Parameterized algorithms for the 2-clustering problem with minimum sum and minimum sum of squares objective functions

Bang Ye Wu*and Li-Hsuan Chen
Dept. of Computer Science and Information Engineering
National Chung Cheng University, Taiwan

Abstract

In the Min-Sum 2-Clustering *problem, we are given a graph and a parameter k, and the goal is to determine if there exists a 2-partition of the vertex set such that the total conflict number is at most k, where the conflict number of a vertex is the number of its non-neighbors in the same cluster and neighbors in the different cluster. The problem is equivalent to* 2-Cluster Editing *and* 2-Correlation Clustering *with an additional multiplicative factor two in the cost function. In this paper we show an algorithm for* Min-Sum 2-Clustering *with time complexity $O(n \cdot 2.619^{r/(1-4r/n)} + n^3)$, where n is the number of vertices and $r = k/n$. Particularly, the time complexity is $O^*(2.619^{k/n})$ for $k \in o(n^2)$ and polynomial for $k \in O(n \log n)$, which implies that the problem can be solved in subexponential time for $k \in o(n^2)$. We also design a parameterized algorithm for a variant in which the cost is the sum of the squared conflict-numbers. For $k \in o(n^3)$, the algorithm runs in subexponential $O(n^3 \cdot 5.171^{\theta})$ time, where $\theta = \sqrt{k/n}$.*

Key words. parameterized algorithm, kernelization, cluster graph, clustering, graph modification.

AMS subject classifications. 65W05, 68R10, 68Q25, 05C85, 91C20

1 Introduction

Problem definition and motivation. Clustering is an important concept with applications in numerous fields, and the problem Cluster Editing, also known as Correlation Clustering, is a graph theoretic approach to clustering [2, 26]. A *cluster graph* is a graph whose every connected component is a clique. Cluster Editing asks for the minimum number of edge insertions and deletions to modify the input graph into a cluster graph. One of the studied variants is p-Cluster Editing, in which one is asked to modify the input graph into exactly p disjoint maximal cliques [26]. In many applications, graph edges represent the similarities between items (vertices), and one wants to partition items into clusters such that items in the same cluster are similar and items in different clusters are dissimilar. Hence, an edited edge can be thought of

*National Chung Cheng University, ChiaYi, Taiwan 621, R.O.C., E-mail: bangye@cs.ccu.edu.tw as a *conflict* (or *disagreement* as named in CORRELATION CLUSTERING) between two items in the clustering. Let the *conflict number* of a vertex be the number of edited edges incident to it. Then, feasible sets of edits of cardinality k correspond one-to-one to clusterings with total conflict number $2k$. That is, CLUSTER EDITING is equivalent to finding a vertex partition with minimum total conflict number. The transformation of the problem definition provides us easier ways to define other meaningful objective functions on the conflict numbers, such as the maximum conflict number and the sum of squared conflict numbers.

In this paper we focus on 2-clusterings which partition the vertices into two subsets called clusters. For an input graph $G = (V, E)$ and a 2-partition $\pi = (V_1, V_2)$ of V, two vertices u and v are in *conflict* if they are in the same cluster but $(u, v) \notin E$ or they are in different clusters but $(u, v) \in E$. Let $c_\pi(v)$ denote the number of vertices in conflict with v in π. A 2-clustering problem in general asks for a 2-partition such that the conflict numbers are as "small" as possible. As in many optimization problems, there are different ways to define "small" for a set of quantities (or a vector). Possibly the most frequently used cost functions are min-sum, min-sum of squares, and min-max, which are equivalent to the 1-norm, 2-norm, and ∞-norm for defining the length of a vector in linear algebra, respectively. Let $h_1(\pi) = \sum_{v\in V} c_\pi(v)$ and $h_2(\pi) = \sum_{v\in V} c_\pi^2(v)$ be costs of a 2-partition π. The two problems studied in this paper are formally defined as follows. We focus on their decision versions.

PROBLEM: MIN-SUM 2-CLUSTERING
INSTANCE: A graph $G = (V, E)$ and a nonnegative integer k.
QUESTION: Is there a 2-partition $\pi = (V_1, V_2)$ of V such that $h_1(\pi) \leq k$ and $V_1, V_2 \neq \emptyset$?

Following the definition in the literature, we exclude the case that V_1 or V_2 is empty. The second problem, named MIN-SQUARE 2-CLUSTERING, is defined similarly except the cost is defined by h_2. Intuitively, while MIN-SUM 2-CLUSTERING seeks to minimize the total conflict number, MIN-SQUARE 2-CLUSTERING looks for 2-partitions simultaneously minimizing the total and the individual conflict numbers because $h_2(\pi) = \sum_{v\in V} c_\pi^2(v) = h_1^2(\pi)/n + n\sigma^2$, where n is the number of vertices and σ^2 is the *variance* of the conflict numbers.

Previous results. Shamir et al. [26] studied the computational complexities of three edge modification problems. CLUSTER EDITING asks for the minimum total number of edge insertions and deletions to modify a graph into a cluster graph, while in CLUSTER DELETION (respectively, CLUSTER COMPLETION), only edge deletions (respectively, insertions) are allowed. They showed that CLUSTER EDITING is NP-hard, CLUSTER DELETION is Max SNP-hard, and CLUSTER COMPLETION is polynomial-time solvable. They also showed that p-CLUSTER DELETION is NP-hard for any $p > 2$ but polynomial-time solvable for $p = 2$, and p-CLUSTER EDITING is NP-hard for any $p \geq 2$. The parameterized version of CLUSTER EDITING and variants of it were studied intensively [3, 4, 8, 10, 11, 18, 19, 20]. A variant with vertex (rather than edge) deletions was considered in [23], and another variant in which overlapping clusters are allowed was studied in [13].

CLUSTER EDITING is equivalent to CORRELATION CLUSTERING on complete signed graphs. In a signed graph, each edge is labelled by "+" or "-", representing that the two items are similar or dissimilar; or the two persons like or dislike each other in social network analysis. For a clustering, a positive edge within a cluster or a negative edge between clusters is an *agreement*, and a positive edge across clusters or a negative edge inside a cluster is a *disagreement*. The maximization version of the CORRELATION CLUSTERING problem seeks to maximize the number of agreements, while the minimization version aims to minimize the number of disagreements. CORRELATION CLUSTERING on complete signed graphs was formulated and studied in [2], in which the authors presented a PTAS for the maximization version and a constant factor approximation algorithm for the minimization version. In [1], Ailon et al. proposed a simple randomized algorithm for the minimization version. For the unweighted case, the expected approximation ratio is three. They also showed that it is a 5-approximation algorithm for the weighted case with the *probability constraints*, i.e., it is assumed that $w_{ij}^- + w_{ij}^+ = 1$ for each pair (i, j) of vertices, where w_{ij}^+ and w_{ij}^- are the nonnegative weights of the "+" and "-" edges between i and j, respectively. If in addition the weights satisfy the triangle inequality ($w_{ik}^- \leq w_{ij}^- + w_{jk}^-$ for all vertices i, j, k), then the approximation ratio is two.

For a constant $p \geq 2$, p-CORRELATION CLUSTERING is a variant of CORRELATION CLUSTERING such that the vertices are partitioned into exactly p clusters. While the minimization version of CORRELATION CLUSTERING on complete signed graphs is APX-hard [7], Giotis and Guruswami showed that both the minimization and the maximization versions of p-CORRELATION CLUSTERING admit PTAS for unweighted complete signed graphs [17]. 2-CORRELATION CLUSTERING is also known as BALANCED SUBGRAPH which name comes from the application in social network analysis [21, 22, 27]. Another related problem studied in the literature is CONSENSUS CLUSTERING [1, 5, 6, 14].

For p-CLUSTER EDITING, a kernel with $(p+2)k+p$ vertices was given by Guo [20]. A variant such that the conflict number of each vertex must be bounded by a parameter was studied in [24]. The problem of finding a 2-clustering minimizing the maximum conflict number is NP-hard [9]. Very recently, Fomin et al. gave a parameterized algorithm with time complexity $O(2^{O(\sqrt{pk})}+n^2)$, where n is the number of vertices [16]. They also showed a lower bound for the parameterized complexity: For any constant $0 \leq \sigma \leq 1$ there exists a function $p(k) \in \Theta(k^\sigma)$ such that p-CLUSTER EDITING restricted to instances with $p = p(k)$ cannot be solved in $2^{o(\sqrt{pk})} \cdot n^{O(1)}$ time unless the Exponential Time Hypothesis fails.

Our contributions. We develop parameterized algorithms for MIN-SUM 2-CLUSTERING and MIN-SQUARE 2-CLUSTERING. For MIN-SUM 2-CLUSTERING, the algorithm runs in $O(n \cdot \phi^{2r/(1-4r/n)} + n^3)$ time, where $\phi \approx 1.618$ and $r = k/n$. When $k \in o(n^2)$, the time complexity is $O(n \cdot 2.619^{k/n} + n^3)$. Our result implies that the problem, as well as 2-CLUSTER EDITING can be solved in subexponential, i.e. $O^*(2^{o(n)})$, time for $k \in o(n^2)$, where the $O^*(\cdot)$ notation ignores factors polynomial in n. In particular it is polynomial-time solvable for $k \in O(n \log n)$. We also note that the time complexity is better than $O^*(2^{O(\sqrt{pk})})$ recently obtained by Fomin et al. [16] for the special case of $p = 2$. Even when $k = \delta n^2$ with small constant δ, our algorithm improves the brute-force algorithm significantly. For example, when $\delta = 0.1$, we have that $r = k/n = 0.1n$, and the time complexity is $O^*(\phi^{2r/(1-4r/n)}) \approx O^*(1.174^n)$, much better than $O^*(2^n)$.

For MIN-SQUARE 2-CLUSTERING with cost bound $k \in o(n^3)$, our algorithm runs in subexponential $O(n^3 \cdot 5.171^{\sqrt{k/n}})$ time, which also implies that the problem is polynomial-time solvable for $k \in O(n \log^2 n)$.

Both of the algorithms look for a 2-partition and the main steps are sketched as follows.

1. Guess a vertex that has the smallest conflict number in an optimal 2-partition and set an initial 2-partition according to its neighborhood.
2. Using kernelization-style rules, determine for most of the vertices whether they should be swapped to the other cluster, and leave a set of undetermined vertices of size $O(k/n)$ in the case of Min-Sum 2-Clustering, or $O(\sqrt{k/n})$ in the case of Min-Square 2-Clustering.
3. Apply a standard branching algorithm to the undetermined vertices.

Organization of the paper. In Section 2 we give some notation and definitions, as well as some properties used in this paper. The reduction algorithm is in Section 3. In Sections 4 and 5 we show the algorithms for the two problems, respectively. Finally some concluding remarks are in Section 6.

2 Preliminaries

An instance of a parameterized problem consists of (I, k), where k is the parameter. A problem is *fixed-parameter tractable* (FPT) if it can be solved in time complexity $O(f(k) \cdot q(|I|))$, where f is an arbitrary computable function of k and q is a polynomial in the input size. For more details about parameterized complexity, we refer to the book of Downey and Fellows [12]. *Kernelization* is a widely-used technique for parameterized algorithms. In polynomial time, a kernelization algorithm converts an instance (I, k) to a reduced instance (I', k'), called a *kernel* such that the answer is not changed, $k' \leq k$ and $|I'|$ is bounded by a computable function of k.

For two sets S_1 and S_2, the set difference is denoted by $S_1 \setminus S_2$, and the symmetric difference is denoted by $S_1 \ominus S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. For simplicity, $S_1 \ominus v = S_1 \ominus \{v\}$. Throughout this paper, $G = (V, E)$ is the input graph and $n = |V|$. For a vertex v, let $N_G[v] = \{u \mid (u, v) \in E\} \cup \{v\}$ denote the closed neighborhood of v in G.

For a vertex set V, a *2-partition* of V is an unordered pair $\pi = (V_1, V_2)$ of subsets of V such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$. The two subsets V_1 and V_2 are called *clusters*. Two vertices u and v are in *conflict* if they are in the same cluster but $(u, v) \notin E$ or they are in different clusters but $(u, v) \in E$. Let $C_\pi(v)$ denote the set of vertices in conflict with v in π and $c_\pi(v) = |C_\pi(v)|$ be the *conflict number* of v. We assume that $v \notin C_\pi(v)$ for each vertex v. When there is no confusion, we shall omit the subscript and simply use $c(\cdot)$ instead of $c_\pi(\cdot)$. For $u, v \in V$ and $S \subseteq V$, let $C_\pi(v, S) = C_\pi(v) \cap S$, $c_\pi(v, S) = |C_\pi(v, S)|$, and $c_\pi(v, u) = c_\pi(v, \{u\})$. For two vertex subsets S_1 and S_2, let $c_\pi(S_1, S_2) = \sum_{v \in S_1} c_\pi(v, S_2)$. Note that $c_\pi(S_1, S_2) = c_\pi(S_2, S_1)$. When the 2-partition π is clear from the context, we shall also omit the subscript in $c_\pi(\cdot, \cdot)$ and $C_\pi(\cdot, \cdot)$.

A graph $G = (V, E)$ is a *2-cluster graph* if it consists of exactly two disjoint maximal cliques. In the literature, Min-Sum 2-Clustering is also known as 2-Cluster Editing. For $G = (V, E)$, a set $D \subseteq V \times V$ is a *set of edits* (to 2-cluster graph) for G if $G' = (V, E \ominus D)$ is a 2-cluster graph. In other words, G can be modified into a 2-cluster graph by inserting $D \setminus E$ and deleting $D \cap E$. Given a graph G and integer k, 2-Cluster Editing asks if there is a set of edits D for G such that $|D| \leq k$. Let

$E^*(\pi) = \bigcup_{i=1,2}\{(u,v) \mid u,v \in V_i\}$ which is the edge set of the 2-cluster graph. By definition, if D is a set of edits and π is the corresponding 2-partition, then $E \ominus D = E^*(\pi)$ and $D = \bigcup_{v\in V}\{(v,u) \mid u \in C_\pi(v)\}$. Therefore, finding a set of edits is equivalent to finding the corresponding 2-partition. Furthermore, $|D| = (1/2)\sum_{v\in V} c_\pi(v)$. Consequently Min-Sum 2-Clustering is equivalent to 2-Cluster Editing with an additional multiplicative factor two in the cost function.

Both Min-Sum 2-Clustering and Min-Square 2-Clustering look for a 2-partition without empty clusters. The cost functions of the two problems are $h_1(\pi) \equiv \sum_v c_\pi(v)$ and $h_2(\pi) \equiv \sum_v c_\pi^2(v)$, respectively. Given a graph G and a 2-partition π, computing $C_\pi(v)$ for all $v \in V$, as well as $h_1(\pi)$ and $h_2(\pi)$, can be easily done in $O(n^2)$ time by checking all pairs of vertices.

To *flip* a vertex v in a 2-partition $\pi = (V_1, V_2)$ is to move v to the other cluster, that is, we change π to $\pi \ominus v \equiv (V_1 \ominus v, V_2 \ominus v)$. Flipping a subset S of vertices changes (V_1, V_2) to $(V_1 \ominus S, V_2 \ominus S)$. Suppose that $\pi' = \pi \ominus v$. Then $u \in C_{\pi'}(v)$ if and only if $u \notin C_\pi(v)$ for all $u \neq v$. That is, flipping a vertex exchanges its conflicting-relations with all the other vertices. Furthermore, when flipping a set F, only those conflicting pairs in $F \times \bar{F}$ change, where $\bar{F} = V \setminus F$. If $\pi' = \pi \ominus F$, then

$$C_{\pi'}(v) = \begin{cases} C_\pi(v,F) \cup (\bar{F} \setminus C_\pi(v,\bar{F})) & \text{if } v \in F; \\ C_\pi(v,\bar{F}) \cup (F \setminus C_\pi(v,F)) & \text{if } v \in \bar{F}. \end{cases} \tag{1}$$

The *profit* of a flipping set F, denoted by $\Delta(F)$, is the decrement of total conflict number after flipping F, i.e., $\Delta(F) = h_1(\pi) - h_1(\pi \ominus F) = \sum_v c_\pi(v) - \sum_v c_{\pi\ominus F}(v)$.

Lemma 1: $\Delta(F) = 4c_\pi(F,\bar{F}) - 2|F||\bar{F}|$.

Proof: By (1), we only need to count the conflict pairs crossing $(F,\bar{F})$. Since $c_\pi(F,\bar{F}) = c_\pi(\bar{F},F)$, we have

$$\Delta(F) = 2(c_\pi(F,\bar{F}) - (|F||\bar{F}| - c_\pi(F,\bar{F}))) = 4c_\pi(F,\bar{F}) - 2|F||\bar{F}|.$$

□

Corollary 2: $\Delta(\{v\}) = 4c_\pi(v) - 2(n-1)$.

3 Reduction algorithm

We shall solve Min-Sum 2-Clustering and Min-Square 2-Clustering by finding flipping sets. In this section we propose a reduction algorithm which reduces the search space of flipping sets, and the reduction will be used in the next two sections.

Definition 1: Let π be a 2-partition of V and K, t be nonnegative integers. A vertex subset F is a (K,t)-feasible flipping set for π if $\sum_v c_{\pi'}(v) \leq K$ and $c_{\pi'}(v) \leq t$ for any vertex $v \in V$, where $\pi' = \pi \ominus F$. When K and t are clear from the context, we shall simply say that F is a feasible flipping set.

By definition, an empty set may be a (K,t)-feasible flipping set. The goal of this section is a reduction algorithm achieving the next lemma.

Algorithm 1 : REDUCTION(G, π_0, K, t, f)

```
Input: a graph $G = (V, E)$, a 2-partition $\pi_0$, and integers $K$, $t$ and $f$.
Output: a 2-partition $\pi$, a vertex subset $U$, and an integer $m$.
 1: initially $\pi = \pi_0$;
 2: compute $c_\pi(u)$ for each $u$ and construct $U \leftarrow \{u \mid c_\pi(u) \geq n - t - f\}$;
 3: while $\exists v \in U$ such that $c_\pi(v) > t + f$ do
 4:     $\pi \leftarrow \pi \ominus v$;                                    ▷ flipping $v$
 5:     remove $v$ from $U$ and update $c_\pi(u)$ for each $u$;
 6:     $f \leftarrow f - 1$ and $U \leftarrow \{u \mid c_\pi(u) \geq n - t - f\}$;
 7:     if $f = 0$ then goto step 14;
 8: end while
 9: if $|U| > \frac{K}{n-t-2f} - f$ then
10:     $f \leftarrow f - 1$ and $U \leftarrow \{u \mid c_\pi(u) \geq n - t - f\}$;
11:     if $f = 0$ then goto step 14;
12:     goto step 3;
13: end if
14: return $(U, \pi, m = f)$;
```

Lemma 3: Let π_0 be a 2-partition of V and K, t, f be nonnegative integers satisfying $t + 2f < n$. Given (G, π_0, K, t, f), one can in $O(n^2)$ time compute a vertex subset U, a 2-partition π, and an integer m such that

(*i*) $m \leq f$;

(*ii*) $|U| \leq K/(n - t - 2m) - m$ or $m = 0$; and

(*iii*) if there exists a (K, t)-feasible flipping set for π_0 of size at most f, then there exists a (K, t)-feasible flipping set $F \subseteq U$ for π of size at most m.

The bound f on the size of flipping sets will be called *flipping quota.* In the remaining paragraphs of this section, we first describe the reduction algorithm, and then show the correctness of the reduction rules in Lemmas 4 and 5. The proof of Lemma 3 will be delayed to the end of this section. The reduction algorithm is based on the following reduction rules for any 2-partition π and integer f. We omit the parameters (K, t) in the rules.

R1: If $c_\pi(v) > t + f$, then v must be in any feasible flipping set for π of size at most f.

R2: If $c_\pi(v) < n - t - f$, then v cannot be in any feasible flipping set for π of size at most f.

R3: If $c_\pi(v) \leq t + f$ for all $v \in V$ and $|U| > K/(n - t - 2f) - f$, where $U = \{v \mid c_\pi(v) \geq n - t - f\}$, then there is no feasible flipping set for π of size exactly f.

Algorithm 1 is the reduction algorithm. We note that the upper bound of $|U|$ in R3 is for the case of flipping set of size *exactly* f. When $|U| > K/(n - t - 2f) - f$, it is still possible that there is a feasible flipping set of size less than f. Therefore, the algorithm iteratively decreases f until the bound is satisfied or the flipping quota is zero.

Lemma 4: The reduction rules R1 and R2 are correct.

Proof: Since the conflict number of v is decreased by at most one when another vertex is flipped, if $c_\pi(v) > t + f$ and v is not flipped, then its conflict number will be larger than t. So R1 is correct.

For R2, if $c_\pi(v) < n - t - f$, then flipping v will change its conflict number to $n - 1 - c_\pi(v) > t + f - 1$, and further flipping $f - 1$ vertices make the conflict number at least $t + 1$. Therefore R2 is correct. □

We now show the upper bound of $|U|$ in rule R3.

Lemma 5: Suppose that $t + 2f < n$ and $c_\pi(v) \le t + f$ for all $v \in V$. If there exists a (K, t)-feasible flipping set $F \subseteq U$ for π of size f, then $|U| \le K/(n - t - 2f) - f$, where $U = \{v \mid c_\pi(v) \ge n - t - f\}$.

Proof: Let $\bar{F} = V \setminus F$ and $X = V \setminus U$. We omit the subscript π in the proof. By Lemma 1,

$$\sum_{v \in V} c(v) \le K + 4c(F, \bar{F}) - 2f(n - f). \tag{2}$$

Let $Y = U \setminus F$. Since $V = Y \cup F \cup X$ and Y, F, X are mutually disjoint, we have that

$$\sum_{v \in V} c(v) = \sum_{v \in Y} c(v) + \sum_{v \in F} c(v) + \sum_{v \in X} c(v).$$

For $v \in F$, since $c(v) \ge c(v, \bar{F})$,

$$\sum_{v \in F} c(v) \ge c(F, \bar{F}). \tag{3}$$

Since (X, Y) is a 2-partition of $\bar{F}$, we have that

$$\begin{aligned} c(X, F) &= c(F, X) = \sum_{v \in F} c(v, X) \\ &= \sum_{v \in F} (c(v, \bar{F}) - c(v, Y)) \\ &\ge \sum_{v \in F} (c(v, \bar{F}) - |Y|) \\ &= c(F, \bar{F}) - f|Y|, \end{aligned}$$

and then

$$\sum_{v \in X} c(v) \ge c(X, F) \ge c(F, \bar{F}) - f|Y|. \tag{4}$$

Therefore, by (2), (3) and (4),

$$\begin{aligned} \sum_{v \in Y} c(v) &= \sum_{v \in V} c(v) - \sum_{v \in F} c(v) - \sum_{v \in X} c(v) \\ &\le (K + 4c(F, \bar{F}) - 2f(n - f)) - c(F, \bar{F}) - (c(F, \bar{F}) - f|Y|) \\ &= K + 2c(F, \bar{F}) - 2f(n - f) + f|Y|. \end{aligned}$$

Since $c(v) \geq n - t - f$ for any $v \in Y$, we have that

$$(n-t-f)|Y| \leq \sum_{v \in Y} c(v) \leq K + 2c(F, \bar{F}) - 2f(n-f) + f|Y|,$$

and then

$$|Y| \leq \frac{K + 2c(F, \bar{F}) - 2f(n-f)}{n-t-2f} \tag{5}$$

by the assumption $t + 2f < n$. Since $c(v) \leq t + f$ for any $v \in F$, we have that $c(F, \bar{F}) \leq f(t+f)$, and thus

$$\begin{aligned} |Y| &\leq \frac{K + 2f(t+f) - 2f(n-f)}{n-t-2f} \\ &= \frac{K + 2f(t-n+2f)}{n-t-2f} = \frac{K}{n-t-2f} - 2f. \end{aligned} \tag{6}$$

Finally,

$$|U| \;=\; f + |Y| \leq \frac{K}{n-t-2f} - f. \tag{7}$$

□

Proof of Lemma 3 : First we show the time complexity. The initial conflict numbers for all vertices can be computed in $O(n^2)$ time. The while-loop is executed at most n times, and each loop takes $O(n)$ time for finding and flipping a vertex, as well as updating the conflict numbers and the set U. The total time complexity is therefore $O(n^2)$.

The conclusions (i) and (ii) are trivial from the reduction algorithm, and (iii) follows from the correctnesses of the reduction rules, which are shown in Lemmas 4 and 5. Note that, by rules R1 and R2, we only need to find the flipping set in U. □

4 Minimum total conflict number

In this section we show an algorithm for Min-Sum 2-Clustering, in which the cost function is defined by $h_1(\pi) = \sum_v c_\pi(v)$, i.e., the total conflict number. First we show how to cope with the simple case which will be excluded in the main procedure. A 2-partition (V_1, V_2) is *trivial* if V_1 or V_2 is empty; and is *extreme* if $|V_1| = 1$ or $|V_2| = 1$.

Lemma 6: Finding an extreme 2-partition π with minimum $h_1(\pi)$ can be done in $O(n^2)$ time.

Proof: The conflict number of any vertex v in the 2-partition $(V, \emptyset)$ is $n - |N_G[v]|$. By Corollary 2, if v is a vertex with minimum $|N_G[v]|$, then $(\{v\}, V \setminus \{v\})$ is an extreme 2-partition with minimum cost h_1. That is, we only need to find a vertex with minimum degree in G. □

Lemma 7: If π is a 2-partition such that $h_1(\pi) \leq h_1(\pi \ominus v)$ for any vertex v, then $c_\pi(v) \leq (n-1)/2$ for each v.

Proof: If $c_\pi(v) > (n-1)/2$, then by Corollary 2 the profit of flipping v is $4c_\pi(v) - 2(n-1) > 0$, and thus flipping v decreases the h_1 cost. □

Lemma 8: Suppose that the h_1 cost of any extreme 2-partition is larger than k. If there exists a non-trivial and non-extreme 2-partition π with $h_1(\pi) \leq k$, then there exists a (k,t)-feasible flipping set F for π_0 of size at most f, where $\pi_0 = (N_G[s], V \setminus N_G[s])$ for some vertex s, $t = (n-1)/2$, and $f = k/n$.

Proof: Let π be a non-trivial and non-extreme 2-partition with minimum $h_1(\pi)$. Thus, $h_1(\pi) \leq k$ and both clusters of π contain at least two vertices. By the minimality of π, if $h_1(\pi \ominus v) < h_1(\pi)$ for some v, then $\pi \ominus v$ must be extreme. However, it contradicts to the assumption that the h_1 cost of any extreme 2-partition is larger than k. Therefore $h_1(\pi) \leq h_1(\pi \ominus v)$ for any v, and by Lemma 7 we have that $c_\pi(v) \leq (n-1)/2$ for each v.

By the pigeonhole principle, there exists a vertex s with $c_\pi(s) \leq h_1(\pi)/n \leq k/n$. Let $F = C_\pi(s)$. By definition, no vertex is in conflict with s in $\pi \ominus F$, i.e., $\pi \ominus F = (N_G[s], V \setminus N_G[s]) \equiv \pi_0$ which is the unique 2-partition with no conflict incident to s. Therefore, $|F| \leq k/n$, and flipping F in π_0 yields a 2-partition π such that $h_1(\pi) \leq k$ and $c_\pi(v) \leq (n-1)/2$ for each v. That is, F is a (k,t)-feasible flipping set for π_0 of size at most k/n. □

By Lemma 3, we can use the reduction algorithm with $K = k$, $t = (n-1)/2$ and $f = k/n$. The assumption $t + 2f < n$ in Lemma 3 is satisfied when $k \leq n^2/4$. In fact, the largest value of k we will use in this section is $0.185n^2$ (Corollary 12). The reduction algorithm returns (U, π, m), and the remaining work is to search a flipping set $F \subseteq U$ with $h_1(\pi \ominus F) \leq k$ and $|F| \leq m$. This work can be done by a simple search-tree algorithm which picks an arbitrary undetermined vertex v and recursively solves the problem for two cases: flipping v or not. Algorithm 2 is the proposed algorithm. Note that we need to try every vertex s as the one in Lemma 8.

A naive implementation of the search-tree algorithm takes $O(n^2)$ time for each recursive call, and the time complexity of the search-tree algorithm will be $O(n^2)$ multiplied by the number of recursive calls. Similar to the technique usually used in the design of fixed-parameter algorithms [25], if the time complexity of each recursive call is a function in $|U|$ but not in n, then we can reduce the polynomial factor in the total time complexity, which is exactly the case shown in Lemma 10. To this aim, the recursive procedure is designed in Algorithm 3.

Lemma 9: The procedure SEARCH1 is correct and each recursive call takes $O(|U|)$ time.

Proof: The correctness of the algorithm follows from the following three simple observations. First, it explores all subsets of U with size at most m in the worst case. Second, the total conflict number $h_1(\pi)$ can be computed as $c_\pi(X,X) + 2c_\pi(U,X) + c_\pi(U,U)$, where $X = V \setminus U$. Third, when the vertex u is flipped, the two sets of vertices conflicting

Algorithm 2 : Min-sum 2-clustering

Input: a graph $G = (V, E)$ and integer k.
Output: determining if existing π with $h_1(\pi) \le k$.

```
1: if existing an extreme 2-partition with cost at most $k$ then
2:     return True;
3: end if
4: for each $s \in V$ do
5:     $\pi_0 \leftarrow (N_G[s], V \setminus N_G[s])$;
6:     call REDUCTION$(G, \pi_0, k, (n-1)/2, k/n)$ to compute $(U, \pi, m)$;
7:     $\chi \leftarrow c_\pi(X, X)$, where $X = V \setminus U$;        ▷ Preparing for tree-search
8:     construct the list $\mathcal{L}_1$ of $C_\pi(v, U)$ and the list $L_2$ of $c_\pi(v, X)$ for each $v \in U$;
9:     if SEARCH1$(U, m, L_2, \chi)$=True then
10:        return True;
11:    end if
12: end for
13: return False.
```

and non-conflicting with u exchange, and therefore the formulas at steps 15 and 16 of Algorithm 3 are correct.

At each recursive call, there are only $O(|U|)$ data to be updated. The number of conflicting pairs in U may be up to $\Theta(|U|^2)$. To avoid copying the conflicting pairs, the list $\mathcal{L}_1$ is stored as a global variable and the modifications are stored in a local variable L_3. When returning from the recursive call with "False", $\mathcal{L}_1$ is recovered. Note that it is not necessary to recover $\mathcal{L}_1$ when the recursive call returns "True". Since the number of modifications stored in L_3 is upper bounded by $O(|U|)$, the total time complexity for each recursive call is $O(|U|)$. $\square$

The algorithm we show here only returns True or False. In the case that a desired 2-partition needs to be output, we can record the flipped vertex at each recursive call (in constant time), and the 2-partition can be found by back tracking on the search tree in an additional $O(n)$ time. Another thing that should be remarked is how to exclude trivial 2-partitions in the search-tree algorithm, which by definition are invalid. Let $\pi = (V_1, V_2)$ be the 2-partition returned by the reduction algorithm at step 6 of Algorithm 2. Recall that the initial 2-partition is $(N_G[s], V \setminus N_G[s])$ for some vertex s. Since the initial conflict number of s is zero, the reduction algorithm never flips s, and therefore $s \in V_1 \setminus U$. Thus, V_2 is the only possible flipping set to result in a trivial 2-partition. A simple way to avoid returning a trivial 2-partition uses a boolean variable which indicates whether there is a vertex fixed in V_2. When reaching a leaf of the branching tree, it can be easily verified if it is the invalid flipping set.

Lemma 10: Algorithm 3 runs in $O(\phi^{|U|+m})$ time, where $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$.

Proof: Let $T(a, b)$ denote the time complexity of the search-tree algorithm with $|U| = a$ and $m = b$. There are two branches at each non-leaf node of the search-tree. For the branch that a vertex is removed from U without flipping, $|U|$ is decreased by one and m is unchanged. For the branch that a vertex is flipped and removed, both $|U|$ and m are decreased by one. Therefore, for some constant p_1, $T(a, b) \le T(a-1, b) + T(a-$

Algorithm 3 Search-tree algorithm for MIN-SUM 2-CLUSTERING

Input: a set U of undetermined vertices, a flipping quota m, a list L_2 of $c(v,X)$ for each $v \in U$, and $\chi = c(X,X)$, where $X = V \setminus U$. In addition, a list $\mathcal{L}_1$ of $C(v,U)$ for each $v \in U$ is stored as a global variable.

```
1:  procedure SEARCH1($U, m, L_2, \chi$)
2:      if $\chi > k$ then return False;
3:      if $U = \emptyset$ or $m = 0$ then                                  ▷ no vertex can be flipped
4:          $q \leftarrow \chi + \sum_{v \in U}(|C(v,U)| + 2c(v,X))$;            ▷ total conflict number
5:          if $q \le k$ then return True else return False;
6:      end if
7:      pick an arbitrary vertex $u \in U$;
8:      $U' \leftarrow U \setminus \{u\}$;                                          ▷ $X' = V \setminus U'$
9:      modify $\mathcal{L}_1$: $C(v,U') \leftarrow C(v,U) \setminus \{u\}, \forall v \in U'$; record the modifications in $L_3$;
10:     $\chi' \leftarrow \chi + 2c_\pi(u,X)$;                               ▷ move $u$ to $X$ without flipping
11:     construct $L'_2$ from $L_2$ by $c(v,X') \leftarrow c(v,X) + c(v,u), \forall v \in U'$;
12:     if SEARCH1($U', m, L'_2, \chi'$)=True then
13:         return True;
14:     end if
15:     $\chi'' \leftarrow \chi + 2(|X| - c(u,X))$;                          ▷ flip and move $u$ to $X$
16:     construct $L''_2$ from $L_2$ by $c(v,X') \leftarrow c(v,X) + 1 - c(v,u), \forall v \in U'$;
17:     if SEARCH1($U', m-1, L''_2, \chi''$)=True then
18:         return True;
19:     end if
20:     recover $\mathcal{L}_1$ by undoing the modifications in $L_3$;
21:     return False;
22: end procedure
```

$1, b-1) + p_1 a$ for $a, b > 0$; $T(a,0) \le p_1 a$ for any a; and $T(0,b) \le p_1$. We shall show by induction that

$$T(a,b) \le p_2 \phi^{a+b} - p_1(a+2) \tag{8}$$

for some constant p_2. Then, the time complexity is $T(|U|, m) \in O(\phi^{|U|+m})$.

It is easy to see that, for sufficiently large p_2, $T(0,b) \le p_1 \le p_2\phi^b - 2p_1$; and $T(a,0) \le p_1 a \le p_2\phi^a - p_1(a+2)$. Suppose by induction hypothesis that (8) holds for $T(a-1, b-1)$ and $T(a-1,b)$. Note that ϕ is the solution of Fibonacci recursion and therefore $\phi^{i+2} = \phi^{i+1} + \phi^i$. For $a, b > 0$,

$$\begin{aligned} T(a,b) &\le T(a-1,b) + T(a-1,b-1) + p_1 a \\ &\le (p_2\phi^{a+b-1} - p_1(a+1)) + (p_2\phi^{a+b-2} - p_1(a+1)) + p_1 a \\ &= p_2\phi^{a+b} - p_1(a+2). \end{aligned}$$

□

Theorem 11: For $k \le n^2/4$, MIN-SUM 2-CLUSTERING can be solved in $O(n \cdot 2.619^{r/(1-4r/n)} + n^3)$ time, where $r = k/n$.

Proof: A non-trivial 2-partition is either extreme or non-extreme. Algorithm 2 copes with the case of extreme 2-partitions at step 1 which takes $O(n^2)$ time by Lemma 6. If there is no extreme 2-partition with h_1 cost at most k, then Lemma 8 can be applied, and the non-extreme case is coped by the remaining steps. By (7), the number of undetermined vertices after reductions is

$$|U| \leq \frac{K}{n-t-2m} - m = \frac{k}{(n+1)/2-2m} - m \leq \frac{k}{n/2-2m} - m, \tag{9}$$

where $K = k$ is the required bound of the total conflict number, $t = (n-1)/2$, and m is the returned flipping quota. Since $m \leq k/n$, $|U| + m \leq k/(n/2 - 2m) \leq 2r/(1 - 4r/n)$. By Lemma 10, the time complexity of the search-tree algorithm is

$$O(\phi^{|U|+m}) = O(\phi^{2r/(1-4r/n)}) \subset O(2.619^{r/(1-4r/n)}) \tag{10}$$

since $\phi^2 = 1 + \phi < 2.619$. By Lemma 3, the reduction algorithm takes $O(n^2)$ time, and then the total time complexity follows from that the for-loop in Algorithm 2 is executed n times. □

For $k \in \Theta(n^2)$, the time complexity can be expressed as follows, in which the condition $\delta \leq 0.185$ is to ensure the result is better than the naive $O^*(2^n)$-time algorithm.

Corollary 12 : For $k = \delta n^2$ with $\delta \leq 0.185$, Min-Sum 2-Clustering can be solved in $O(n \cdot 2.619^{\delta n/(1-4\delta)} + n^3)$ time.

When $k \in o(n^2)$ and n is sufficiently large, we have that $r = k/n \in o(n)$, and then $1/(1 - 4r/n) < 1 + \varepsilon$ for any constant $\varepsilon > 0$. The next corollary directly follows from Theorem 11.

Corollary 13: For $k \in o(n^2)$, Min-Sum 2-Clustering can be solved in $O(n \cdot 2.619^{k/n} + n^3)$ time.

If $k \in o(n^2)$, then $k/n \in o(n)$. Thus, by Corollary 13, Min-Sum 2-Clustering can be solved in $O^*(2^{o(n)})$ time, that is, in subexponential time.

Corollary 14 : Min-Sum 2-Clustering can be solved in polynomial time for $k \in O(n \log n)$.

5 Minimizing the sum of squares

Recall that $h_2(\pi) = \sum_v c_\pi^2(v)$ is the sum of squared conflict-numbers for a 2-partition π. Given a graph G and an integer k, Min-Square 2-Clustering determines if there exists a 2-partition π with $h_2(\pi) \leq k$. In this section, we show a parameterized algorithm for parameter k.

Lemma 15: If $h_2(\pi) \leq k$, then $\sum_v c_\pi(v) \leq \sqrt{nk}$ and there exists a vertex s with $c_\pi(s) \leq \sqrt{k/n}$.

Proof: By Cauchy-Schwarz inequality,

$$\left(\sum_v c_\pi(v)\right)^2 \leq \left(\sum_{i=1}^{n} 1^2\right)\left(\sum_v c_\pi^2(v)\right) = nh_2(\pi) \leq nk,$$

and we have that $\sum_v c_\pi(v) \le \sqrt{nk}$. The second consequence follows from $\min_v\{c_\pi(v)\} \le (1/n)\sum_v c_\pi(v)$. $\square$

Lemma 16: If π is a 2-partition such that $h_2(\pi) \le h_2(\pi \ominus v)$ for any vertex v, then $c_\pi(v) \le \sqrt{n(n-1)/2}$ for each vertex v.

Proof: Consider $\pi' = \pi \ominus v$ for any vertex v. First, $c_{\pi'}(v) = n - 1 - c_\pi(v)$. Let $Y(v) = C_\pi(v)$ and $\bar{Y}(v) = V \setminus C_\pi(v) \setminus \{v\}$. For each $u \in Y(v)$, $c_{\pi'}(u) = c_\pi(u) - 1$; and, for $u \in \bar{Y}(v)$, $c_{\pi'}(u) = c_\pi(u) + 1$. Therefore,

$$\begin{aligned}
& h_2(\pi) - h_2(\pi') \\
= \ & c_\pi^2(v) - (n-1-c_\pi(v))^2 + \sum_{u \in Y(v)} \left(c_\pi^2(u) - (c_\pi(u)-1)^2\right) \\
& + \sum_{u \in \bar{Y}(v)} \left(c_\pi^2(u) - (c_\pi(u)+1)^2\right) \\
= \ & 2(n-1)c_\pi(v) - (n-1)^2 + \sum_{u \in Y(v)} (2c_\pi(u) - 1) - \sum_{u \in \bar{Y}(v)} (2c_\pi(u)+1) \\
= \ & 2(n-1)c_\pi(v) - n(n-1) + 2\sum_{u \in Y(v)} c_\pi(u) - 2\sum_{u \in \bar{Y}(v)} c_\pi(u).
\end{aligned}$$

Since by the assumption $h_2(\pi) - h_2(\pi') \le 0$, we have that

$$c_\pi(v) \le \frac{n}{2} + \frac{1}{n-1}\left(\sum_{u \in \bar{Y}(v)} c_\pi(u) - \sum_{u \in Y(v)} c_\pi(u)\right). \tag{11}$$

Let $s = \arg\max_v\{c_\pi(v)\}$. By (11),

$$\begin{aligned}
c_\pi(s) & \le \frac{n}{2} + \frac{1}{n-1}\left(\sum_{u \in \bar{Y}(s)} c_\pi(u) - \sum_{u \in Y(s)} c_\pi(u)\right) \\
& \le \frac{n}{2} + \frac{1}{n-1}\left(\sum_{u \in \bar{Y}(s)} c_\pi(u)\right) \\
& \le \frac{n}{2} + \frac{|\bar{Y}(s)| \cdot c_\pi(s)}{n-1} \\
& = \frac{n}{2} + \frac{(n-1-c_\pi(s))c_\pi(s)}{n-1} = \frac{n}{2} + c_\pi(s) - \frac{c_\pi^2(s)}{n-1}.
\end{aligned}$$

That is, $c_\pi^2(s) \le n(n-1)/2$, and we obtain

$$c_\pi(s) \le \sqrt{\frac{n(n-1)}{2}}. \tag{12}$$

$\square$

Similar to Lemma 8, we have the next corollary from Lemmas 15 and 16. Recall that a 2-partition is extreme if one of the two clusters is singleton.

Algorithm 4 Search-tree algorithm for MIN-SQUARE 2-CLUSTERING

```
Input: a set $U$ of undetermined vertices, a flipping quota $m$, and a 2-partition $\pi$.
  procedure SEARCH2($U, m, \pi$)
      if $U = \emptyset$ or $m = 0$ then                       ▷ no vertex can be flipped
          compute $h_2(\pi)$ and return True or False accordingly;
      end if
      pick an arbitrary vertex $u \in U$;
      if SEARCH2($U \setminus \{u\}, m, \pi$)=True then        ▷ move $u$ to $X$ without flipping
          return True;
      else if SEARCH2($U \setminus \{u\}, m-1, \pi \ominus u$)=True then    ▷ flip and move $u$ to $X$
          return True;
      else
          return False;
      end if
  end procedure
```

Corollary 17 : Suppose that the h_2 cost of any extreme 2-partition is larger than k. If there exists a non-trivial and non-extreme 2-partition π with $h_2(\pi) \leq k$, then there exists a (K,t)-feasible flipping set F for π_0 of size at most f, where $\pi_0 = (N_G[s], V \setminus N_G[s])$ for some vertex s, $K = \sqrt{nk}$, $t = \sqrt{n(n-1)/2}$, and $f = \sqrt{k/n}$.

The main steps of the algorithm for MIN-SQUARE 2-CLUSTERING are quite similar to Algorithm 2 in the previous section. First, an extreme 2-partition with minimum cost h_2 can be found in $O(n^3)$ time since there are only n extreme 2-partitions. Then, we can focus on non-extreme 2-partitions. By Corollary 17, we use the reduction algorithm with the bound of total conflict number $K = \sqrt{nk}$, individual bound $t = \sqrt{n(n-1)/2}$ and flipping quota $f = \sqrt{k/n}$. The assumption $t + 2f < n$ in Lemma 3 is satisfied when $k \leq 0.021n^3$, and the largest value of k we will use in this section is $0.0118n^3$ (Corollary 19). For the reduced instance U and m, a search-tree algorithm is employed to check if any desired 2-partition can be resulted from flipping a subset of U with size at most m. The recursive procedure is shown in Algorithm 4. Unlike Algorithm 3, we did not find a way to compute the cost h_2 with time complexity only depending on $|U|$ but not on n. Therefore it takes $O(n^2)$ time for each recursive call, and the time complexity of the search-tree algorithm is $O(n^2)$ multiplied by the number of recursive calls.

Theorem 18: For $k \leq 0.021n^3$, MIN-SQUARE 2-CLUSTERING can be solved in $O(n^3 \cdot 5.171^{\theta/(1-(4+2\sqrt{2})\theta/n)})$, where $\theta = \sqrt{k/n}$.

Proof: By the reduction algorithm,

$$\begin{aligned}
|U| &\leq \frac{K}{n-t-2m} - m \\
&= \frac{\sqrt{nk}}{n - \sqrt{n(n-1)/2} - 2m} - m \\
&\leq \frac{\sqrt{nk}}{(1-1/\sqrt{2})n - 2m} - m \\
&= \frac{(2+\sqrt{2})\theta}{1-(4+2\sqrt{2})m/n} - m. \qquad (13)
\end{aligned}$$

Let $T_0(a,b)$ denote the number of leaf nodes in the search tree explored by the algorithm SEARCH2 with $|U| = a$ and $m = b$. According to the branching rule,

$$T_0(a,b) \le \begin{cases} T_0(a-1,b) + T_0(a-1,b-1) & \text{if } a,b > 0; \\ 1 & \text{otherwise.} \end{cases} \tag{14}$$

Next we show by induction that

$$T_0(a,b) \le \phi^{a+b}. \tag{15}$$

It is clear that (15) holds when $a = 0$ or $b = 0$. Suppose by the induction hypothesis that it holds for $T_0(a-1,b)$ and $T_0(a-1,b-1)$. Then, for $a,b > 0$,

$$T_0(a,b) = T_0(a-1,b) + T_0(a-1,b-1) \le \phi^{a+b-1} + \phi^{a+b-2} = \phi^{a+b}$$

by the identity $\phi^2 = \phi + 1$. Since the search tree is binary, the number of recursive calls is bounded by $2T_0(|U|,m) \in O(\phi^{|U|+m})$. By (13), since $m \le \sqrt{k/n} = \theta$,

$$\begin{aligned} |U| + m &\le \frac{(2+\sqrt{2})\theta}{1-(4+2\sqrt{2})m/n} \\ &\le \frac{(2+\sqrt{2})\theta}{1-(4+2\sqrt{2})\theta/n}. \end{aligned}$$

Similarly to Algorithm 2, the reduction and the search-tree algorithms are executed n times. Since each recursive call take $O(n^2)$ time, the total time complexity is $O(n^3 \cdot \phi^{|U|+m}) \subset O(n^3 \cdot 5.171^{\theta/(1-(4+2\sqrt{2})\theta/n)})$. $\square$

For $k \in \Theta(n^3)$, the time complexity can be expressed as follows, in which the condition $k \le 0.0118n^3$ is to ensure the result is better than the naive $O^*(2^n)$-time algorithm.

Corollary 19 : For $k = \delta^2 n^3$ with $\delta^2 \le 0.0118$, MIN-SQUARE 2-CLUSTERING can be solved in $O(n^3 \cdot 5.171^{\delta n/(1-(4+2\sqrt{2})\delta)})$.

When $k \in o(n^3)$ and n is sufficiently large, we have that $\theta = \sqrt{k/n} \in o(n)$, and then $1/(1-(4+2\sqrt{2})\theta/n) < 1+\varepsilon$ for any constant $\varepsilon > 0$. The next corollary directly follows from Theorem 18, which implies that MIN-SQUARE 2-CLUSTERING can be solved in subexponential time for $k \in o(n^3)$.

Corollary 20 : For $k \in o(n^3)$, MIN-SQUARE 2-CLUSTERING can be solved in $O(n^3 \cdot 5.171^{\theta})$ time, where $\theta = \sqrt{k/n}$.

Corollary 21 : For $k \in O(n \log^2 n)$, MIN-SQUARE 2-CLUSTERING can be solved in polynomial time.

6 Concluding remarks

In this paper, we show parameterized algorithms for MIN-SUM 2-CLUSTERING and MIN-SQUARE 2-CLUSTERING. The first problem is the same as 2-CLUSTER EDITING in the literature with an additional multiplicative factor two in the cost function. The proposed algorithms run in subexponential time and significantly improve the brute-force algorithm when k is relatively small, i.e., $k \in o(n^2)$ for MIN-SUM 2-CLUSTERING and $k \in o(n^3)$ for MIN-SQUARE 2-CLUSTERING.

The time complexities of the search-tree algorithms are shown by induction in Lemma 10 and Theorem 18. It can be shown that the time complexities of both algorithms are almost tight when k is relatively small. We shall show the case of Algorithm 3 with $k \in o(n^2)$, and it is similar for Algorithm 4 with $k \in o(n^3)$. In worst case, the search-tree algorithm explores all subsets of U with sizes at most m. The number of recursive calls is lower bounded by $\sum_{i=0}^{m} \binom{|U|}{i}$. From (9), when $k \in o(n^2)$ and $m = k/n$, we have that $m \in \Theta(|U|)$. Let $m = \alpha|U|$ for some constant α. By [15, Lemma 3.13],

$$\sum_{i=0}^{m} \binom{|U|}{i} \geq p|U|^{-1/2} \cdot 2^{H(\alpha)|U|},$$

where p is a constant and $H(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$ is the *binary entropy function*. When $\alpha = \phi^{-2} = (3-\sqrt{5})/2$, it can be verified that $2^{H(\alpha)|U|} = \phi^{|U|+m}$.

The lower bounds for the two problems are interesting open problems. Another straightforward question is how to generalize the algorithms to the case of more than two clusters. However, the problem seems to become much more difficult when the number of clusters is more than two. In fact, by the following simple transformation, we can show that there is no algorithm solving 3-CLUSTER EDITING in $O^*(2^{k/n})$ time unless NP=P, where k and n are the cost bound and the number of vertices. Let (G', k) be an instance of the NP-complete 2-CLUSTER EDITING problem. We construct G from G' by adding an isolated clique of $k+2$ vertices. One can observe that G can be edited into 3 clusters with cost k if and only if G' can be edited into 2 clusters with the same cost. Since $n > k$, an algorithm solving 3-CLUSTER EDITING in $O^*(2^{k/n})$ time can also solve the NP-complete 2-CLUSTER EDITING problem in polynomial time.

acknowledgements

The authors would like to thank the anonymous referees for their helpful comments which improved the presentation significantly. This work was supported in part by NSC 100-2221-E-194-036-MY3 and NSC 101-2221-E-194-025-MY3 from the National Science Council, Taiwan.

References

[1] Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: Ranking and clustering. Journal of the ACM **55**(5), 23:1–23:27 (2008)

[2] Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning **56**, 89–113 (2004)

[3] Böcker, S., Briesemeister, S., Bui, Q., Truss, A.: Going weighted: Parameterized algorithms for cluster editing. Theoretical Computer Science **410**(52), 5467 – 5480 (2009)

[4] Böcker, S., Damaschke, P.: Even faster parameterized cluster deletion and cluster editing. Information Processing Letters **111**(14), 717 – 721 (2011)

[5] Bonizzoni, P., Vedova, G.D., Dondi, R.: A PTAS for the minimum consensus clustering problem with a fixed number of clusters. In: Eleventh Italian Conference on Theoretical Computer Science (2009)

[6] Bonizzoni, P., Vedova, G.D., Dondi, R., Jiang, T.: On the approximation of correlation clustering and consensus clustering. Journal of Computer and System Sciences **74**(5), 671 – 696 (2008)

[7] Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. Journal of Computer and System Sciences **71**(3), 360 – 383 (2005)

[8] Chen, J., Meng, J.: A $2k$ kernel for the cluster editing problem. Journal of Computer and System Sciences **78**(1), 211 – 220 (2012)

[9] Chen, L.H., Chang, M.S., Wang, C.C., Wu, B.Y.: On the min-max 2-cluster editing problem. Journal of Information Science and Engineering **29**, 1109–1120 (2013)

[10] Damaschke, P.: Bounded-degree techniques accelerate some parameterized graph algorithms. In: J. Chen, F. Fomin (eds.) Parameterized and Exact Computation, *Lecture Notes in Computer Science*, vol. 5917, pp. 98–109. Springer Berlin Heidelberg (2009)

[11] Damaschke, P.: Fixed-parameter enumerability of cluster editing and related problems. Theory of Computing Systems **46**, 261–283 (2010)

[12] Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer-Verlag (1999)

[13] Fellows, M.R., Guo, J., Komusiewicz, C., Niedermeier, R., Uhlmann, J.: Graph-based data clustering with overlaps. Discrete Optimization **8**(1), 2 – 17 (2011)

[14] Filkov, V., Skiena, S.: Integrating microarray data by consensus clustering. International Journal on Artificial Intelligence Tools **13**(04), 863–880 (2004)

[15] Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Springer (2010)

[16] Fomin, F.V., Kratsch, S., Pilipczuk, M., Pilipczuk, M., Villanger, Y.: Tight bounds for Parameterized Complexity of Cluster Editing. In: N. Portier, T. Wilke (eds.) 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013), *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 20, pp. 32–43. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013)

[17] Giotis, I., Guruswami, V.: Correlation clustering with a fixed number of clusters. Theory of Computing **2**(13), 249 – 266 (2006)

[18] Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In: R. Petreschi, G. Persiano, R. Silvestri (eds.) Algorithms and Complexity, *Lecture Notes in Computer Science*, vol. 2653, pp. 108–119. Springer Berlin Heidelberg (2003)

[19] Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Automated generation of search tree algorithms for hard graph modification problems. Algorithmica **39**, 321–347 (2004)

[20] Guo, J.: A more effective linear kernelization for cluster editing. Theoretical Computer Science **410**(810), 718 – 726 (2009)

[21] Harary, F.: On the notion of balance of a signed graph. The Michigan Mathematical Journal **2**(2), 143–146 (1953)

[22] Hüffner, F., Betzler, N., Niedermeier, R.: Optimal edge deletions for signed graph balancing. In: C. Demetrescu (ed.) Experimental Algorithms, *Lecture Notes in Computer Science*, vol. 4525, pp. 297–310. Springer Berlin Heidelberg (2007)

[23] Hüffner, F., Komusiewicz, C., Moser, H., Niedermeier, R.: Fixed-parameter algorithms for cluster vertex deletion. Theory of Computing Systems **47**, 196–217 (2010)

[24] Komusiewicz, C., Uhlmann, J.: Cluster editing with locally bounded modifications. Discrete Applied Mathematics **160**(15), 2259 – 2270 (2012)

[25] Niedermeier, R., Rossmanith, P.: A general method to speed up fixed-parameter-tractable algorithms. Information Processing Letters **73**(34), 125 – 129 (2000)

[26] Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. Discrete Applied Mathematics **144**(12), 173 – 182 (2004)

[27] Wasserman, S., Faust, K.: Social network analysis: Methods and applications, vol. 8. Cambridge university press (1994)