On the Parameterized Complexity of Consensus Clustering[⋆]

Martin Dörnfelder[1], Jiong Guo[1], Christian Komusiewicz[2], Mathias Weller[2]

[1] Universität des Saarlandes,
Campus E 1.7, D-66123 Saarbrücken, Germany.
`{mdoernfe,jguo}@mmci.uni-saarland.de`

[2] Institut für Softwaretechnik und Theoretische Informatik,
Technische Universität Berlin, D-10587 Berlin, Germany
`{christian.komusiewicz, mathias.weller}@tu-berlin.de`

Abstract. Given a collection $\mathcal{C}$ of partitions of a base set S, the NP-hard Consensus Clustering problem asks for a partition of S which has a total Mirkin distance of at most t to the partitions in $\mathcal{C}$, where t is a non-negative integer. We present a parameterized algorithm for Consensus Clustering with running time $O(4.24^k \cdot k^3 + |\mathcal{C}| \cdot |S|^2)$, where $k := t/|\mathcal{C}|$ is the average Mirkin distance of the solution partition to the partitions of $\mathcal{C}$. Furthermore, we strengthen previous hardness results for Consensus Clustering, showing that Consensus Clustering remains NP-hard even when all input partitions contain at most two subsets. Finally, we study a local search variant of Consensus Clustering, showing W[1]-hardness for the parameter "radius of the Mirkin-distance neighborhood". In the process, we also consider a local search variant of the related Cluster Editing problem, showing W[1]-hardness for the parameter "radius of the edge modification neighborhood".

1 Introduction

The NP-hard Consensus Clustering problem aims at reconciling the information that is contained in multiple clusterings of a base set S. More precisely, the input of a Consensus Clustering instance is a multi-set $\mathcal{C}$ of partitions of a base set S into subsets, also referred to as *clusters*, and the aim is to find a partition of S that is similar to $\mathcal{C}$. Herein, the similarity between two partitions is measured as follows. Two elements $a, b \in S$ are *co-clustered* in a partition C of S, if a and b are in the same cluster of C, and *anti-clustered*, if a and b are in different clusters of C. For two partitions C and C' of S and a pair of elements $a, b \in S$, let $\delta_{\{C,C'\}}(a, b) = 1$ if a and b are anti-clustered in C and co-clustered in C' or vice versa, and $\delta_{\{C,C'\}}(a, b) = 0$, otherwise. Then, the *Mirkin distance* $\operatorname{dist}(C, C') := \sum_{\{a,b\}\subseteq S} \delta_{\{C,C'\}}(a, b)$ between two partitions C and C' of S is the number of pairs $a, b \in S$ that are clustered "differently" by C and C'.

[⋆] Supported by the DFG Excellence Cluster on Multimodal Computing and Interaction (MMCI) and DFG project DARE (NI 369/11).

To appear in Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC'11), *Yokohama, Japan, Decemer 2011, Volume 7074 of LNCS.* © *Springer*

The *total Mirkin distance* between a partition C and a multi-set $\mathcal{C}$ of partitions is defined as $\operatorname{dist}(C, \mathcal{C}) := \sum_{C' \in \mathcal{C}} \operatorname{dist}(C, C')$. Altogether, the Consensus Clustering problem is defined as follows.

> **Input:** A multi-set of partitions $\mathcal{C} = (C_1, \ldots, C_n)$ of a base set $S = \{1, 2, \ldots, m\}$ and an integer $t \geq 0$.
> **Question:** Is there a partition C of S with $\operatorname{dist}(C, \mathcal{C}) \leq t$?

Consensus Clustering has applications, for instance, in gene expression data analysis [13], clustering categorical data, improving clustering robustness, and preserving privacy [9]. The NP-hardness of Consensus Clustering was shown several times [12,16]. For $n = 2$, that is, with two input partitions, it is solvable in polynomial time: either input partition minimizes t. In contrast, already for $n = 3$ minimizing t is APX-hard [5]. The variant of Consensus Clustering where the output partition is required to have at most $d \geq 2$ subsets, d being a constant, is NP-hard for every $d \geq 2$ [4] but it admits a PTAS for minimizing t [4,6,11]. Various heuristics for Consensus Clustering have been experimentally evaluated [2,10]. Consensus Clustering is closely related to Cluster Editing [15], also known as Correlation Clustering [1].

Until now, the study of the parameterized complexity [7,8,14] of Consensus Clustering seems to be neglected. One reason for this might be the lack of an obvious reasonable parameter for this problem: First, the overall Mirkin distance of solutions is usually not small in practice: every element pair that is co-clustered in at least one partition and anti-clustered in at least one other partition contributes at least one to this parameter. Second, Consensus Clustering is trivially fixed-parameter tractable with respect to the number m of elements but m is also unlikely to take small values in real-world instances. Finally, Consensus Clustering is NP-hard for $n = 3$, ruling out fixed-parameter tractability with respect to n. Betzler et al. [3] considered the parameter "average Mirkin distance p between the input partitions", that is, $p := \sum_{i \neq j} \operatorname{dist}(C_i, C_j)/(n(n-1))$, and presented a "partial kernelization" for this parameter. More precisely, they presented a set of polynomial-time data reduction rules whose application yields an instance with $|S| = O(p)$ [3]. Then, checking all possible partitions of S gives an optimal solution, resulting in a fixed-parameter algorithm for the parameter p. Since the Mirkin distance is a metric, the average Mirkin distance of solution partitions $k := t/n$ is at least $p/2$ [3]. Hence, the above also implies fixed-parameter tractability with respect to k. However, the brute-force check of all possible partitions of S leads to an impractical running time of roughly $2^{O(k \log k)} \operatorname{poly}(n, m)$.

Motivated by these observations, we study several parameterizations of Consensus Clustering. First, we complement the partial kernelization result by presenting a search tree algorithm with running time $O(4.24^k \cdot k^3 + nm^2)$. Second, we consider the parameter "maximal number of clusters in any input partition". We show that Consensus Clustering remains NP-hard even if every input partition consists of at most two clusters, ruling out fixed-parameter tractability for this parameter. We also strengthen the result of Bonizzoni et al. [4] by showing that, even if all input partitions contain at most two clusters, seeking a solution partition with at most two clusters remains NP-hard. Finally, we consider a local search variant of Consensus Clustering showing that, given in addition to $\mathcal{C}$ and S a partition C of S, the problem of deciding whether there is a partition C' such that $\text{dist}(C', \mathcal{C}) < \text{dist}(C, \mathcal{C})$ and $\text{dist}(C', C) \leq d$ for some integer $d \geq 0$, is W[1]-hard with respect to d. Moreover, we also show W[1]-hardness of a local search variant of Cluster Editing. Due to the lack of space, several details are deferred to a full version of this article.

Preliminaries. Given a base set S and a multi-set $\mathcal{C}$ of partitions of S, let $n := |\mathcal{C}|$ and $m := |S|$. We use $\text{co}(a, b)$ for $a, b \in S$ to denote the number of partitions in $\mathcal{C}$ where a and b are co-clustered and use $\text{anti}(a, b)$ to denote the number of partitions where a and b are anti-clustered. Clearly, $n = \text{co}(a, b) + \text{anti}(a, b)$. For a partition C of S and elements $a, b \in S$, the function $\text{dist}_C(a, b)$ is defined as the number of partitions in $\mathcal{C}$ in which a, b are clustered in a different way than in C. More precisely, if a and b are co-clustered in C, then $\text{dist}_C(a, b) = \text{anti}(a, b)$; otherwise, $\text{dist}_C(a, b) = \text{co}(a, b)$. Clearly, $\text{dist}(C, \mathcal{C}) = 1/2 \cdot \sum_{\{a,b\} \subseteq S} \text{dist}_C(a, b)$.

2 A Search Tree Algorithm for the Average Mirkin Distance

In this section, we present a search tree algorithm for Consensus Clustering parameterized by the average Mirkin distance $k := t/n$ of a solution partition to the set of input partitions $\mathcal{C}$. The main idea of this search tree algorithm follows the standard paradigm of branching algorithms in parameterized algorithmics: branch into a bounded number of cases and decrease the parameter in each case. The difficulty for using this approach for the parameter k lies in the fact that for a pair of elements $a, b \in S$ the values of either $\text{co}(a, b)$ or $\text{anti}(a, b)$ can be arbitrarily small for increasing n. When branching on such element pairs, the parameter might not really decrease in some cases. We circumvent this problem by finding a way to always branch into at most two cases, decreasing k by at least $1/3$ in each case. In the following, we describe this approach in detail.

Description of the algorithm. The algorithm consists of two phases, the first phase is a search tree algorithm and the second phase is a polynomial-time algorithm solving the remaining instances at the leaves of the search tree. Each node v of the search tree is associated with a partition C_v of S, called "temporary solution", and a list L_v, called "separation list", that contains pairs of subsets in C_v. These two data structures restrict the partitions we are seeking in the subtree rooted at v: The temporary solution C_v requires that the elements co-clustered by C_v will remain co-clustered in the final solution sought for. The separation list L_v requires that in the solution for each pair $\{K_1, K_2\} \in L_v$ the elements in K_1 are in different subsets than the elements in K_2. In each search tree node, we keep track of the average Mirkin distance that is already caused by the constraints of C_v and L_v. To this end, consider the following.

Let U be the set of all unordered pairs $\{a, b\}$ of elements $a, b \in S$ with $a \neq b$. Based on C_v and L_v, we divide U into two subsets. The first subset U_v^1

contains the "resolved pairs", that is, the pairs of elements that are either co-clustered in C_v or contained in two subsets forming a pair in L_v. The other subset $U_v^2 := U \setminus U_v^1$ contains the "unresolved pairs". Then, each node carries a rational number k_v, called the "average Mirkin distance bound for unresolved pairs", which means that in the subtree rooted at v we seek only partitions C with $1/n \cdot \sum_{\{a,b\} \in U_v^2} \mathrm{dist}_C(a,b) \leq k_v$.

At the root r of the search tree, we start with the partition $C_r := \{\{i\} \mid i \in S\}$ where all elements are in distinct sets, an empty separation list, and $k_r = k = t/n$. At every node v of the search tree, we branch into two cases, each performing one of the following two operations on two subsets X_i and X_j in C_v. One operation "merges" X_i and X_j, that is, it removes X_i and X_j from C_v and adds $X_i \cup X_j$ to C_v. The other operation "separates" X_i and X_j, that is, it adds the subset pair $\{X_i, X_j\}$ to the separation list L_v.

To give a formal description of the search tree we introduce the following notations. Let X and Y be two subsets of S that are contained in the temporary solution C_v of a search tree node v, and let L_v be the separation list of v. If X and Y do not form a pair in L_v, then we define $\mathrm{co}_v(X,Y) := \sum_{a \in X} \sum_{b \in Y} \mathrm{co}(a,b)$ and $\mathrm{anti}_v(X,Y) := \sum_{a \in X} \sum_{b \in Y} \mathrm{anti}(a,b)$; otherwise, we set $\mathrm{co}_v(X,Y) := 0$ and $\mathrm{anti}_v(X,Y) := \infty$. Moreover, we say that the predicate $(XY)_v$ is true iff $\mathrm{anti}_v(X,Y) < n/3$, $X \not\leftrightarrow_v Y$ is true iff $\mathrm{co}_v(X,Y) < n/3$, and $X \#_v Y$ is true iff $\mathrm{co}_v(X,Y) \geq n/3$ and $\mathrm{anti}_v(X,Y) \geq n/3$. If $X \#_v Y$ holds, we call X and Y a *dirty subset pair*. Three subsets X, Y, and Z are called a *dirty subset triple*, if $(XY)_v$, $(YZ)_v$, and $X \not\leftrightarrow_v Z$ are true.

The search tree algorithm uses two branching rules, the *dirty pair rule* and the *dirty triple rule*. In the following, let v denote the node of the search tree in which the rules are applied. Both rules branch into two cases, referred to as v_1 and v_2. Furthermore, branching into case v_1 (case v_2) is only performed if $k_{v_1} \geq 0$ ($k_{v_2} \geq 0$); we refer to this as the *stop criterion*.

Branching Rule 1 (Dirty pair rule) *If C_v contains two subsets X and Y with $X \#_v Y$, then branch into the following two cases.*

- *Case v_1: merge X and Y and set $k_{v_1} := k_v - 1/n \cdot \mathrm{anti}_v(X,Y)$.*
- *Case v_2: separate X and Y and set $k_{v_2} := k_v - 1/n \cdot \mathrm{co}_v(X,Y)$.*

In case the dirty pair rule is not applicable, because there is no dirty pair, we apply the following rule.

Branching Rule 2 (Dirty triple rule) *If C_v contains three subsets X, Y, and Z such that $(XY)_v$, $(YZ)_v$, and $X \not\leftrightarrow_v Z$, then branch into the following two cases.*

- *Case v_1: separate X and Y and set $k_{v_1} := k_v - 1/n \cdot \mathrm{co}_v(X,Y)$.*
- *Case v_2: separate Y and Z and set $k_{v_2} := k_v - 1/n \cdot \mathrm{co}_v(Y,Z)$.*

We call a search tree node in which neither branching rule can be applied a *leaf* of the search tree. At the leaves the algorithm enters its second phase in which the temporary solution C_v is modified into a complete solution as follows.

As long as possible, merge all subset pairs X and Y for which $(XY)_v$ holds and, after each merge operation, update $k_v := k_v - 1/n \cdot \mathrm{anti}_v(X, Y)$. Afterwards, if $k_v \geq 0$ then output C_v (or, alternatively, answer "yes") and terminate the algorithm. If there is no search tree leaf in which a partition is output, then answer "no".

Correctness of the algorithm. We now show the correctness of the algorithm. In the following, a partition C of S *satisfies* the restrictions of a search tree node v if C fulfills the following three conditions:

(C1) for every subset $X \in C_v$, there is a subset $Z \in C$ with $X \subseteq Z$,
(C2) for every pair $\{X, Y\}$ in L_v, there are two subsets $Z, Z' \in C$ with $Z \neq Z'$, $X \subseteq Z$, and $Y \subseteq Z'$, and
(C3) $1/n \cdot \sum_{\{a,b\} \in U_v^2} \mathrm{dist}_C(a, b) \leq k_v$.

We say that a branching rule is *sound* if each partition C satisfying the restrictions of a node v, satisfies the restrictions of one of the child nodes created by applying this rule to v.

Lemma 1. *Both branching rules are sound.*

The following lemma shows the correctness of the second phase of the algorithm. More precisely, it states that the operations performed in a leaf v of the search tree yield a partition C that, of all partitions satisfying the restrictions of v, has minimum distance to the input partitions.

Lemma 2. *Let v be a leaf of the search tree, and let $\mathcal{D}$ be the set of partitions satisfying the restrictions of v. Then, there exists a partition $C \in \mathcal{D}$ such that* $\mathrm{dist}(C, \mathcal{C}) = \min_{C' \in \mathcal{D}} \mathrm{dist}(C', \mathcal{C})$ *and for all $X, Y \in C_v$, the following holds:*

(a) If $(XY)_v$, then there is a subset $Z \in C$ with $X \subseteq Z$ and $Y \subseteq Z$.
(b) If $X \leftrightarrow_v Y$, then there are two subsets $Z, Z' \in C$ with $Z \neq Z'$, $X \subseteq Z$, and $Y \subseteq Z'$.

Altogether, this implies the following.

Proposition 1. *The algorithm is correct.*

Running time analysis. Next, we bound the running time of the algorithm. The exponential part of the running time clearly depends on the size of the search tree, that is, on the number of search tree nodes. A rough estimation of this size is as follows.

At each node v of the search tree, we either merge or separate two subsets $X, Y \in C_v$. Both operations cause a decrease of the average Mirkin distance bound k_v. More precisely, the dirty pair rule decreases k_v either by $1/n \cdot \mathrm{anti}_v(X, Y)$ or $1/n \cdot \mathrm{co}_v(X, Y)$. Since X and Y form a dirty subset pair, k_v is decreased by at least $1/3$ in both cases. Branching on a dirty triple X, Y, and Z with $(XY)_v$, $(YZ)_v$, and $X \leftrightarrow_v Z$ causes separation of X and Y in one case and separation of Y and Z in the other case. The bound k_v is decreased by $1/n \cdot \mathrm{co}_v(X,Y)$ and $1/n \cdot \mathrm{co}_v(Y,Z)$, respectively. Since $(XY)_v$ and $(YZ)_v$ hold, the distance bound k_v is decreased by at least $2/3$ in both cases. Since $k_v < 0$ is a stop criterion for the search tree, the size of the tree is thus $O(2^{3k}) = O(8^k)$. Using this simple analysis, one obtains a running time bound of $8^k \cdot \mathrm{poly}(n,m)$. In the following, we give a more detailed analysis of the search tree size. In the proof, we use $\phi := (1+\sqrt{5})/2$ to denote the golden ratio.

Theorem 1. Consensus Clustering *can be solved in* $O(4.24^k \cdot k^3 + nm^2)$ *time.*

Proof. We show only the size of the search tree, the proof of the polynomial running time part is deferred to a long version of this article. More precisely, we show that the search tree has size at most $(2/\sqrt{5})\phi^{3k+2} - 1$. To this end, we consider an arbitrary node v in the tree and estimate the size of the subtree rooted at v. Clearly, $k_v \geq 0$ for every node v in the tree. We consider three cases for the value of k_v and prove the size bound for each case.

Case 1: $0 \leq k_v < 1/3$**.** Then, the two cases created by applying one of the two branching rules both have average Mirkin distance bounds at most $k_v - 1/3 < 0$. Hence, the search tree has only one node. Since $3k_v + 2 \geq 2$ for $k_v \geq 0$, it holds that $(2/\sqrt{5})\phi^{3k_v+2} - 1 \geq (2/\sqrt{5})\phi^2 - 1 = \frac{1+2\sqrt{5}+5}{2\sqrt{5}} - 1 = \frac{3}{\sqrt{5}} > 1$. Thus, the claimed search tree size bound holds in this case.

Case 2: $1/3 \leq k_v < 1/2$**.** If the dirty triple rule is applied, then k_v is decreased by at least $2/3$ in both cases. Therefore, the rule creates no child node for v, and the claimed search tree size bound holds as shown above. If the dirty pair rule is applied to a dirty subset pair $X, Y \in C_v$, then k_v is decreased by $1/n \cdot \mathrm{co}_v(X,Y)$ in one case and by $1/n \cdot \mathrm{anti}_v(X,Y)$ in the other case. Since $\mathrm{co}_v(X,Y) + \mathrm{anti}_v(X,Y) = |X| \cdot |Y| \cdot n$, at least one of $1/n \cdot \mathrm{co}_v(X,Y)$ and $1/n \cdot \mathrm{anti}_v(X,Y)$ is greater than $1/2$. Consequently, v has at most one child node. Since Case 1 applies to this child node, the subtree rooted at v contains at most two nodes. Since $3k_v + 2 \geq 3$ for $k \geq 1/3$ and

$$(2/\sqrt{5})\phi^3 - 1 = \frac{1 + 3\sqrt{5} + 15 + 5\sqrt{5}}{4\sqrt{5}} - 1 = \frac{4 + 2\sqrt{5}}{\sqrt{5}} - 1 = \frac{4}{\sqrt{5}} + 1 > 2,$$

the claimed search tree size bound holds for this case.

Case 3: $k_v \geq 1/2$**.** As argued above, the dirty pair rule creates at most two child nodes, v_1 with $k_{v_1} := k_v - 1/n \cdot \mathrm{co}_v(X,Y)$ and v_2 with $k_{v_2} := k_v - 1/n \cdot \mathrm{anti}_v(X,Y)$, while the dirty triple rule adds at most two child nodes, v_1 with $k_{v_1} := k_v - 1/n \cdot \mathrm{co}_v(X,Y)$ and v_2 with $k_{v_2} := k_v - 1/n \cdot \mathrm{co}_v(Y,Z)$. Since $\mathrm{co}_v(X,Y) + \mathrm{anti}_v(X,Y) = |X| \cdot |Y| \cdot n$, we have $\mathrm{anti}_v(X,Y) \geq n/3$ and $\mathrm{co}_v(X,Y) \geq n/3$ for a dirty subset pair X, Y. Therefore, we have $k_{v_1} \leq k_v - \alpha$ and $k_{v_2} \leq k_v - 1 + \alpha$ for some α with $1/3 \leq \alpha < 2/3$. Due to symmetry, we can assume $1/3 \leq \alpha \leq 1/2$. Moreover, for the dirty subset triple, we have $(XY)_v$ and $(YZ)_v$, implying $1/n \cdot \mathrm{co}_v(X,Y) \geq 2/3$ and $1/n \cdot \mathrm{co}_v(Y,Z) \geq 2/3$. As the function $(2/\sqrt{5})\phi^{3k+2} - 1$ is monotonically increasing on k, we can use $k_{v_1} = k_v - \alpha$ and $k_{v_2} = k_v - 1 + \alpha$ with $1/3 \leq \alpha \leq 1/2$ to obtain an upper bound on the size of the subtree rooted at v, that is, in our analysis the worst-case search tree size bound is obtained for the dirty pair rule. Assume, by an inductive argument, that the search tree size bound holds for all $k' < k_v$. Clearly, the size of the subtree rooted at v is at most

$$\left[(2/\sqrt{5})\phi^{3(k_v-1+\alpha)+2}-1\right]+\left[(2/\sqrt{5})\phi^{3(k_v-\alpha)+2}-1\right]+1.$$

We differentiate this bound with respect to α to find local extrema:

$$\begin{aligned}&\tfrac{d}{d\alpha}(2/\sqrt{5})\phi^{3(k_v-1+\alpha)+2}+(2/\sqrt{5})\phi^{3(k_v-\alpha)+2}-1\\ =&\tfrac{6}{\sqrt{5}}\log\left(\tfrac{2}{1+\sqrt{5}}\right)\left[\phi^{3(k_v-1+\alpha)}-\phi^{3(k_v-\alpha)}\right].\end{aligned}$$

This term equals zero only if $k_v-1+\alpha=k_v-\alpha$, that is, if $\alpha=1/2$. Thus, the candidates for the maximum are the critical point $\alpha=1/2$ and the endpoints of the interval $[1/3,1/2]$. For $\alpha=1/2$, the search tree has size at most $(4/\sqrt{5})\cdot\phi^{3k_v+1/2}-1$ and for $\alpha=1/3$ the search tree has size at most $(2/\sqrt{5})\cdot\phi^{3k_v+2}-1$. Hence, the claimed search tree size bound holds in this case as well.

Summarizing, the upper bound of $(2/\sqrt{5})\phi^{3k_v+2}-1$ holds for all $k_v\geq 0$ and thus we can construct the search tree in $O(\phi^{3k+2}\cdot k^3)=O(4.24^k\cdot k^3)$ time. The overall running time bound follows. □

3 NP-Hardness for Input Partitions with a Bounded Number of Clusters

Bonizzoni et al. [4] proved that the variation of Consensus Clustering in which the solution C is required to contain at most d clusters, is NP-hard for every $d\geq 2$. First, we consider—instead of *solution partitions* with a bounded number of clusters—instances $(\mathcal{C},t)$ in which each *input partition* has at most d' clusters, that is, $d':=\max_{C\in\mathcal{C}}|C|$. We show that Consensus Clustering is fixed-parameter intractable with respect to d' by proving the following.

Theorem 2. Consensus Clustering *remains NP-hard, even if all input partitions have at most two subsets.*

Next, we strengthen the hardness result of Bonizzoni et al. [4] by showing that it is NP-hard to find *solution partitions* with at most two clusters even if every input partition has at most two clusters.

Consensus Clustering with 2-Partitions (CC2P)
Input: A multi-set of partitions $\mathcal{C}=(C_1,\dots,C_n)$ of a base set $S=\{1,2,\dots,m\}$, where $|C_i|\leq 2$ for all $1\leq i\leq n$, and an integer $t\geq 0$.
Question: Is there a partition C of S with $|C|\leq 2$ and $\text{dist}(C,\mathcal{C})\leq t$?

Theorem 3. *CC2P is NP-hard.*

With introducing some dummy elements, one can then easily prove that Consensus Clustering is also NP-hard if the input partitions have at most $d\geq 3$ subsets and we ask for a partition with at most d subsets. Moreover, for every $d\geq 2$, a similar reduction can be used to show the NP-hardness in case the input partitions contain *exactly* d subsets and we ask for a partition with *exactly* d subsets.

4 Hardness of Local Search

In this section, we study the parameterized complexity of the following local search variant of Consensus Clustering:

Consensus Clustering with Mirkin-Local Search (CCML)
Input: A multi-set $\mathcal{C} = (C_1, \ldots, C_n)$ of partitions of a base set $S = \{1, 2, \ldots, m\}$, a partition C of S, a nonnegative integer d.
Question: Is there a partition C' of S such that $\text{dist}(C', \mathcal{C}) < \text{dist}(C, \mathcal{C})$ and $\text{dist}(C, C') \leq d$?

The study of CCML is motivated as follows. For a given multi-set $\mathcal{C}$ of input partitions, a partition that has an average Mirkin distance at most k to $\mathcal{C}$ trivially has Mirkin distance at most k to at least one of the input partitions. Moreover, it could be that there is one input partition to which this optimal partition has a Mirkin distance $d \ll k$. Hence, a good strategy to find a partition with average distance at most k to $\mathcal{C}$ could be to search in the local neighborhood of the input partitions. Unfortunately, as we show in the following, it is unlikely that a running time of $f(d) \cdot \text{poly}(n, m)$ can be achieved for this local search problem.

We present a parameterized reduction from the W[1]-hard Clique problem [7]. More precisely, we reduce a variant of Clique in which there is at least one vertex in the input graph that is adjacent to all other vertices.

Clique with Universal Vertex
Input: An undirected graph $G = (V, E)$ with a vertex $u \in V$ such that $N[u] = V$, and a nonnegative integer k.
Question: Is there a clique of size k in G?

The W[1]-hardness of Clique with Universal Vertex with respect to k follows from a straightforward reduction from Clique. For notational simplicity, we assume that k is an odd number in the following; the problem clearly remains W[1]-hard with this further restriction.

Given an instance $(G = (V, E), k)$ of Clique with Universal Vertex, we construct an instance of CCML as follows. The base set S consists of the vertex set V and of $(|V| \cdot (k-1)/2) - 1$ further elements. More precisely, for each vertex $v \in V \setminus \{u\}$, we create an element set $S_v := \{v_1, \ldots, v_{(k-1)/2}\}$, and for the universal vertex u, we create an element set $S_u := \{u_1, \ldots, u_{(k-1)/2-1}\}$. The complete element set is then $S := V \cup S_u \cup \bigcup_{v \in V \setminus \{u\}} S_v$.

We construct a CCML instance in which $\mathcal{C}$ consists of $n := 2|E| + 1$ partitions of S. The first $|E|$ partitions consist of one cluster that completely contains S:

$$C_i := \{S\}, 1 \leq i \leq |E|.$$

For each $\{v, w\} \in E$ we create one partition containing $\{v, w\}$ as one cluster and a singleton cluster for each of the other elements. Let $E = \{e_1, \ldots, e_{|E|}\}$. Then these partitions are formally defined as

$$C_{|E|+i} := \{e_i\} \cup \{\{s\} \mid s \in S \setminus e_i\}, 1 \leq i \leq |E|.$$

Finally, we create one partition that, for each $v \in V$, contains a cluster that contains v and S_v:

$$C_{2|E|+1} := \{\{v\} \cup S_v \mid v \in V\}.$$

Overall, the following can be observed for this set of partitions. Two vertices that are adjacent in G are co-clustered in $|E|+1$ partitions of the CCML instance. Two vertices that are not adjacent in G are co-clustered in $|E|$ partitions. Furthermore, each pair of elements $v \in V$ and $w \in S \setminus V$, is co-clustered in $|E|+1$ partitions if $w \in S_v$, and co-clustered in $|E|$ partitions, otherwise. Finally, each pair of elements $v, w \in S \setminus V$ is co-clustered in $|E|+1$ partitions if there is an $x \in V$ such that $v \in S_x$ and $w \in S_x$ and co-clustered in $|E|$ partitions, otherwise. Consequently, for each pair of elements $v, w \in S$ we have $|\operatorname{co}(v,w) - \operatorname{anti}(v,w)| = 1$.

The partition C of the instance is defined exactly as the partition $C_{2|E|+1}$, that is, for each $v \in V$, C contains the cluster $\{v\} \cup S_v$. We conclude the construction of the CCML instance by setting $d := k \cdot (k-1) - 1$.

The main idea behind the construction is that with the S_v's and by setting $d := k \cdot (k-1) - 1$ we can enforce that in a "better" partition within distance d there is a cluster with exactly k elements from V. The elements of this cluster must then induce a clique in G since otherwise the partition is not better than C.

Theorem 4. *CCML parameterized by the radius d of the Mirkin-distance neighborhood is W[1]-hard.*

In the construction above we have $|\operatorname{co}(v,w) - \operatorname{anti}(v,w)| = 1$ for each element pair $v, w \in S$. Consequently, each element pair causes a Mirkin distance of at least $|E|$ and at most $|E|+1$ in any solution. Hence, the CONSENSUS CLUSTERING instance can also be formulated as an "equivalent" instance of CLUSTER EDITING. We can modify the construction above to also show the hardness of a local search variant for CLUSTER EDITING.

CLUSTER EDITING WITH EDGE-MODIFICATION-LOCAL SEARCH
Input: An undirected graph $G = (V, E)$, a cluster graph $C = (V, E')$, and a nonnegative integer k.
Question: Is there a cluster graph $C' = (V, E'')$ such that $\operatorname{dist}(G, C') < \operatorname{dist}(G, C)$ and $\operatorname{dist}(C, C') \leq k$?

Herein, a cluster graph is a disjoint union of complete graphs and $\operatorname{dist}(G = (V, E), H = (V, E')) := |(E \setminus E') \cup (E' \setminus E)|$ denotes the number of edge modifications needed to transform a graph G into a graph H. In complete analogy to Theorem 4, we can show the following.

Theorem 5. CLUSTER EDITING WITH EDGE-MODIFICATION-LOCAL SEARCH *parameterized by the radius k of the edge-modification neighborhood is W[1]-hard.*

5 Conclusion

There are many possibilities for further research concerning CONSENSUS CLUSTERING. For instance, comparing our algorithm with known heuristics for CONSENSUS CLUSTERING would be interesting. Also, further parameters should be considered for Consensus Clustering. For example, what is the complexity of Consensus Clustering when for each input partition, every cluster has a bounded number of elements? Also, the previously known partial kernelization implies fixed-parameter tractability [3] for the parameter "number of dirty element pairs". Are there efficient fixed-parameter algorithms for this parameter? Finally, it would be interesting to consider further parameters for the local search variant of Consensus Clustering. For example, is this problem fixed-parameter tractable when the number n of input partitions is bounded?

References

1. N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Mach. Learn.*, 56(1):89–113, 2004.
2. M. Bertolacci and A. Wirth. Are approximation algorithms for consensus clustering worthwhile? In *Proc. 7th SDM*, pages 437–442. SIAM, 2007.
3. N. Betzler, J. Guo, C. Komusiewicz, and R. Niedermeier. Average parameterization and partial kernelization for computing medians. *J. Comput. Syst. Sci.*, 77(4):774–789, 2011.
4. P. Bonizzoni, G. D. Vedova, and R. Dondi. A PTAS for the minimum consensus clustering problem with a fixed number of clusters. In *Proc. 11th ICTCS*, 2009.
5. P. Bonizzoni, G. D. Vedova, R. Dondi, and T. Jiang. On the approximation of correlation clustering and consensus clustering. *J. Comput. Syst. Sci.*, 74(5):671–696, 2008.
6. T. Coleman and A. Wirth. A polynomial time approximation scheme for k-consensus clustering. In *Proc. 21st SODA*, pages 729–740. SIAM, 2010.
7. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.
8. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.
9. A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1), 2007.
10. A. Goder and V. Filkov. Consensus clustering algorithms: Comparison and refinement. In *Proc. 10th ALENEX*, pages 109–117. SIAM, 2008.
11. M. Karpinski and W. Schudy. Linear time approximation schemes for the Gale-Berlekamp game and related minimization problems. In *Proc. 41st STOC*, pages 313–322. ACM, 2009.
12. M. Křivánek and J. Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Inform.*, 23(3):311–323, 1986.
13. S. Monti, P. Tamayo, J. P. Mesirov, and T. R. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Mach. Learn.*, 52(1–2):91–118, 2003.
14. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.
15. R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Appl. Math.*, 144(1-2):173–182, 2004.
16. Y. Wakabayashi. The complexity of computing medians of relations. *Resenhas*, 3(3):323–350, 1998.