.*

Theorem 2 *Algorithm 1 is a $((k - 1) \cdot \alpha + 1)$-approximation algorithm.*

Proof. Note that by monotonicity of the cost function, we conclude that for every $S \subseteq E$, $c'(S) \leq c(S)$. Moreover, for i, j and k we note that $c'(\{i, j\}) + c'(\{j, k\}) = c(\{i, j\}) + c(\{j, k\}) \geq c(\{i, j, k\}) \geq c(\{i, k\}) = c'(\{i, k\})$ where the first inequality holds by property 3 and the second inequality holds by property 2, and therefore there is an α-approximation algorithm for the kCluster instance that we create in step 2 (i.e., the metric assumption holds for c'). Denote by OPT the optimal solution with respect to c, and by APX the solution of Algorithm 1. Assume that APX $\neq$ OPT.

Let G' be a graph defined as follow: For every cluster in OPT there is a vertex in G' and for every pair I and J of clusters of OPT, there is an edge in G' between their vertices if and only if there is a cluster C in APX such that both $I \cap C \neq \emptyset$ and $J \cap C \neq \emptyset$.

For every edge (i, j) in G' representing the clusters I and J in OPT let $v_i \in I \cap C$ and $v_j \in J \cap C$ where C is a cluster in APX that causes this edge to exist. Then by Property 3 of c, the following is satisfied:

$$c(I) + c(J) + c(\{v_i, v_j\}) \geq c(I) + c(J \cup \{v_i, v_j\}) \geq c(I \cup J)$$

Note that $c(\{v_i, v_j\}) \leq c'(\text{APX})$, but APX is an α-approximation algorithm with respect to c', and therefore $c'(\text{APX}) \leq \alpha \cdot c'(\text{OPT}) \leq \alpha \cdot c(\text{OPT})$. For each connected component $\mathcal{C}$ of G', we union all the clusters that correspond to vertices in $\mathcal{C}$. Denote by p the number of connected components of G. Finally, we receive a partition of E into p sets that costs at most $c(\text{OPT}) + (k - p) \cdot \alpha \cdot c(\text{OPT}) \leq ((k - 1) \cdot \alpha + 1) \cdot c(\text{OPT})$. But APX is a refinement of the resulting one, and therefore costs less. Therefore, $c(\text{APX}) \leq ((k - 1) \cdot \alpha + 1) \cdot c(\text{OPT})$. ■

We next show that our algorithm is best possible up to a constant factor.

Theorem 3 *No polynomial-time algorithm for the k min-max problem has an approximation ratio better than k.*

Proof. Consider a ground set E of size n where $n = pk^2$ and p is an arbitrary large integer to be selected afterwards. Consider a partition of E into k equal size sets $E_1, E_2, \ldots, E_k$. For a set $S \subseteq E$ denote by $n(S) = |\{i : E_i \cap S \neq \emptyset\}|$. We define a cost function c as follows:

$$c(S) = \begin{cases} 0 & \text{if } |S| = 1 \\ \min\left\{n(S), \left\lceil \frac{|S|}{p} \right\rceil\right\} & \text{otherwise.} \end{cases}$$

We next show that c satisfies properties 1, 2 and 3. For $i \in E$, $c(\{i\}) = 0$ because $|\{i\}| = 1$, and therefore property 1 holds. Let $S' \subseteq S$, then $n(S') \leq n(S)$ as $E_i \cap S' \neq \emptyset$ implies that $E_i \cap S \neq \emptyset$. Moreover, $\frac{|S'|}{p} \leq \frac{|S|}{p}$, and therefore $\left\lceil \frac{|S'|}{p} \right\rceil \leq \left\lceil \frac{|S|}{p} \right\rceil$. Therefore, for $S' \subseteq S$, $c(S') \leq c(S)$, and hence property 2 holds. It remains to consider property 3. Let $S, S' \subseteq E$ such that $S \cap S' \neq \emptyset$. If $|S| = 1$ (and similarly if $|S'| = 1$), then the property clearly holds as both sides of the inequality equal to $c(S')$. It remains to consider the case where $|S| \geq 2$ and $|S'| \geq 2$. Note that $n(S \cup S') \leq n(S) + n(S')$ as if $(S \cup S') \cap E_i \neq \emptyset$, then either $S \cap E_i \neq \emptyset$ or $S' \cap E_i \neq \emptyset$ (or both). Moreover, $\frac{|S \cup S'|}{p} \leq \frac{|S|}{p} + \frac{|S'|}{p}$, and therefore $\left\lceil \frac{|S \cup S'|}{p} \right\rceil \leq \left\lceil \frac{|S|}{p} \right\rceil + \left\lceil \frac{|S'|}{p} \right\rceil$. Therefore, $c(S \cup S') \leq c(S) + c(S')$, and property 3 also holds.

The optimal solution is the partition of E into $E_1, E_2, \ldots, E_k$. By definition $n(E_i) = 1$ for all i, and therefore the optimal solution has a unit cost. In order to prove the theorem it suffices to show that any polynomial-time algorithm cannot guarantee a solution whose cost is at most $k - 1$. Assume otherwise; then the algorithm must identify (in polynomial time) a set S such that $c(S) \leq k - 1$ and $|S| \geq pk$ (this is so as the set with the largest cardinality in the returned solution has at least pk elements). Consider this set S; then since $|S| \geq pk$, we conclude that $c(S) = n(S)$. Since the algorithm is not aware of the optimal partition, it must identify a set whose size is at least $p + 1$ and that intersects at most $k - 1$ blocks of the optimal partition.

Now assume that the optimal partition is selected randomly. Then the probability that a given set S will intersect at most $k-1$ blocks of the partition is at most $k \cdot \left(1 - \frac{1}{k}\right)^p$. When an algorithm queries a set S' and finds out that $n(S') = k$, it learns only that any set that contains S' has a value of n that equals k. However, for all remaining sets the algorithm does not gain new information. Therefore, the expected number of steps until such S will be found is at least $\frac{1}{k \cdot \left(1 - \frac{1}{k}\right)^p}$. Note that this last bound is also a lower bound on the number of steps that the assumed algorithm performs. However, $\frac{1}{k \cdot \left(1 - \frac{1}{k}\right)^p} \geq \frac{e^{p/k}}{k}$ where the last inequality holds because $\frac{1}{\left(1 - \frac{1}{k}\right)^k} \geq e$ for all $k \geq 2$. Since p can be arbitrary large (significantly larger than k), the last lower bound on the number of steps that the algorithm performs is exponential in the size of E, and this is a contradiction to the assumption that the algorithm performs only a polynomial number of steps. ■

References

[1] T. Feder and D.H. Greene, "Optimal algorithms for approximate clustering," *Proceedings of the 20th Annual ACM Symposyum on Theory of Computing (STOC 1988)*, 434-444, 1988.

[2] T.F. Gonzalez, "Clustering to minimize the maximum intercluster distance", *Theoretical Computer Science*, **38**, 293-306, 1985.

[3] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical Programming*, **79**, 191-215, 1997.

[4] D.S. Hochbaum, "When are NP-hard location problems easy?", *Annalls of Operations Research*, **1**, 201-214, 1984.

[5] D.S. Hochbaum and D.B. Shmoys, "A unified approach to approximation algorithms for bottleneck problems," *Journal of the ACM*, **33**, 533-550, 1986.

[6] W.L. Hsu, and G.L. Nemhauser, "Easy and hard bottleneck location problems," *Discrete Applied Mathematics*, **1**, 209-216, 1979.

[7] C. L. Monma and S. Suri, "Partitioning points and graphs to minimize the maximum or the sum of diameters," in *Graph Theory, Combinatorics and Applications*, 880-912, Alavi et al. (editors), Wiley, 1991.