A generalized minimum cost \(k\)-clustering

Asaf Levin*

July 19, 2009

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 the problem of minimizing the maximum cost of a cluster 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. For the problem of minimizing the total cost of all the clusters, we present a 2-approximation algorithm for the case where \(k\) is a fixed constant, a \((4k - 3)\)-approximation where \(k\) is unbounded, and we show a lower bound of 2 on the approximation ratio of any polynomial time algorithm. Our lower bounds do not depend on the common assumption that \(P \neq NP\).

Computing Classification System Terms: F.2.2 Nonnumerical Algorithms and Problems; G.2.1 Combinatorics.

General Terms: Algorithms; Theory.

Additional keywords: clustering, oracle cost functions, approximation algorithms.

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 \(S \cap S' = \emptyset\), 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.

*Chaya fellow. Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel. E-mail: levinas@ie.technion.ac.il
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 $P$ we denote its cost by $c(P)$ (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 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 ($k$Cluster) is defined as follows: Given a complete undirected graph $G = (V, A)$ where the edges are endowed with a metric length function $\ell : A \rightarrow R^+\ast$, 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 $k$Cluster 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 [4] and Hochbaum and Shmoys [7] showed that the $k$Cluster problem has a 2-approximation algorithm. Hsu and Nemhauser [8] and Hochbaum [6] showed that $k$Cluster cannot be approximated within a factor better than 2 (assuming $P \neq NP$). Gonzales [4] showed that the same lower bound on the approximability of the $k$Cluster applies also to the special case where the input is restricted to points in 3-dimensional Euclidean space. Feder and Greene [3] showed that the $k$Cluster 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 $k$Cluster, and therefore it is NP-hard for all $k \geq 3$.

The $k$ CLUSTERING PROBLEM TO MINIMIZE THE SUM OF DIAMETERS ($k$SumDiam) is defined as follows: Given a complete undirected graph $G = (V, A)$ where the edges are endowed with a metric length function $\ell : A \rightarrow R^+\ast$, the goal is to partition the vertex set into $k$ sets $V_1, V_2, \ldots, V_k$ so as to minimize $\sum_i \max_{u, v \in V_i} \ell(u, v)$. Since the cost of a cluster function is the same as the cost function defined above for the $k$Cluster problem, we conclude that $k$SumDiam is a special case of the $k$ min-sum problem. The $k$SumDiam problem was suggested in the literature as an alternative to the $k$Cluster problem in order to avoid the so called dissection effect (see [5, 9]). Doddi et al. [2] presented a bicriteria approximation algorithm for this problem. That is, their algorithm returns a solution whose cost is at most $O(\log n)$ times the optimal cost and the returned solution partitions $V$ into a constant-times-$k$ clusters (i.e., a constant blowup in the number of clusters). They also showed a 2-approximation algorithm for the case where the num-
ber of blocks in the partition, \( k \), is a constant. Moreover, \([2]\) proves that the 2SumDiam is polynomially solvable. On the negative side they proved that unless \( P = NP \) there is no \((2−\varepsilon)\)-approximation algorithm for all \( \varepsilon > 0 \). Charikar and Panigrahy \([1]\) presented a 2-approximation algorithm based on the primal-dual framework for the related problem of partitioning the set \( V \) into \( k \) subsets \( V_1, V_2, \ldots, V_k \) and picking for each subset a representative \( v_i \in V_i \) where the goal is to minimize the sum of radii, i.e., to minimize \( \sum_{i=1}^{k} \max_{u \in V_i} \ell(u, v_i) \). Using the triangle inequality they noted that this last algorithm gives a 4-approximation algorithm for kSumDiam. So this last result improves the result of \([2]\) for non-constant values of \( k \). The \( k \) min-sum problem generalizes the \( k \)SumDiam problem, and therefore it is NP-hard as well for all values of \( k \) such that \( k \geq 3 \).

**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 \((k−1)\alpha + 1\) for all \( k \geq 3 \) where \( \alpha = 2 \) denotes the approximation ratio of an approximation algorithm for the \( k \)Cluster 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. In Section 3 we study the \( k \) min-sum problem. We first consider the case where \( k = 2 \) is a fixed constant, and we present a 2-approximation algorithm for this case of the \( k \) min-sum problem. Afterwards, we turn to the general case where the value of \( k \) is unbounded, and we present a \((4k−3)\)-approximation algorithm for this case. We conclude this section by showing that any polynomial-time algorithm cannot guarantee an approximation ratio strictly better than 2 for the \( k \) min-sum problem. Our lower bounds (for both problems) do not depend on the assumption that \( P \neq NP \). We conclude the paper with a short discussion on a possible direction for future research.

## 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 \( \alpha \) is the approximation ratio of any approximation algorithm for the \( k \)Cluster 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 \( k \)Cluster 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 \([8, 6]\) a similar reduction from \( k \)-coloring of a simple graph that shows that approximating the \( k \)Cluster 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 \( k \)Cluster instance has a solution with cost one if and only if \( G_c \) is \( k \)-colorable, and otherwise the optimal solution to the \( k \)Cluster instance costs two. For the \( k \)Cluster problem Gonzales \([4]\) and Hochbaum and Shmoys \([7]\) provided 2-approximation algorithms (for all values of \( k \)). Using the results of \([4, 7]\), when \( k \geq 3 \) we denote \( \alpha = 2 \). So \( \alpha \) 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 $\alpha$-approximation algorithm for the $k$-Cluster 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 $c'({i,j}) + c'(\{j,k\}) = c({i,j}) + c(j,k) \geq c({i,j,k}) \geq c'(\{i,j,k\})$ where the first inequality holds by property 3 and the second inequality holds by property 2, and therefore there is an $\alpha$-approximation algorithm for the $k$-Cluster 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 $c({v_i,v_j}) \leq c'(APX)$, but $APX$ is an $\alpha$-approximation algorithm with respect to $c'$, and therefore $c'(APX) \leq \alpha \cdot c'(OPT) \leq \alpha \cdot c(OPT)$. For each connected component $C$ of $G'$, we union all the clusters that correspond to vertices in $C$. Denote by $p$ the number of connected components of $G$. Finally, we receive a partition of $E$ into $p$ sets that costs at most $c(OPT) + (k-p) \cdot \alpha \cdot c(OPT) \leq ((k-1) \cdot \alpha + 1) \cdot c(OPT)$. But $APX$ is a refinement of the resulting one, and therefore costs less. Therefore, $c(APX) \leq ((k-1) \cdot \alpha + 1) \cdot c(OPT)$. ■

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 $E_i \cap S \neq \emptyset$. 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 \( \frac{1}{k^p} \). 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 \( \frac{1}{k^p} \). Note that this last bound is also a lower bound on the number of steps that the assumed algorithm performs. However, \( \frac{1}{k^p} \geq e^{p/k} \) where the last inequality holds because \( \frac{1}{1-\frac{1}{k}} \geq e \) 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.

3 The \( k \) min-sum problem

In this section we explore the problem of partitioning a set into \( k \) subsets with a minimum total cost. We first consider the case where \( k \) is a fixed constant, and we present a \( 2 \)-approximation algorithm for this case. Afterwards, we turn to the general case where the value of \( k \) is unbounded, and we present a \( (4k - 3) \)-approximation algorithm for this case. We conclude this section by showing that any polynomial-time algorithm cannot guarantee an approximation ratio strictly better than \( 2 \). Our lower bound improves the lower bound of Doddi et al. [2] in two aspects:

- It applies also for the case \( k = 2 \) in which the \( 2 \)SumDiam is polynomially solvable whereas the \( 2 \) min-sum problem cannot be approximated within a factor better than \( 2 \).
- Our lower bound does not depend on the assumption that \( P \neq NP \).

3.1 Fixed values of \( k \)

For the \( k \) min-max problem we were able to show a lower bound of \( k \) on the approximation ratio of any polynomial-time algorithm. A similar lower bound cannot be shown for the \( k \) min-sum problem, as for constant values of \( k \) we are able to present a \( 2 \)-approximation algorithm such as the following one:

Algorithm 4

1. For each block in the optimal partition \( V_i \) guess a representative member of this block \( r_i \in V_i \) and the maximum radius of \( V_i \) with respect to \( r_i \). That is, for all \( i \), we guess \( r_i \) and \( R_i = \max_{u \in V_i} c(\{u, r_i\}) \).

2. Construct the partition \( U_1, \ldots, U_k \) as follows:
   (a) Initialize: For all \( i = 1, 2, \ldots, k \), set \( U_i = \{r_i\} \).
   (b) For each \( u \in E \setminus \{r_1, r_2, \ldots, r_k\} \), denote by \( S_u = \{i : c(\{u, r_i\}) \leq R_i \} \). Add \( u \) to the set \( U_i \) such that \( R_i = \min_{k \in S_u} R_k \) if the minimum is over a non-empty set of values.
Algorithm 4 performs a guessing step of some information about the optimal solution. This guessing step is performed by using exhaustive enumeration over all possibilities and picking the cheapest feasible solution returned by the algorithm (among all the possible guesses). In the analysis it suffices to consider the iteration of the exhaustive enumeration in which the algorithm guesses the correct values of a fixed optimal solution that we denote by $OPT$.

We next note that for the special case of the $k$SumDiam problem, Algorithm 4 is a 2-approximation, as it produces an optimal solution to the related problem of minimizing the sum of clusters’ radii. However, we need to pick for each item $u$ the set $U_i$ such that $i$ is the minimizer of $\min_{k \in S_u} R_k$. Other selection rules, although producing a 2-approximation algorithm for the $k$SumDiam, do not allow our proof for the general case to work out.

**Theorem 5** Algorithm 4 is a 2-approximation algorithm for the $k$ min-sum problem when $k$ is a fixed constant.

**Proof.** The algorithm runs in polynomial time as the exhaustive enumeration needs to check at most $n^{2k}$ different values of the guess, and each such guess is checked in polynomial time. The algorithm returns a feasible solution as for the correct set of guesses we know that each element $u \in E$ such that $u \in V_i$ satisfies $c(\{u, r_i\}) \leq R_i$. Therefore, $S_u \neq \emptyset$ for all $u$, and hence for this guess we return a partition of $E$ (i.e., a feasible solution). It remains to prove the approximation ratio of the algorithm.

Denote by $APX$ the solution of Algorithm 4. Assume that $APX \neq OPT$. Our proof is similar to the proof of Theorem 2.

Let $G'$ be a graph defined as follow: For every cluster in $OPT$ there is a vertex in $G'$ and for every pair $V_i$ and $V_{i'}$ of clusters of $OPT$, there is an edge in $G'$ between their vertices if and only if there is a cluster $U_j$ in $APX$ such that both $r_i \in U_j$ and $V_{i'} \cap U_j \neq \emptyset$ or vice versa. For such a pair of adjacent vertices in $G'$, assume w.l.o.g. that $r_i \in U_j$, and let $w \in V_{i'} \cap U_j$. We now argue that

$$c(\{r_i, w\}) \leq \min\{R_i, R_{i'}\}. \quad (1)$$

This is so because by the fact that the algorithm adds $w$ to $U_j$, we conclude that $c(\{w, r_i\}) \leq R_i$. By the fact that $w \in V_{i'}$ and using the fact that $R_{i'}$ is the correct value, we conclude that $c(\{w, r_{i'}\}) \leq R_{i'}$. Since our algorithm decides to add $w$ to the set that contains $r_i$ and not to the set that contains $r_{i'}$, we conclude that $R_i \leq R_{i'}$, and thus we establish (1).

Therefore, by (1) and Property 3 of $c$, we conclude that for every edge $(i, i')$ in $G'$ representing the clusters $V_i$ and $V_{i'}$ of $OPT$, the following is satisfied:

$$c(V_i \cup V_{i'}) \leq c(V_i) + c(V_{i'}) + \min\{R_i, R_{i'}\}. \quad (2)$$

For each connected component $C$ of $G'$ we apply the following. We consider only edges of a spanning tree of $C$. We root this tree in an arbitrary vertex of $G'$. By (2), we are able to union each vertex $i$ of this component to its parent in the tree by paying at most another cost of $R_i$. So as each vertex in $C$ pays for at most one edge (of the spanning tree), we conclude that the following inequality holds:

$$c \left( \bigcup_{i \in C} V_i \right) \leq \sum_{i \in C} (c(V_i) + R_i) \leq 2 \sum_{i \in C} c(V_i), \quad (3)$$

\[ \text{Return } U_1, U_2, \ldots, U_k \text{ if } \bigcup_{i=1}^k U_i = E \text{ and otherwise return that the current guess is infeasible.} \]
Algorithm 6

1. Compute a cost function \( c' \) defined as \( c'(S) = \max_{i,j \in S} \{c(\{i, j\})\} \).

2. Find a partition of \( E \) into \( k \) subsets such that its cost with respect to \( c' \) is minimized using a \( \beta \)-approximation algorithm for the \( k \)\text{SumDiam} problem.

The proof of the next theorem is similar to the proof of Theorem 2.

**Theorem 7** Algorithm 6 is a \( ((k - 1) \cdot \beta + 1) \)-approximation algorithm for the \( k \) min-sum problem.

**Proof.** Note that \( c' \leq c \). Denote by \( \OPT \) the optimal solution with respect to \( c \), and by \( \APX \) the solution of Algorithm 6. 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 of clusters of \( \OPT \), \( I \) and \( J \), 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 the following is satisfied:

\[
c(I) + c(J) + c(\{v_i, v_j\}) \geq c(I) + c(J \cup \{v_i, v_j\}) \geq c(I \cup J \cup \{v_i, v_j\}) = c(I \cup J).
\]

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

**Proposition 8** The approximation ratio of Algorithm 6 is at least \( k \).
Proof. We construct a bad example for the algorithm. Consider a set $E$ composed of $k + 1$ disjoint sets $S_i$ for all $i \leq k + 1$, such that $|S_i| = 2$, $\forall i \leq k - 1$ and $|S_k| = |S_{k+1}| = 1$. Define the cost function $c$ as $c(S) = |\{i | S_i \cap S \neq \emptyset\}| - 1$. For this cost function every solution with respect to $c'$ must cost at least one, and therefore the solution $S^1, S^2, \ldots, S^k$ is optimal (with respect to $c'$) where $S^k$ is a set such that $|\{S^k \cap S_i\}| = 1$, $\forall i$ and $S^i = S_i \setminus S^k$ $\forall i \leq k - 1$. This solution costs $k$ with respect to $c$ whereas the solution $S_1, S_2, \ldots, S_{k-1}, S_k \cup S_{k+1}$ costs $1$. Therefore, the ratio obtained by Algorithm 6 in this example is $k$. ■

### 3.3 Lower bound for the $k$ min sum problem

We first show that the $2$ min-sum problem is hard to approximate as shown by the following theorem (this is different from the case of 2SumDiam that is polynomially solvable):

**Theorem 9** No polynomial-time algorithm for the $2$ min-sum problem has an approximation ratio better than $2$.

**Proof.** Consider a ground set $E$ of size $n$ where $n = 8p$ and $p$ is an integer. Denote by $E'$ a subset of $E$ of size $\frac{n}{2}$. Let $c$ be defined as follows:

$$
c(S) = \begin{cases} 
0 & \text{if } |S| = 1 \\
1 & \text{if } 2 \leq |S| \leq p \text{ or } S \subseteq E' \text{ or } S \subseteq E \setminus E' \\
2 & \text{if } p + 1 \leq |S| \leq 6p \text{ and } S \not\subseteq E' \text{ and } S \not\subseteq E \setminus E' \\
3 & \text{if } 6p + 1 \leq |S| \leq 8p - 2 \\
4 & \text{otherwise.}
\end{cases}
$$

Consider the (feasible) partition into the sets $E'$ and $E \setminus E'$. Each of these sets costs $1$, and therefore the optimal solution costs at most $2$. It suffices to show that any polynomial-time algorithm cannot guarantee a partition of cost at most $3$. Assume otherwise that there exists a polynomial-time algorithm that always returns a partition of cost at most three. In order to obtain a partition of cost at most three, the algorithm must find a set $S$ such that $p + 1 \leq |S|$ and $c(S) = 1$. This is so because the solution returned by the algorithm must include a set and its complement where at least one of these has a unit cost and the other set has cost at most $2$; therefore the two returned sets must have cardinality at least $2p$ and at most $6p$.

Now assume that we select $E'$ randomly. Then the probability of finding a set $S$ such that $p + 1 \leq |S| \leq 6p$ and $c(S) = 1$ is at most $2 \cdot \frac{(n \cdot 2)^p}{\binom{2p}{n}} \leq 2 \cdot \binom{2p}{n} \leq 2^{p+1}$. Note that given a query of a set $S$ such that $p + 1 \leq |S| \leq 6p$ and $c(S) = 2$, we do not gain any information about other sets except the ones that contain $S$. Therefore, the expected number of sets $S$ that must be queried until we find a set $S$ such that $p + 1 \leq |S| \leq 6p$ and $c(S) = 1$ is at least $2^{p-1} = 2^{n/8-1}$ and this bound is exponential in $n$. ■

We next argue that the lower bound of $2$ can be extended to all values of $k$. To see this last claim note that given the instance in the proof of Theorem 9, we can add another $k - 2$ dummy elements such that each set that contains at least two elements from which at least one element is a dummy element has an infinite cost. Then any finite cost solution must place each dummy element in its own cluster, and we are left to partition the original set of elements into two sets so that the sum of their costs is minimized. By the proof of Theorem 9, we conclude the following:

**Theorem 10** No polynomial-time algorithm for the $k$ min-sum problem has an approximation ratio better than $2$. 

8
Recall that the lower bound that we obtained for approximating the $k$ min-max problem is $k$, and for the $k$ min-sum problem the lower bound is 2 for all values of $k$. Note that for a fixed value of $k$ this lower bound is tight as we showed a 2-approximation algorithm for this case.

4 Concluding remarks

In this paper we study a family of clustering problems defined by a given set of properties that are satisfied by the cost of a cluster function $c$.

One can try to address the similar problems for other families of clustering problems defined using alternative properties of $c$. For example, an alternative property that seems interesting to study is the case where $c$ is a non-negative submodular set function. The study of such families of optimization problem is left for future research.

References


