

Approximation algorithms for min-sum p -clustering

Nili Guttman-Beck, Refael Hassin*

Department of Statistics and Operations Research, Tel Aviv University, Ramat Aviv,
Tel Aviv 69978, Israel

Received 7 January 1997; received in revised form 4 April 1998

Abstract

We consider the following problem: Given a graph with edge lengths satisfying the triangle inequality, partition its node set into p subsets, minimizing the total length of edges whose two ends are in the same subset. For this problem we present an approximation algorithm which comes to at most twice the optimal value. For clustering into two equal-sized sets, the exact bound on the maximum possible error ratio of our algorithm is between 1.686 and 1.7. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Approximation algorithm; p -clustering

1. Introduction

Let $G=(V,E)$ be a given complete undirected graph, with node set V and edge set E . The edges in E have lengths that satisfy the triangle inequality. The *min-sum p -clustering problem* requires to partition V into p subsets, possibly of given sizes, minimizing the total length of edges whose two ends are in the same subset. The problem has numerous applications in various areas (see, for example, [2–5]).

In this paper we derive an approximation algorithm for a version of the problem in which the sizes of the clusters are given. Our results also apply when the sizes are not given, only their number, giving the same error bound but with higher complexity.

The problem is known to be NP-hard for $p \geq 2$. Sahni and Gonzalez [6] proved that without the triangle inequality assumption for $p > 2$, a polynomial-time algorithm that guarantees a bounded error ratio is not possible unless $P=NP$. Kann et al. [4] strengthened this negative result and showed that there exists a constant α such that no polynomial-time $\alpha|V|^{(2-\epsilon)}$ -approximation algorithm exists for $\epsilon > 0$, unless $P=NP$. The complexity of a more general *G-partitioning problem* is analyzed in [3].

* Corresponding author. E-mail: hassin@math.tau.ac.il.

The *minimum edge deletion bipartition problem* is to find a minimum cost subset $E' \subset E$ such that $(V, E \setminus E')$ is bipartite. This is another version of min-sum 2-clustering. Garg, Vazirani and Yannakakis [1] gave an $O(\log |V|)$ approximation algorithm for this problem, without assuming the triangle inequality.

We consider the problem under the assumption that the distances satisfy the triangle inequality. With this assumption we demonstrate that for any fixed p it is possible to obtain in polynomial time an approximation of at most twice the optimal value. Indeed, the actual factor may be smaller. For the special case of clustering into two equal-sized sets, we prove that the approximate solution is within a factor 1.7 of the optimal. We further show that this bound is not far from the exact one by presenting an example in which the approximation is 1.686 times the optimal value. The example is tedious and therefore we chose to present a simpler one in which the approximate solution is $1\frac{2}{3}$ times the optimal. The stronger bound is presented in an appendix.

The common ways to solve the problem are either through constructive greedy heuristics that sequentially merge two subsets so that the addition to the objective is minimal or through local improvement methods which attempt to improve the solution iteratively by moving objects from one cluster to another. We note that these methods do not guarantee any constant bound on the error ratio. We demonstrate this fact for local search.

We will denote with $l(e)$ the length of edge e , for a subset $E' \subset E$, $l(E) = \sum_{e \in E} l(e)$, and for a subset $V' \subset V$ $l(V')$ is the total cost of the edges in the subgraph induced by V' . Thus, given a set of p positive integers $\{k_i\}_{i=1}^p$ such that $\sum_{i=1}^p k_i = |V| = n$, the *min-sum p -clustering problem* (MCP) is to find a partition of V into disjoint sets $\{P_i\}_{i=1}^p$ such that $\forall i \in \{1, \dots, p\} |P_i| = k_i$, and $\sum_{i=1}^p l(P_i)$ is minimized.

Throughout the paper the optimal value for MCP is denoted by opt and the approximation value for MCP is denoted by apx .

2. The star partitioning problem

The *min star partitioning problem* (SPP) is to find a set of p distinct nodes $\{v_i\}_{i=1}^p \subset V$ and a partition of V into disjoint sets $\{P_i\}_{i=1}^p$ such that $\forall i \in \{1, \dots, p\} |P_i| = k_i$, $v_i \in P_i$ and $\sum_{i=1}^p k_i l(v_i, P_i)$ is minimized. We will use S^* for optimal value of the SPP.

We will show that

- SPP can be solved optimally in polynomial time.
- The cost of an optimum solution for an instance of SPP is at most twice the cost of an optimum solution for the associated MCP.

Combining these 2 facts we establish a polynomial approximation algorithm for the MCP.

Find_partition

input

1. A graph $G = (V, E)$.
2. Constants k_1, \dots, k_p such that $\sum_{i=1}^p k_i = |V|$.

output

1. $\{v_i\}_{i=1}^p \subset V$
2. $P_1, \dots, P_p \subset V, \forall i \in \{1, \dots, p\} v_i \in P_i, |P_i| = k_i$ and $\forall i \neq j P_i \cap P_j = \emptyset$.

begin

for every $\{v_i\}_{i=1}^p \subset V$

Define $\{a_1, \dots, a_{|V|-p}\} = V \setminus \{v_1, \dots, v_p\}$.

Solve the following transportation problem:

$$\text{minimize } \sum_{j=1}^{|V|-p} \sum_{i=1}^p k_i l(a_j, v_i) x_{ji}$$

subject to

$$\begin{aligned} \sum_{i=1}^p x_{ji} &= 1, \\ \sum_{j=1}^{|V|-p} x_{ji} &= k_i - 1, \\ x_{ji} &\in \{0, 1\}. \end{aligned}$$

Let \bar{x} be an optimal solution to this transportation problem.

Define $\forall i \in \{1, \dots, p\} P_i^{\{v_1, \dots, v_p\}} = \{v_i\} \cup \{a_j | 1 \leq j \leq |V| - p \bar{x}_{ji} = 1\}$.

end for

Find $\{v_1^*, \dots, v_p^*\} \subset V$ for which $\sum_{i=1}^p k_i d(v_i, P_i^{\{v_1^*, \dots, v_p^*\}})$ is minimal.

return $(v_1^*, \dots, v_p^*, P_1^{\{v_1^*, \dots, v_p^*\}}, \dots, P_p^{\{v_1^*, \dots, v_p^*\}})$

end Find_partition

Fig. 1. Algorithm Find_Partition.

Theorem 2.1. Procedure Find_partition (see Fig. 1) finds an optimal solution for SPP. It can be executed in time $O(n^{(p+1)})$.

Proof. Let $\bar{v}_1, \dots, \bar{v}_p, \bar{V}_1, \dots, \bar{V}_p$ be an optimal partition for SPP. Since the algorithm checks all the subsets of V of size p it will check the subset $\{\bar{v}_1, \dots, \bar{v}_p\}$ and for this subset find the optimal solution to the transportation problem, so the algorithm will find the optimal solution to the SPP.

For a fixed value of p we can solve the transportation problem in time $O(n)$, using the algorithms given by Tokuyama and Nakano [7]. There are $O(n^p)$ subsets $\{v_1, \dots, v_p\} \subset V$, so altogether the time complexity is $O(n^{(p+1)})$. \square

3. Clustering into p sets

Theorem 3.1. Let P_1, \dots, P_p be the partition offered by *Find-partition*. Then

$$apx = \sum_{i=1}^p l(P_i) \leq 2 \text{opt}.$$

Proof. Let $v_1, \dots, v_p, P_1, \dots, P_p$ be the output of *Find-partition*. Then by Theorem 2.1, $S^* = \sum_{i=1}^p k_i l(v_i, P_i)$.

Let $O_1, \dots, O_p \subset V$ be an optimal partition for MCP, that is $\text{opt} = \sum_{i=1}^p l(O_i)$. Choose $x_i \in O_i$ such that

$$l(x_i, O_i) = \min_{w \in O_i} l(w, O_i).$$

Then,

$$2l(O_i) = \sum_{w \in O_i} l(w, O_i) \geq \sum_{w \in O_i} l(x_i, O_i) = k_i l(x_i, O_i).$$

Therefore,

$$2 \text{opt} = 2 \sum_{i=1}^p l(O_i) \geq \sum_{i=1}^p k_i l(x_i, O_i) \geq S^*, \quad (1)$$

where the last inequality follows from the definition of S^* . On the other hand, using the triangle inequality and the fact that $l(v, v) = 0$ for all $v \in V$,

$$\begin{aligned} 2l(P_i) &= \sum_{p_1 \in P_i} \sum_{p_2 \in P_i} l(p_1, p_2) \\ &\leq \sum_{p_1 \in P_i} \sum_{p_2 \in P_i} (l(p_1, v_i) + l(v_i, p_2)) \\ &= \sum_{p_2 \in P_i} \sum_{p_1 \in P_i} l(v_i, p_1) + \sum_{p_1 \in P_i} \sum_{p_2 \in P_i} l(v_i, p_2) \\ &= |P_i| l(v_i, P_i) + |P_i| l(v_i, P_i) \\ &= 2|P_i| l(v_i, P_i) \\ &= 2k_i l(v_i, P_i). \end{aligned}$$

Therefore,

$$apx = \sum_{i=1}^p l(P_i) \leq \sum_{i=1}^p k_i l(v_i, P_i) = S^*,$$

and by Eq. (1),

$$apx \leq S^* \leq 2 \text{opt}. \quad \square$$

4. Clustering into 2 equal-sized sets

In this section we prove that for the case of partitioning into 2 equal-sized sets the error ratio of our algorithm is no more than 1.7. First, we present a general lemma.

Lemma 4.1. *Let $S, T \subset V$, $S \cap T = \emptyset$, then (assuming the triangle inequality)*

$$|T|l(S) \leq (|S| - 1)l(S, T).$$

Proof. Denote $|S| = k, |T| = m$. According to the triangle inequality, for every $t \in T$

$$\begin{aligned} 2l(S) &= \sum_{s \in S} l(s, S) \\ &= \sum_{s \in S} \sum_{s_1 \in S \setminus \{s\}} l(s, s_1) \leq \sum_{s \in S} \sum_{s_1 \in S \setminus \{s\}} (l(s, t) + l(t, s_1)) \\ &= \sum_{s \in S} ((k - 1)l(s, t) + l(t, S \setminus \{s\})) = \sum_{s \in S} ((k - 2)l(s, t) + l(t, S)) \\ &= (k - 2)l(t, S) + kl(t, S) = 2(k - 1)l(t, S). \end{aligned}$$

Summing over all $t \in T$ we get that

$$2ml(S) \leq 2(k - 1)l(S, T). \quad \square$$

In the following discussion let x, y, P, Q be the solution offered by *Find_partition* and denote by $T^* = l(x, P) + l(y, Q)$. Let O_1, O_2 be an optimal partition. Define $A = P \cap O_1$, $B = P \cap O_2$, $C = Q \cap O_1$ and $D = Q \cap O_2$. So $P = A \cup B$, $Q = C \cup D$, $O_1 = A \cup C$ and $O_2 = B \cup D$. Let $2n$ be the number of nodes in the graph. So $|P| = |Q| = |O_1| = |O_2| = n$. Let $|A| = k$, then $|D| = k$, $|B| = n - k$ and $|C| = n - k$. Without loss of generality, assume that $k \geq n - k$. Fig. 2 demonstrates the relationship between the different sets. Translating Eq. (1) to the notation of this section we obtain

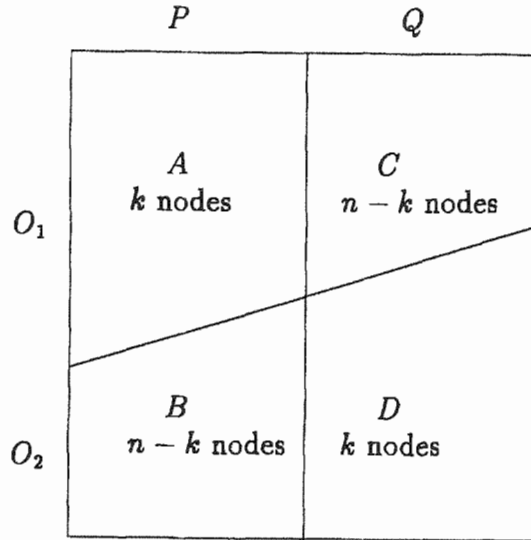
$$2opt \geq n(l(x, P) + l(y, Q)) = nT^*. \tag{2}$$

Now, we show that the inequality can be achieved as the sum of the next 2 inequalities:

Lemma 4.2.

$$2(l(A) + l(D)) + l(A, C) + l(B, D) \geq kT^*,$$

$$2(l(B) + l(C)) + l(A, C) + l(B, D) \geq (n - k)T^*.$$

Fig. 2. The sets A, B, C and D .

Proof.

$$2l(A) = \sum_{a \in A} l(a, A), \quad 2l(D) = \sum_{d \in D} l(d, D),$$

$$l(A, C) = \sum_{a \in A} l(a, C), \quad l(B, D) = \sum_{d \in D} l(d, B).$$

So

$$\begin{aligned} 2(l(A) + l(D)) + l(A, C) + l(B, D) &= \sum_{a \in A} l(a, A \cup C) + \sum_{d \in D} l(d, D \cup B) \\ &= \sum_{a \in A} l(a, O_1) + \sum_{d \in D} l(d, O_2). \end{aligned}$$

Since x, y, P, Q solve SPP

$$\forall a \in A, d \in D \quad l(a, O_1) + l(d, O_2) \geq l(x, P) + l(y, Q).$$

Thus,

$$2(l(A) + l(D)) + l(A, C) + l(B, D) \geq k(l(x, P) + l(y, Q)) = kT^*.$$

Similarly,

$$2(l(B) + l(C)) + l(A, C) + l(B, D) \geq (n - k)(l(x, P) + l(y, Q)) = (n - k)T^*. \quad \square$$

In the next lemmas we will show that $l(A, B) + l(C, D) \leq 3(l(A, C) + l(B, D))$ so when $l(A, C) + l(B, D)$ is small our approximation is good. On the other hand, we will show

that when $l(A, C) + l(B, D) > 0.35 \text{ opt}$ then $l(A, B) + l(C, D) < 0.7 \text{opt} + l(A, C) + l(B, D)$ taking care of the case when $l(A, C) + l(B, D)$ is large.

Lemma 4.3.

$$l(A, B) \leq (n - k)l(x, A) + kl(x, B),$$

$$l(C, D) \leq (n - k)l(y, D) + kl(y, C).$$

Proof. By the triangle inequality

$$\begin{aligned} l(A, B) &= \sum_{a \in A} \sum_{b \in B} l(a, b) \\ &\leq \sum_{a \in A} \sum_{b \in B} (l(a, x) + l(x, b)) \\ &= (n - k) \sum_{a \in A} l(a, x) + k \sum_{b \in B} l(x, b) = (n - k)l(x, A) + kl(x, B). \end{aligned}$$

Similarly,

$$l(C, D) \leq (n - k)l(y, D) + kl(y, C).$$

It follows from the above two lemmas that

Corollary 4.4.

$$l(A, B) + l(C, D) \leq 2(l(A) + l(D)) + l(A, C) + l(B, D) - (2k - n)(l(x, A) + l(y, D)).$$

Proof. It follows from Lemma 4.3 that

$$l(A, B) + l(C, D) \leq k(l(x, B) + l(y, C)) + (n - k)(l(x, A) + l(y, D)).$$

Hence,

$$\begin{aligned} l(A, B) + l(C, D) &\leq k(l(x, B) + l(y, C) + l(x, A) + l(y, D)) - (2k - n)(l(x, A) + l(y, D)) \\ &= kT^* - (2k - n)(l(x, A) + l(y, B)). \end{aligned}$$

Now from Lemma 4.2

$$l(A, B) + l(C, D) \leq 2(l(A) + l(D)) + l(A, C) + l(B, D) - (2k - n)(l(x, A) + l(y, D)).$$

□

The next corollary can be proved in a similar way.

Corollary 4.5.

$$l(A, B) + l(C, D) \leq 2(l(B) + l(C)) + l(A, C) + l(B, D) + (2k - n)(l(x, B) + l(y, C)).$$

Lemma 4.6.

$$l(x, B) + l(y, C) \leq l(x, C) + l(y, B).$$

Proof. Since x, y, P, Q solve SPP $l(x, A) + l(x, B) + l(y, C) + l(y, D) = l(x, P) + l(y, Q)$ is the value of an optimal solution to SPP.

- If $x \in A$ and $y \in D$ (and similarly if $x \in B$ and $y \in C$) then Since $x, y, \{x\} \cup A \cup C, \{y\} \cup B \cup D$ is a possible solution for SPP

$$l(x, A) + l(x, B) + l(y, C) + l(y, D) \leq l(x, A) + l(x, C) + l(y, B) + l(y, D),$$

proving the claim for this case.

- If $x \in B$ and $y \in D$ (and similarly if $x \in A$ and $y \in C$) then for a node $c \in C$ x, y, P', Q' is a possible solution, where $P' = \{x\} \cup A \cup (C \setminus \{c\})$ and $Q' = \{y\} \cup (B \setminus \{x\}) \cup D \cup \{c\}$. Therefore,

$$\begin{aligned} l(x, A) + l(x, B) + l(y, C) + l(y, D) &\leq l(x, A) + l(x, C) - l(x, c) \\ &\quad + l(y, B) - l(y, x) + l(y, D) + l(y, c). \end{aligned}$$

By the triangle inequality $l(y, c) - l(y, x) - l(x, c) \leq 0$, proving the claim for this case as well. \square

Lemma 4.7.

$$l(A, B) + l(C, D) \leq 3(l(A, C) + l(B, D)).$$

Proof. By the triangle inequality

$$kl(x, c) = \sum_{a \in A} l(x, a) \leq \sum_{a \in A} (l(x, a) + l(a, c)) = l(x, A) + l(c, A).$$

Summing over $c \in C$

$$kl(x, C) \leq \sum_{c \in C} (l(x, A) + l(c, A)) = (n - k)l(x, A) + l(A, C).$$

Similarly,

$$kl(y, B) \leq (n - k)l(y, D) + l(B, D).$$

Since x, y, P, Q solve SPP and from Lemma 4.6

$$l(x, B) + l(y, C) \leq l(x, C) + l(y, B),$$

so

$$k(l(x, B) + l(y, C)) \leq (n - k)(l(x, A) + l(y, D)) + l(A, C) + l(B, D).$$

Lemma 4.3 states that

$$l(A, B) + l(C, D) \leq k(l(x, B) + l(y, C)) + (n - k)(l(x, A) + l(y, D)).$$

Therefore,

$$l(A, B) + l(C, D) \leq 2(n - k)(l(x, A) + l(y, D)) + l(A, C) + l(B, D). \quad (3)$$

There are two cases to be considered:

- $(n - k)(l(x, A) + l(y, D)) \leq l(A, C) + l(B, D)$. In this case the claim of the lemma follows directly from Eq. (3).
- $(n - k)(l(x, A) + l(y, D)) > l(A, C) + l(B, D)$.

Lemma 4.2 states that

$$(n - k)(l(x, A) + l(y, D) + l(x, B) + l(y, C)) \leq 2(l(B) + l(C)) + l(A, C) + l(B, D).$$

Under the assumption of this case it follows that

$$(n - k)(l(x, B) + l(y, C)) \leq 2(l(B) + l(C)). \quad (4)$$

Corollary 4.5 states that

$$l(A, B) + l(C, D) \leq 2(l(B) + l(C)) + l(A, C) + l(B, D) + (2k - n)(l(x, B) + l(y, C)).$$

By our assumption $2k - n \geq 0$, and with Eq. (4) it follows that

$$\begin{aligned} l(A, B) + l(C, D) &\leq 2(l(B) + l(C)) + l(A, C) + l(B, D) + \frac{2k - n}{n - k}(2(l(B) + l(C))) \\ &= l(A, C) + l(B, D) + 2(l(B) + l(C)) \left(1 + \frac{2k - n}{n - k}\right) \\ &= l(A, C) + l(B, D) + 2(l(B) + l(C)) \frac{k}{n - k}. \end{aligned}$$

According to Lemma 4.1 $[k/(n - k)]l(B) \leq l(B, D)$ and $[k/(n - k)]l(C) \leq l(A, C)$, so that

$$l(A, B) + l(C, D) \leq l(A, C) + l(B, D) + 2(l(A, C) + l(B, D)),$$

proving the lemma for this case too. \square

Lemma 4.8. *If $0.35opt < l(A, C) + l(B, D)$ then*

$$l(A, B) + l(C, D) \leq 0.7opt + l(A, C) + l(B, D).$$

Proof. According to the way the sets were defined

$$opt = l(A) + l(B) + l(C) + l(D) + l(A, C) + l(B, D).$$

Define α to satisfy $l(B) + l(C) = \alpha opt$. Then, by the assumption of the lemma

$$l(A) + l(D) \leq (1 - 0.35 - \alpha)opt = (0.65 - \alpha)opt.$$

If $(2k - n)(l(x, A) + l(y, D)) > (0.6 - 2\alpha)opt$ then

$$\begin{aligned} 2(l(A) + l(D)) - (2k - n)(l(x, A) + l(y, D)) &< (2(0.65 - \alpha) - (0.6 - 2\alpha))opt \\ &= 0.7opt. \end{aligned}$$

By Corollary 4.4

$$l(A, B) + l(C, D) \leq 0.7opt + l(A, C) + l(B, D).$$

Otherwise, $(2k - n)(l(x, A) + l(y, D)) \leq (0.6 - 2\alpha)opt$. Now there are 2 cases to be considered:

- $k \geq [(19 - 40\alpha)/(26 - 40\alpha)]n$.

In this case $n - k \leq [7/(26 - 40\alpha)]n$ and $2k - n \geq [(12 - 40\alpha)/(26 - 40\alpha)]n$. Since $(2k - n)(l(x, A) + l(y, D)) \leq (0.6 - 2\alpha)opt$

$$\begin{aligned} (n - k)(l(x, A) + l(y, D)) &\leq \frac{7}{12 - 40\alpha}(2k - n)(l(x, A) + l(y, D)) \\ &\leq \frac{7}{12 - 40\alpha} \frac{3 - 10\alpha}{5} opt \leq 0.35opt. \end{aligned}$$

Therefore, using Eq. (3)

$$l(A, B) + l(C, D) \leq 0.7opt + l(A, C) + l(B, D).$$

- $k < [(19 - 40\alpha)/(26 - 40\alpha)]n$.

In this case $n - k \geq [7/(26 - 40\alpha)]n$ and $2k - n \leq [(12 - 40\alpha)/(26 - 40\alpha)]n$. If $(n - k)(l(x, A) + l(y, D)) \leq 0.35opt$ the lemma is proven as before. Otherwise, since $k \geq n/2$,

$$\frac{n}{2}(l(x, A) + l(y, D)) \geq (n - k)(l(x, A) + l(y, D)) > 0.35opt.$$

By Eq. (2),

$$\begin{aligned} \frac{n}{2}(l(x, A) + l(y, D)) + \frac{n}{2}(l(x, B) + l(y, C)) &\leq opt \\ \Rightarrow \frac{n}{2}(l(x, B) + l(y, C)) &\leq 0.65opt. \\ \Rightarrow (2k - n)(l(x, B) + l(y, C)) &\leq \frac{12 - 40\alpha}{13 - 20\alpha} \frac{n}{2}(l(x, B) + l(y, C)) \\ &\leq \frac{12 - 40\alpha}{13 - 20\alpha} 0.65opt. \end{aligned}$$

By the definition of α ,

$$2(l(B) + l(C)) + (2k - n)(l(x, B) + l(y, C)) \leq \left(2\alpha + \frac{12 - 40\alpha}{13 - 20\alpha} 0.65\right)opt.$$

Define $f(\alpha) = 2\alpha + [(12 - 40\alpha)/(13 - 20\alpha)]0.65$. f has a local maximum when $(13 - 20\alpha)^2 = 91 \Rightarrow \alpha \approx 0.173$.

For this value of α , $f(\alpha) = 0.6921$. f has an asymptote at $\alpha = 0.65$, and by approaching this value from the right, the function goes off to infinity. However, since we assume that $k \leq [(19 - 40\alpha)/(26 - 40\alpha)]n$, we only care about values of α for which $0 \leq (19 - 40\alpha)/(26 - 40\alpha) \leq 1$ which holds for $\alpha \leq \frac{19}{40}$, giving that the local maximum we found is an upper bound for $f(\alpha)$ for all relevant values of α . So in any case

$$2(l(B) + l(C)) + (2k - n)(l(x, B) + l(y, C)) < 0.7opt.$$

Corollary 4.5 states that

$$l(A, B) + l(C, D) \leq 2(l(B) + l(C)) + l(A, C) + l(B, D) + (2k - n)(l(x, B) + l(y, C)),$$

hence

$$l(A, B) + l(C, D) \leq 0.7opt + l(A, C) + l(B, D). \quad \square$$

Theorem 4.9.

$$apx \leq 1.7opt.$$

Proof. There are 2 cases to be considered

1. $l(A, C) + l(B, D) \leq 0.35opt$.

Lemma 4.7 states that $l(A, B) + l(C, D) \leq 3(l(A, C) + l(B, D))$. Therefore, in this case $apx = opt + l(A, B) + l(C, D) - l(A, C) - l(B, D) \leq opt + 2(l(A, C) + l(B, D)) \leq 1.7opt$.

2. $0.35opt < l(A, C) + l(B, D)$.

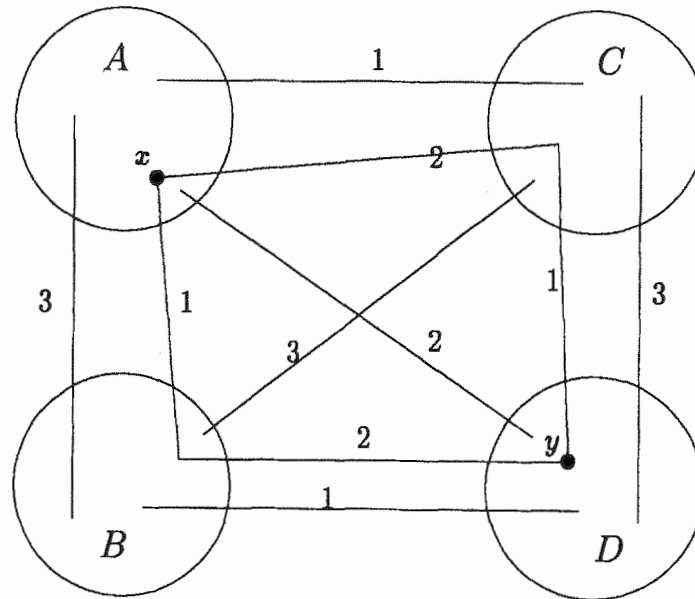
By Lemma 4.8 it follows that

$$l(A, B) + l(C, D) \leq 0.7opt + l(A, C) + l(B, D) \\ \Rightarrow apx \leq 1.7opt. \quad \square$$

We present now a family of instances in which apx is asymptotically $1\frac{2}{3}opt$. Consider the graph described in Fig. 3.

In the graph there are 4 sets of nodes A, B, C and D , each containing $q = n/2$ nodes. In A there is one specific node denoted x . In D there is one specific node denoted y . The length of an edge connecting two nodes from the same set is 2. The other edge lengths are defined according to the following rules:

- $a \in A \setminus \{x\}, c \in C \Rightarrow l(a, c) = 1,$
- $a \in A \setminus \{x\}, b \in B \Rightarrow l(a, b) = 3,$
- $b \in B, d \in D \setminus \{y\} \Rightarrow l(b, d) = 1,$
- $c \in C, d \in D \setminus \{y\} \Rightarrow l(c, d) = 3,$
- $c \in C \Rightarrow l(x, c) = 2, l(y, c) = 1,$

Fig. 3. A $1\frac{2}{3}$ example.

- $b \in B \Rightarrow l(y, b) = 2, l(x, b) = 1,$
- $b \in B, c \in C \Rightarrow l(b, c) = 3,$
- $a \in A, d \in D \Rightarrow l(a, d) = 2.$

It is clear from the above definitions that

$$l(A) = l(B) = l(C) = l(D) = q(q-1),$$

$$l(A, C) = l(B, D) = (q-1)q + 2q = q^2 + q,$$

$$l(A, B) = l(C, D) = 3(q-1)q + q = 3q^2 - 2q.$$

For a node $b \in B$, the closest nodes to it are of distance 1. There are exactly q nodes of distance 1 to b , the $q-1$ nodes in $D \setminus \{y\}$ and the node x . The other nodes have a greater distance to b . So for every $P \subset V, |P| = 2q, b \in P, l(b, P) \geq q + 2(q-1) = 3q - 2$.

Similarly, for every node $c \in C$ and every $P \subset V, |P| = 2q, c \in P, l(c, P) \geq 3q - 2$.

For every node $a \in A \setminus \{x\}$, the closest nodes to it are of distance 1. There are exactly q nodes of this distance, the nodes in C . The other nodes have a greater distance to a of length at least 2. So for every $P \subset V, |P| = 2q, a \in P, l(a, P) \geq 3q - 2$. Similarly, for every node $d \in D \setminus \{y\}$ $P \subset V, |P| = 2q, d \in P, l(d, P) \geq 3q - 2$.

For x there are q nodes of distance 1 (the nodes in B) and $q-1$ nodes of distance 2 (the nodes in A), so that $l(x, A \cup B) = 3q - 2$. In the same way, $l(y, C \cup D) = 3q - 2$, so that $x, y, A \cup B, C \cup D$ solve SPP. So the approximation may offer the partition

$A \cup B, C \cup D$ and

$$\begin{aligned} apx &= l(A) + l(B) + l(C) + l(D) + l(A, B) + l(C, D) = 4q(q - 1) + 2(3q^2 - 2q) \\ &= 10q^2 - 8q. \end{aligned}$$

On the other hand, the optimal solution to MCP is the partition $A \cup C, B \cup D$, so that

$$\begin{aligned} opt &= l(A) + l(B) + l(C) + l(D) + l(A, C) + l(B, D) = 4q(q - 1) + 2(q^2 + q) \\ &= 6q^2 - 2q. \end{aligned}$$

So, when q goes to infinity $apx = 1\frac{2}{3}opt$.

5. Local search

We wish to show that the application of local search to MCP may give a bad approximation result, even for the case of partitioning the graph into 2 equal-sized sets.

We consider two variations.

1. In each iteration the algorithm considers swapping two nodes, one from each existing set of nodes.

We will show that for each $0 < \varepsilon < 1$ the algorithm may give an approximation for which $apx = \frac{1}{\varepsilon}opt$.

Let $0 < \varepsilon < 1$. Choose n satisfying $n > 1 + \frac{1}{\varepsilon}$ (hence $\varepsilon(n - 1) > 1$). Consider a graph containing $2n$ nodes (described in Fig. 4). The nodes are divided into 4 sets (A, B, C and D), each containing $\frac{n}{2}$ nodes.

An edge connecting 2 nodes from the same set is of length 0. The other edge lengths are defined according to the following rules:

- $\forall a \in A, b \in B \quad l(a, b) = 1,$
- $\forall c \in C, d \in D \quad l(c, d) = 1,$
- $\forall a \in A, d \in D \quad l(a, d) = 1 + \varepsilon,$
- $\forall b \in B, c \in C \quad l(b, c) = 1 + \varepsilon,$
- $\forall a \in A, c \in C \quad l(a, c) = \varepsilon,$
- $\forall b \in B, d \in D \quad l(b, d) = \varepsilon.$

Suppose that the partition $P = A \cup B, Q = C \cup D$ is given.

Lemma 5.1. *When the algorithm considers exchanging a node $a \in A$ and a node $c \in C$ it will choose not to perform the exchange.*

Proof.

$$l(a, P \setminus \{a\}) = l(c, Q \setminus \{c\}) = \frac{n}{2},$$

while,

$$l(a, Q \setminus \{c\}) = l(c, P \setminus \{a\}) = \frac{n}{2}(1 + \varepsilon) + \left(\frac{n}{2} - 1\right)\varepsilon,$$

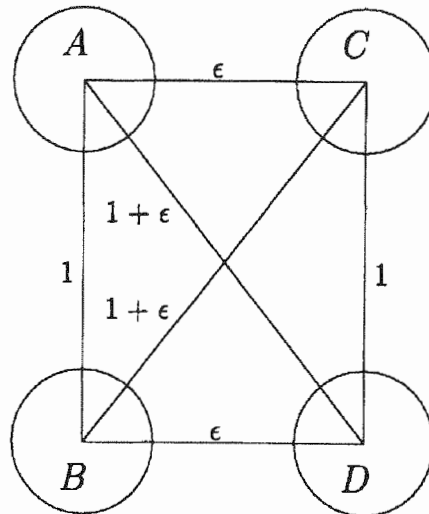


Fig. 4. Bad example for local search.

so that

$$l(a, Q \setminus \{c\}) + l(c, P \setminus \{a\}) > n(1 + \epsilon) > n = l(a, P \setminus \{a\}) + l(c, Q \setminus \{c\}). \quad \square$$

Lemma 5.2. *When the algorithm considers exchanging a node $a \in A$ and a node $d \in D$ it will choose not to perform the exchange.*

Proof.

$$l(a, P \setminus \{a\}) = l(d, Q \setminus \{d\}) = \frac{n}{2},$$

while

$$l(a, Q \setminus \{d\}) = l(d, P \setminus \{a\}) = \frac{n}{2}\epsilon + \left(\frac{n}{2} - 1\right)(1 + \epsilon).$$

So that

$$l(a, Q \setminus \{d\}) + l(d, P \setminus \{a\}) = n + 2(\epsilon(n - 1) - 1) > n,$$

where the inequality follows from the way that n was defined. Therefore,

$$l(a, Q \setminus \{d\}) + l(d, P \setminus \{a\}) > l(a, P \setminus \{a\}) + l(d, Q \setminus \{d\}). \quad \square$$

Similarly, when the algorithm considers exchanging a node from B with a node from C or D , it will choose not to perform the exchange.

Hence the partition P, Q , is locally optimal and $apx = l(P) + l(Q) = n^2/2$, while $opt = l(A \cup C) + l(B \cup D) = (n^2/2)\epsilon$. Thus,

$$apx = \frac{1}{\epsilon} opt.$$

2. Suppose we are looking for the best partition of the graph into 2 sets, not necessary of equal sizes. In this case we may consider another version for the local search:

In each iteration the algorithm considers moving a single node from its set to the other set.

Let $\varepsilon > 0$. Again choose $n > 1 + \frac{1}{\varepsilon}$.

Look at the same instance as before.

Suppose that the partition $P = A \cup B, Q = C \cup D$ is given.

Lemma 5.3. *When the algorithm considers a node $a \in A$ it will not change its set.*

Proof.

$$l(a, P \setminus \{a\}) = \frac{n}{2}.$$

On the other hand,

$$l(a, Q) = \frac{n}{2}\varepsilon + \frac{n}{2}(1 + \varepsilon) = \frac{n}{2} + n\varepsilon > l(a, P \setminus \{a\}). \quad \square$$

Similarly when the algorithm considers a node from B, C or D it will not change its set. So again the local search gives

$$apx = \frac{n^2}{2} = \frac{1}{\varepsilon} opt.$$

Appendix A. 1.686 example

To construct the example we considered the graph shown in Fig. 5. In this graph we defined as variables the lengths of the edges. We formulated a linear program in which the values a, \dots, h are the variables and the objective is to maximize the approximated value subject to a given value of the optimal solution. We took into consideration only costs related to a quadratic number of edges. Other terms became negligible when each node is replaced by m nodes and we let m go to infinity. We defined a set of inequalities to maintain that:

- $A \cup C, B \cup D$ is an optimal solution to the MCP.
- $A \cup B, C \cup D$ is an optimal solution to the SPP.
- The edge lengths satisfy the triangle inequality.

The solution obtained is described in Fig. 6. There are 4 sets of nodes A, B, C and D . A and D contain $0.585q$ nodes each, B and C contain $0.415q$ nodes each. In A there is one distinguished node denoted x . In D there is one distinguished node denoted y .

Let

$$|V| = 2q, \quad \alpha = 14.134276, \quad \beta = 24.161155.$$

The edge lengths are defined according to the following rules:

- $a_1, a_2 \in A \setminus \{x\} \Rightarrow l(a_1, a_2) = \beta,$

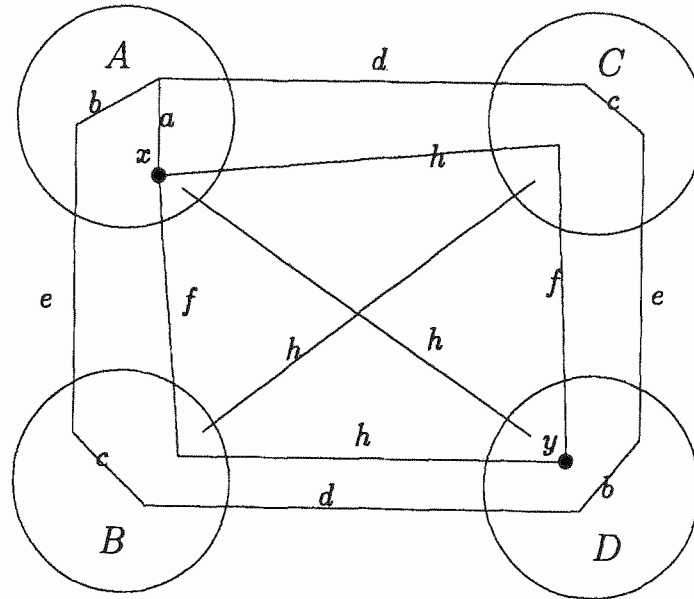


Fig. 5. The graph used to construct the example.

- $d_1, d_2 \in D \setminus \{y\} \Rightarrow l(d_1, d_2) = \beta$,
- $b_1, b_2 \in B \Rightarrow l(b_1, b_2) = 2\alpha$,
- $c_1, c_2 \in C \Rightarrow l(c_1, c_2) = 2\alpha$,
- $a \in A \setminus \{x\}, c \in C \Rightarrow l(a, c) = \alpha$,
- $b \in B, d \in D \setminus \{y\} \Rightarrow l(b, d) = \alpha$,
- $c \in C \Rightarrow l(x, c) = l(y, c) = 2\alpha$,
- $b \in B \Rightarrow l(x, b) = l(y, b) = 2\alpha$,
- $b \in B, a \in A \setminus \{x\} \Rightarrow l(a, b) = 3\alpha$,
- $c \in C, d \in D \setminus \{y\} \Rightarrow l(c, d) = 3\alpha$,
- $b \in B, c \in C \Rightarrow l(b, c) = 2\alpha$,
- $a \in A, d \in D \Rightarrow l(a, d) = 2\alpha$,
- $a \in A \setminus \{x\} \Rightarrow l(x, a) = \alpha$,
- $d \in D \setminus \{y\} \Rightarrow l(y, d) = \alpha$,
- $a \in A \Rightarrow l(y, a) = 2\alpha$,
- $d \in D \Rightarrow l(x, d) = 2\alpha$.

By the above definitions

$$l(A) = l(D) = (0.585q)^2 \beta / 2 + o(q^2) = 4.1342756q^2 + o(q^2),$$

$$l(B) = l(C) = (0.415q)^2 2\alpha / 2 + o(q^2) = 2.4342756q^2 + o(q^2),$$

$$l(A, C) = l(B, D) = 0.585q \cdot 0.415q\alpha + o(q^2) = 3.4314488q^2 + o(q^2),$$

$$l(A, B) = l(C, D) = 0.585q \cdot 0.415q3\alpha + o(q^2) = 10.294346q^2 + o(q^2).$$

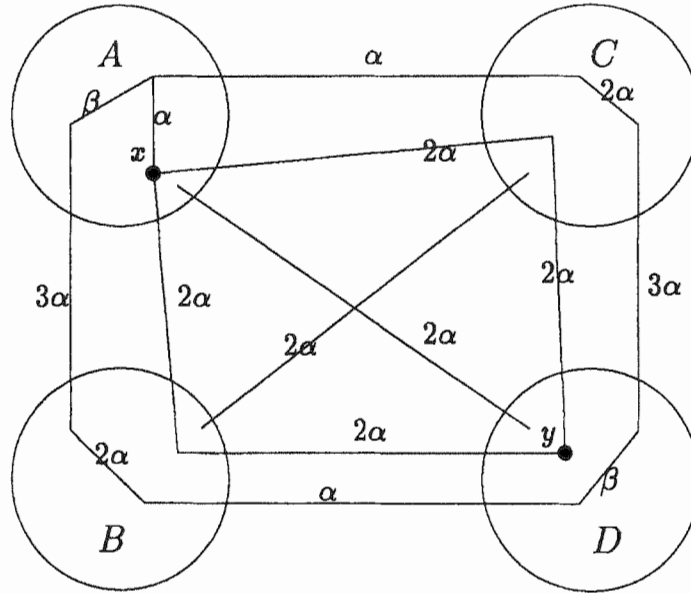


Fig. 6. A 1.68 example.

For a node $b \in B$, the closest nodes to it are of distance α . There are exactly $0.585q$ nodes of distance α to b , the $0.585q$ nodes in $D \setminus \{y\}$ and the node x . The other nodes have at least distance 2α to b . So for every $P \subset V$, $|P|=q$, $b \in P$, $l(b, P) \geq 0.585q\alpha + 0.415q2\alpha = 1.415q\alpha = 20q$.

Similarly, for every node $c \in C$ and every $P \subset V$, $|P|=q$, $c \in P$, $l(c, P) \geq 20q$.

For every node $a \in A \setminus \{x\}$, the closest nodes to it are of distance α . There are exactly $0.415q$ nodes of this distance, the nodes in C . The other nodes have a greater distance to a of length at least β . So for every $P \subset V$, $|P|=q$, $a \in P$, $l(a, P) \geq 0.415\alpha + 0.585\beta = 20q$. Similarly for every node $d \in D \setminus \{y\}$ and for every $P \subset V$, $|P|=q$, $d \in P$, $l(d, P) \geq 20q$.

For x there are $0.415q$ nodes of distance 2α (the nodes in B) and $0.585q - 1$ nodes of distance α (the nodes in A), so that $l(x, A \cup B) \leq 0.585q\alpha + 0.415q2\alpha = 20q$. In the same way, $l(x, C \cup D) \leq 20q$, so the partition $x, y, A \cup B, C \cup D$ solves SPP. Hence the approximation may offer the partition $A \cup B, C \cup D$ and

$$apx = l(A) + l(B) + l(C) + l(D) + l(A, B) + l(C, D) = 33.7257944q^2 + o(q^2).$$

On the other hand, the optimal solution to MCP is the partition $A \cup C, B \cup D$, so that

$$opt = l(A) + l(B) + l(C) + l(D) + l(A, C) + l(B, D) = 20q^2 + o(q^2).$$

So, when q goes to infinity

$$apx = 1.686opt.$$

References

- [1] N. Garg, V.V. Vazirani, M. Yannakakis, Approximate max-flow min-(multi)cut theorems and their applications, *SIAM J. Comput.* 25 (1995) 235–251.
- [2] J.A. Hartigan, *Clustering Algorithms*, New York, Wiley, 1975.
- [3] D.G. Kirkpatrick, P. Hell, On the completeness of a generalized min matching problem, *Proc. 10th Ann. ACM Symposium Theory of Computing*, 1978, pp. 240–245.
- [4] V. Kann, S. Khanna, J. Lagergren, A. Panconesi, On the hardness of approximating Max K-CUT and its dual, *Israeli Symp. on Theory of Computing and Systems*, 1996, pp. 61–67.
- [5] K.B. Lipkowitz, D.B. Boyd, *Reviews in Computational Chemistry*, vol. 7, VCH Publishers, New York, 1996.
- [6] S.K. Sahni, T.F. Gonzalez, P-complete approximation problems, *J. ACM* 23 (1976) 555–565.
- [7] T. Tokuyama, J. Nakano, Geometric algorithms for the minimum cost assignment problem, *Random Structures and Algorithms* 6 (1995) 393–405.