On the Parameterized Complexity of Consensus Clustering[☆,☆☆]

Martin Dörnfelder[a], Jiong Guo[a], Christian Komusiewicz[b], Mathias Weller[b]

[a] *Universität des Saarlandes, Campus E 1.7, D-66123 Saarbrücken, Germany.*
[b] *Institut für Softwaretechnik und Theoretische Informatik, Technische Universität Berlin, D-10587 Berlin, Germany*

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 nonnegative 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".

Keywords: NP-hard problem, data clustering, search tree algorithm, local search

1. Introduction

The NP-hard Consensus Clustering problem (also known as Cluster Ensemble [37] or Clustering Aggregation [17]) 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

[☆] Supported by the DFG Excellence Cluster on Multimodal Computing and Interaction (MMCI) and DFG project DARE (NI 369/11).
[☆☆] A preliminary version appeared in *Proceedings of the* 22^{nd} *International Symposium on Algorithms and Computation*, pages 624–633, volume 7074 of LNCS, Springer 2011.

Email addresses: mdoernfe@mmci.uni-saarland.de (Martin Dörnfelder), jguo@mmci.uni-saarland.de (Jiong Guo), christian.komusiewicz@tu-berlin.de (Christian Komusiewicz), mathias.weller@tu-berlin.de (Mathias Weller)

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'. 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.

Consensus Clustering
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 a wide array of applications, for example in gene expression data analysis and classification [12, 14, 34], classification of electrocardiographic (ECG) test records [23], clustering categorical data [21], subtopic retrieval [8], detecting behavioral anomalies across multiple data sources [29], improving clustering robustness [14, 23, 37, 39], and preserving privacy [17]. AbedAllah and Shimshoni [2] applied Consensus Clustering to the k-nearest neighbor classifier in machine learning, implementing heuristic data reduction techniques. The NP-hardness of Consensus Clustering was shown by Křivánek and Morávek [27] and Wakabayashi [38]. 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 [6]. 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 $d \geq 2$ [7] but it admits a PTAS for minimizing t [7, 9, 22]. Various heuristics for Consensus Clustering have been experimentally evaluated [4, 8, 18, 28, 30, 37]. Consensus Clustering is closely related to Cluster Editing [36], also known as Correlation Clustering [3].

So far, the study of the parameterized complexity [10, 11, 15, 35] 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 assumption that the overall Mirkin distance of solutions is usually small is not realistic 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. [5] 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 $|S| = m < 9p$ [5].[1] Then, checking all possible partitions of S gives an optimal solution, resulting in a fixed-parameter algorithm for the parameter p. The term "partial" refers to the fact that not the overall instance size is bounded but rather some "part" of the instance, in this case m. Since the Mirkin distance is a metric, the average Mirkin distance of solution partitions $k := t/n$ is at least $p/2$ [5]. Hence, the above also implies fixed-parameter tractability with respect to k. However, there are currently no efficient algorithms for parameter m (a 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 hardness result of Bonizzoni et al. [7] 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 Consensus Clustering under the local-search paradigm, which is one of the most popular approaches for solving NP-hard optimization problems. The basic idea is to improve a given solution by considering solutions in "close proximity" (with respect to some to-be-defined distance measure) to the given solution [1, 33]. The combination of local search and parameterized complexity is relatively new. It has been initially considered for the Traveling Salesman problem by Marx [31], who showed W[1]-hardness for the local search variant using the k-exchange neighborhood (other neighborhoods were examined by Guo et al. [19]). On the positive side, Khuller et al. [24] showed that, the k-exchange neighborhood local search variant of the problem of finding a feedback edge set that is incident to a minimum number of vertices is fixed-parameter tractable with respect to k. Fellows et al. [13] considered local-search variants of graph problems and show that "local search versions of most graph problems are W[1]-hard or W[2]-hard on general graphs." Further parameterized complexity results are known for local search variants of Boolean constraint satisfaction problems [26], Stable Marriage [32], Weighted Feedback Arc Set in Tournaments [16], and List Coloring [20]. In this work, we examine a canonical local search variant of Consensus Clustering, where, in addition to $\mathcal{C}$ and S, a partition C of S is given and the task is to decide whether there is a partition C' such that $\operatorname{dist}(C', \mathcal{C}) < \operatorname{dist}(C, \mathcal{C})$ and $\operatorname{dist}(C', C) \leq d$ for some integer $d \geq 0$. We show this problem to be W[1]-hard with respect to d. Moreover, our reduction can also be used to show W[1]-hardness of a natural

[1] Subsequently, this was improved to a data reduction routine that yields an instance with $m < 16p/3$ [25].

local search variant of Cluster Editing.

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 $\mathrm{co}(a, b)$ for $a, b \in S$ to denote the number of partitions in $\mathcal{C}$ where a and b are co-clustered and use $\mathrm{anti}(a, b)$ to denote the number of partitions where a and b are anti-clustered. Clearly, $n = \mathrm{co}(a, b) + \mathrm{anti}(a, b)$. For a partition C of S and elements $a, b \in S$, the function $\mathrm{dist}_C(a, b)$ 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 $\mathrm{dist}_C(a, b) = \mathrm{anti}(a, b)$; otherwise, $\mathrm{dist}_C(a, b) = \mathrm{co}(a, b)$. Clearly, $\mathrm{dist}(C, \mathcal{C}) = \sum_{\{a,b\} \subseteq S} \mathrm{dist}_C(a, b)$.

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 value of either $\mathrm{co}(a, b)$ or $\mathrm{anti}(a, b)$ can be arbitrarily small compared to 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 if and only if $\mathrm{anti}_v(X,Y) < n/3$, $X \leftrightarrow_v Y$ is true if and only if $\mathrm{co}_v(X,Y) < n/3$, and $X\#_vY$ is true if and only if $\mathrm{co}_v(X,Y) \geq n/3$ and $\mathrm{anti}_v(X,Y) \geq n/3$. If $X\#_vY$ 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 $X \leftrightarrow_v Z$ 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 $X\#_vY$, 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 $X \leftrightarrow_v Z$, 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. To show that our branching rules are sound, we use three relations for element pairs with respect to the input partitions.

Definition 1 ([5, Definition 2])**.** *A pair of elements $a, b \in S$ is called a* dirty element pair*, denoted by $a\#b$, if $\mathrm{co}(a, b) \geq n/3$ and $\mathrm{anti}(a, b) \geq n/3$. Moreover, the predicate (ab) is true if and only if $\mathrm{co}(a, b) > 2n/3$, and the predicate $a \leftrightarrow b$ is true if and only if $\mathrm{anti}(a, b) > 2n/3$.*

In particular, we employ the following lemma of Betzler et al. [5].

Lemma 1 ([5, Lemma 5])**.** *For $a, b \in S$, it holds that*

1. $(ab) \wedge (bc) \Rightarrow (ac) \vee a\#c$
2. $(ab) \wedge b \leftrightarrow c \Rightarrow a \leftrightarrow c \vee a\#c$.

Hence, in the first statement we infer that $\mathrm{co}(a, c) \geq n/3$ and in the second statement we infer that $\mathrm{anti}(a, c) \geq n/3$. We can now prove the correctness of the branching rules.

Lemma 2. *Both branching rules are sound.*

Proof. Consider an arbitrary node v of the search tree with the temporary solution C_v, the separation list L_v, and the bound k_v.

First, consider the dirty pair rule. Let C be a partition satisfying the restrictions of v. Obviously, for any two subsets $X, Y \in C_v$, either X and Y are contained in the same subset or they are in two separate subsets in C. The dirty pair rule creates a case for each of these two possibilities. Thus, C fulfills the conditions (C1) and (C2) of exactly one of these two cases. First, assume that C fulfills the conditions of Case v_1. Since X and Y form a dirty subset pair, the pairs formed by one element from X and one from Y are unresolved, that is, they are contained in U_v^2. However, in Case v_1, they become resolved and, therefore, we have to decrease the distance bound accordingly. Corresponding to the co-clustering or anti-clustering status of these pairs in a partition C, we have $\mathrm{dist}_C(a, b) = \mathrm{anti}(a, b)$ or $\mathrm{dist}_C(a, b) = \mathrm{co}(a, b)$ for all $a \in X$ and $b \in Y$. Therefore, C satisfies (C3) at v_1 as well. A similar argument can be used to show that C satisfies (C3) if it fulfills the conditions of Case v_2. Altogether, this proves the soundness of the dirty pair rule.

Next, consider the dirty triple rule. Let $X, Y, Z \in C_v$ with $(XY)_v$, $(YZ)_v$, and $X \leftrightarrow_v Z$ be the three subsets that this rule is applied to. First, we show that L_v contains the pair $\{X, Z\}$. For all elements $x \in X$ and $y \in Y$ we have (xy) since, otherwise, $\text{anti}_v(X, Y) = \sum_{a \in X} \sum_{b \in Y} \text{anti}(a, b) \geq \text{anti}(x, y) \geq n/3$, contradicting $(XY)_v$. Analogously, for all elements $y \in Y$ and $z \in Z$ we have (yz). By Lemma 1, this implies $(xz) \vee x\#z$ for all $x \in X, z \in Z$ and thus $\text{co}(x, z) \geq n/3$ for all $x \in X, z \in Z$. Therefore, if $\{X, Z\} \notin L_v$, then $\text{co}_v(X, Z) = \sum_{x \in X} \sum_{z \in Z} \text{co}(x, z) \geq n/3$, contradicting $X \leftrightarrow_v Z$. By Condition (C2), L_v thus forces X and Z to be anti-clustered in every partition C satisfying the restrictions of node v. Consequently, X and Y or Y and Z must be anti-clustered in C. The dirty triple rule creates a case for each of these two possibilities. With the same argument as for the dirty pair rule, the decrement of the average Mirkin distance bound is correct. Altogether, this shows the soundness of the dirty triple rule.

Finally, we prove that the stop criterion $k_{v_i} < 0$ for the child nodes v_i with $i \in \{1, 2\}$ of v is correct. Since there is at least one dirty subset pair or one dirty subset triple in C_v, there exist some unresolved pairs in U_v^2. Since the branching from v to v_i resolves the pairs from the dirty pair or triple, we have to decrease k_v. Thus $k_{v_i} < 0$ implies that every partition that satisfies Conditions (C1) and (C2) of v_i violates Condition (C3) of v, that is, the operation performed to obtain v_i has cost more than k_v. Consequently, v_i can be ignored. □

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 3. *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 $\text{dist}(C, \mathcal{C}) = \min_{C' \in \mathcal{D}} \text{dist}(C', \mathcal{C})$ 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 $Y \subseteq Z'$.

Proof. Since v is a leaf of the search tree, there are no dirty subset pairs. This implies that for each pair of subsets either $(XY)_v$ or $X \leftrightarrow_v Y$ holds. Furthermore, there are no dirty subset triples, and thus the following holds for all subsets $X, Y, Z \in C_v$,

$$(XY)_v \wedge (YZ)_v \Rightarrow (XZ)_v.$$

Consequently, the predicate $(XY)_v$ for $X, Y \in C_v$ corresponds to an equivalence relation over the subsets in C_v. Hence, by merging all subsets of C_v for which $(XY)_v$ holds we obtain a partition C of S that fulfills Conditions (a) and (b) of the lemma (the subsets in C are precisely the equivalence classes of the

aforementioned equivalence relation). Clearly, C satisfies the Conditions (C1) and (C2) of v.

It therefore remains to show that of all partitions in $\mathcal{D}$, C has minimum distance to $\mathcal{C}$. This can be seen as follows. For each pair of subsets $X, Y \in C_v$ either $(XY)_v$ or $X \leftrightarrow_v Y$ holds. Every subset pair for which $(XY)_v$ holds is co-clustered in C; since $\text{anti}_v(X,Y) < n/3 < co_v(X,Y)$ this implies that, restricted to these subset pairs, C has minimum distance to $\mathcal{C}$ (of all partitions in $\mathcal{D}$). Likewise, every subset pair for which $X \leftrightarrow_v Y$ holds is anti-clustered in C; since $\text{co}(X,Y)_v < n/3 < \text{anti}(X,Y)_v$ this implies that, restricted to these subset pairs, C has minimum distance to $\mathcal{C}$ (of all partitions in $\mathcal{D}$). Summarizing, this shows that C has minimum distance of all partitions in $\mathcal{D}$. □

Altogether, this implies the following.

Proposition 1. *The algorithm is correct.*

Proof. Clearly, the restrictions of the root r can be satisfied by all partitions with an average Mirkin distance at most k to $\mathcal{C}$. In each node of the search tree we either apply one of the branching rules or we enter the second phase of the algorithm. Lemma 2 implies that the proposed branching rules are sound. For each leaf v of the search tree, we obtain a partition C that satisfies Conditions (C1) and(C2) of the restrictions of v. By Lemma 3, of all partitions that satisfy the restrictions of v, C has the minimum distance to $\mathcal{C}$. Hence, if the operations performed at the leaf v lead to $k_v < 0$, then there is no partition that satisfies the restrictions of v. Otherwise, the algorithm finds this partition and outputs it. □

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 $1/n \cdot \text{anti}_v(X,Y)$ or $1/n \cdot \text{co}_v(X,Y)$. 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 $1/n \cdot \text{co}_v(X,Y)$ and $1/n \cdot \text{co}_v(Y,Z)$, 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 $8^k \cdot \text{poly}(n,m)$. 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 first describe the main structure of the algorithm and bound the polynomial part of the running time. Let $\mathcal{C} = (C_1, \ldots, C_n)$ be a multi-set of partitions of a base set $S = \{1, 2, \ldots, m\}$ and let $k \geq 0$ be a rational number. In $O(nm^2)$ time we compute the values of $\mathrm{co}(a, b)$ and $\mathrm{anti}(a, b)$ for each pair of elements $a, b \in S$ and apply the partial kernelization which produces an input instance with $|S| = m \leq 18k$ [5]. On the resulting instance, we then apply the search tree algorithm, which, by Proposition 1, then correctly solves this instance. In each node in our search tree we have to check whether a dirty subset pair or a dirty subset triple exists. Dirty pairs can be found in $O(m^2) = O(k^2)$ time by checking all subset pairs. Analogously, dirty triples can be found in $O(m^3) = O(k^3)$ time. Updating all $\mathrm{co}_v(X, Y)$, $\mathrm{anti}_v(X, Y)$, and k_v after applying a branching rule clearly takes $O(m^2) = O(k^2)$ time. The second phase merges some subsets, which can be done in $O(m^2) = O(k^2)$ time. Summarizing, we need $O(k^3)$ time for each node of the search tree. To complete the proof, it remains to estimate the size of the search tree.

Next, 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:

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

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. [7] 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 $d \geq 2$. In the following, 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.*

Proof. We use CC2 to denote the special case of Consensus Clustering where all input partitions have at most two subsets. In the following, we call a partition that has at most i subsets an *i-partition*. We show the NP-hardness of CC2 by reducing from the NP-hard Cluster Editing problem [3, 36] where the input is an undirected graph $G = (V, E)$ and an integer $k \geq 0$, and the question is whether G can be transformed into a *cluster graph*, that is, a disjoint set of cliques, by modifying (that is, deleting or inserting) at most k edges. Let $n := |V|$ and $m := |E|$. We assume $n - 2 = 2^l$ for an integer l; otherwise, we can add some isolated vertices to G without affecting the solvability of (G, k).

The CC2-instance has the base set $S := V$ and consists of 2-partitions. For each undirected pair of vertices $u, v \in V$, we create a set $\mathcal{C}_{uv}$ of 2^{l+1} 2-partitions, called *uv-partitions*. These partitions have the following properties.

P1 If u and v are adjacent in G, then they are co-clustered in all partitions in $\mathcal{C}_{uv}$. If u and v are nonadjacent in G then they are anti-clustered in all partitions in $\mathcal{C}_{uv}$.

P2 Each pair of elements $x, y \in V$ with $\{x, y\} \neq \{u, v\}$, is co-clustered in exactly half of the partitions in $\mathcal{C}_{uv}$ and anti-clustered in the other half.

We achieve both properties by creating the following two partitions for every $i = 1, \ldots, 2^l$:

$$\begin{array}{llll} S_i^1 = (\{u, v\} \cup A_i, B_i), & S_i^2 = (\{u, v\} \cup B_i, A_i) & (u \text{ and } v \text{ adjacent}), \text{ or} \\ S_i^1 = (\{u\} \cup A_i, \{v\} \cup B_i), & S_i^2 = (\{u\} \cup B_i, \{v\} \cup A_i) & (u \text{ and } v \text{ non-adjacent}). \end{array}$$

Herein A_i and B_i form a 2-partition of $V \setminus \{u, v\}$. Note that A_i or B_i can be empty. The set pairs (A_i, B_i) are constructed in such a way that for every two elements $x, y \in V \setminus \{u, v\}$, there are exactly 2^{l-1} pairs with $\{x, y\} \subseteq A_i$ or $\{x, y\} \subseteq B_i$ and exactly 2^{l-1} pairs with $x \in A_i$ and $y \in B_i$; we call this the *neutrality property*. Such a construction is possible as we show by induction on l. For $l = 1$, there are two elements, say a and b in $V \setminus \{u, v\}$. The two 2-partitions $\{\{a, b\}, \emptyset\}$ and $\{\{a\}, \{b\}\}$ clearly fulfill the neutrality property. For $l > 1$, we can assume that the neutrality property holds for $l-1$. Hence, we can divide $V \setminus \{u, v\}$ in two subsets S_1 and S_2, each of size 2^{l-1}, and for each subset there is a set of 2^{l-1} two-partitions that fulfills the neutrality property. Let $\{(A_1, B_1), \ldots, (A_j, B_j)\}$ and $\{(C_1, D_1), \ldots, (C_j, D_j)\}$ with $j = 2^{l-1}$ be such a set of 2-partitions for S_1 and S_2, respectively. Then we construct the following n 2-partitions for S: For each $1 \leq i \leq j$, we construct two 2-partitions, one is $(A_i \cup C_i, B_i \cup D_i)$ and the other one is $(A_i \cup D_i, B_i \cup C_i)$. Clearly, for each pair $a, b \in S_1$ the neutrality property is fulfilled, since for each 2-partition P of S_1 we construct exactly two 2-partitions in which a and b are clustered in the same way as in P. Obviously, the same holds for pairs with $a, b \in S_2$. For pairs with $a \in S_1$ and $b \in S_2$ the neutrality property can be shown as follows. For each i we create one 2-partition in which a is co-clustered with b and one in which it is anti-clustered with b: Assume without loss of generality that $a \in A_i$ and $b \in C_i$. Then in the partition $(A_i \cup C_i, B_i \cup D_i)$ the two elements are co-clustered and in the partition $(A_i \cup D_i, B_i \cup C_i)$ the two elements are anti-clustered. Summarizing, a set of 2^l 2-partitions of $V \setminus \{u, v\}$ that fulfills the neutrality property exists. Clearly, the construction that is implied by the proof above can be performed in polynomial time.

The set $\mathcal{C}$ of input partitions consists of all $\mathcal{C}_{uv}$. Finally, we set the upper bound t on the total Mirkin distance to

$$k \cdot 2^{l+1} + \frac{n(n-1)}{2} \cdot \left(\frac{n(n-1)}{2} - 1 \right) \cdot 2^l.$$

Next, we show the equivalence between the instances. To this end, observe that, for each pair of elements $u, v \in S$ and each partition C of S, we always have $\mathrm{dist}_C(u, v) = 2^l \cdot (n(n-1)/2 - 1)$, if we exclude the uv-partitions from $\mathcal{C}$.

Suppose the Cluster Editing instance is a yes-instance. Let G' be the resulting cluster graph and let $\mathcal{K}$ denote the set of disjoint cliques. Clearly, $\mathcal{K}$ is a partition of V. We claim that $\mathrm{dist}(\mathcal{K}, \mathcal{C}) \leq t$. We bound $\mathrm{dist}(\mathcal{K}, \mathcal{C}) = \sum_{u,v \in V} \mathrm{dist}_{\mathcal{K}}(u, v)$ by considering all element pairs $u, v \in V$. As argued above, without considering the uv-partitions, we have already $\mathrm{dist}_{\mathcal{K}}(u, v) = 2^l \cdot (n(n-1)/2-1)$ for each element pair. This means that the overall sum of the distances between $\mathcal{K}$ and the corresponding uv-partitions over for all element pairs u, v is at most $k \cdot 2^{l+1}$. If two vertices u and v are in the same clique in G', then they are in the same subset of $\mathcal{K}$. Further, if $\{u, v\} \notin E$, then the distance $\mathrm{dist}_{\mathcal{K}}(u, v)$ restricted to the uv-partitions is 2^{l+1}. This means that every edge addition causes an increase of $\mathrm{dist}(\mathcal{K}, \mathcal{C})$ of exactly 2^{l+1}. By the same reason, each edge deletion also causes an increase of exactly 2^{l+1} of $\mathrm{dist}(\mathcal{K}, \mathcal{C})$. Since G' can be obtained by at most k edge modifications from G, we then have $\mathrm{dist}(\mathcal{K}, \mathcal{C}) = t$. The reversed direction can be shown in a similar way. □

Next, we strengthen the hardness result of Bonizzoni et al. [7] 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 $\mathcal{C} = (C_1, \ldots, C_n)$ of a base set $S = \{1, 2, \ldots, m\}$, 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 $\mathrm{dist}(C, \mathcal{C}) \leq t$?

To show the hardness of CC2P, we reduce a variant of Cluster Editing, where, given G and k, the task is to decide whether G can be transformed into a graph with *at most* two cliques by at most k edge modifications. We use ≤ 2-CE to denote this variant. We thus first show the NP-hardness of ≤ 2-CE, which may be of independent interest.

Theorem 3. *The* ≤ 2-CE *problem is NP-hard.*

Proof. The NP-hard 2-Cluster Editing (2-CE) problem asks whether a given graph G can be transformed into a graph with *exactly* two cliques by at most k edge modifications [36]. In the following, we show the NP-hardness of ≤ 2-CE by reducing 2-CE.

Let $G = (V, E)$ and $k > 0$ be an instance of 2-CE. Then, we reduce this instance to $|V|(|V|-1)/2$ many instances of the ≤ 2-CE problem. Each of these instances corresponds to a distinct pair of vertices $u, v \in V$ and is constructed from G as follows: Add to G two new cliques X and Y, each of size $|V|^2$. Then, add the edges $\{a, x\}$ with $a \in V \setminus \{v\}$ and $x \in X$ and the edges $\{a, y\}$ with $a \in V \setminus \{u\}$ and $y \in Y$. The new graph is denoted by G'. By setting $k' :=$

$|V|^3 - 2|V|^2 + k$, we obtain a $\leq$2-CE instance (G', k') that corresponds to the vertex pair u, v. We now show that the polynomial-time transformation described above is indeed a reduction by proving the following claim. Herein, we assume that $|V| > 6$; all smaller instances of 2-CE can be obviously reduced in constant time to a constant-size equivalent instance of $\leq$2-CE.

> (G, k) is a yes-instance for 2-CE $\Leftrightarrow$ At least one of the generated $\leq$2-CE instances is a yes-instance.

"$\Rightarrow$": Since (G, k) be a yes-instance, there is a solution with two cliques V_1 and V_2 for G. Choose two vertices $u \in V_1$ and $v \in V_2$ and consider the $\leq$2-CE instance (G', k') corresponding to this vertex pair. We count the number edge modifications leading to two cliques $X \cup V_1$ and $Y \cup V_2$. To generate these two cliques, we need exactly $k' = |V|^3 - 2|V|^2 + k$ edge modifications, as we have to edit k edges to transform G into two cliques V_1 and V_2, $|V|^2 \cdot (|V_1| - 1)$ edges $\{a, y\}$ with $a \in V_1$ and $y \in Y$, and $|V|^2 \cdot (|V_2| - 1)$ edges $\{a, x\}$ with $a \in V_2$ and $x \in X$. Thus, (G', k') is a yes-instance for $\leq$ 2-CE.

"$\Leftarrow$": Assume that the set of created $\leq$2-CE instances contains at least one yes-instance (G', k'). We first show that any minimum-cardinality solution M to this instance transforms G' into a cluster graph with two cliques $X \cup V_1$ and $Y \cup V_2$ where $\{V_1, V_2\}$ is a partition of V. First, observe that generating the clique $X \cup Y \cup V$ from G' needs at least $|V|^4 > k'$ edge insertions. Thus, there are exactly two cliques K_1 and K_2 in the graph that results from applying M to G'. We now show that, without loss of generality, $X \subseteq K_1$ and $Y \subseteq K_2$. Suppose that this is not true. Then, $X_1 := X \cap K_1 \neq \emptyset$ and $X_2 := X \cap K_2 \neq \emptyset$. Without loss of generality, assume that $|X_1| \geq |V|^2/2$. Let $Y_1 := Y \cap K_1$ and $Y_2 := Y \cap K_2$. If $|Y_1| > |V|^2/3$, then at least $|V|^4/6$ edge insertions are needed between X_1 and Y_1. This exceeds k' since $|V| > 6$. Hence, $|Y_2| \geq 2|V|^2/3$. Now, this implies $Y \subseteq K_2$. Assume towards a contradiction that this is not the case. We show that moving any vertex $y \in Y_1$ from K_1 to K_2 reduces the number of edge modifications applied to this vertex. First, $|X_1| \geq |X_2|$, so the number of edge insertions between y and X decreases with this move. Second, the number of edge deletions between y and $Y \setminus \{y\}$ decreases by at least $|V|^2/3$ since $|Y_2| - |Y_1| \geq |V|^2/3$. Since at most $|V|$ edge modifications are necessary between y and V, this move thus reduces the overall number of edge modifications applied to y. This contradicts the choice of M and thus $Y \subseteq K_2$. A similar argument can now be applied to show that $X \subseteq K_1$; we omit the details. Summarizing, we can thus assume that there is a solution M that transforms G' into two cliques $X \cup V_1$ and $Y \cup V_2$. By construction, such a solution performs exactly $|V|^3 - 2|V|^2$ edge modifications with at least one endpoint in $X \cup Y$. Hence, G can be transformed into a cluster graph with the two cliques V_1 and V_2 by at most k edge modifications. □

With the hardness of $\leq$2-CE, the hardness of CC2P can be shown by using the same reduction as in the proof of Theorem 2.

Corollary 1. *CC2P is NP-hard.*

By 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. A partition that has average Mirkin distance at most k to $\mathcal{C}$ trivially has Mirkin distance at most k to at least one of the input partitions in $\mathcal{C}$. Moreover, it could be that there is one input partition which has Mirkin distance $d \ll k$ to this partition. 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 [10]. 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 $S := V \cup S_u \cup \bigcup_{v \in V \setminus \{u\}} S_v$.

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 $d := k \cdot (k - 1) - 1$.

Informally, the 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. In the following, we give a formal proof of the correctness of the reduction.

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

Proof. Clearly, the described transformation can be performed in polynomial time and the parameter d is a function of the parameter k. It therefore remains to show the following claim:

> (G, k) is a yes-instance of Clique with Universal Vertex $\Leftrightarrow$
> $(\mathcal{C}, C, d)$ is a yes-instance of CCML.

"$\Rightarrow$": Let K be a k-vertex clique in G. Clearly, we can assume that $u \in K$, since u is adjacent to all vertices in V. We first describe how to construct a partition C' of S from K, and then show that $\operatorname{dist}(C', \mathcal{C}) < \operatorname{dist}(C, \mathcal{C})$ and $\operatorname{dist}(C, C') \leq d$.

One cluster of C' is K. Furthermore, for each vertex $v \in K$ the partition C' contains the cluster S_v. Finally, for each vertex $v \notin K$, C' contains the cluster $S_v \cup \{v\}$. Formally,

$$C' := \{K\} \cup \{S_v \mid v \in K\} \cup \{S_v \cup \{v\} \mid v \notin K\}.$$

First, we show that $\operatorname{dist}(C, C') \leq d$. Let $D := K \cup \bigcup_{v \in K} S_v$. Since for all vertices $v \in V \setminus K$, the sets $S_v \cup \{v\}$ are clusters of both C and C', the distance between C and C' only depends on the elements in D, that is,

$$\operatorname{dist}(C, C') = \sum_{v \in D} \sum_{w \in D} \delta(v, w)/2$$

where $\delta(v, w) = 1$ if v and w are anti-clustered in C and co-clustered in C' or vice versa, and $\delta(v, w) = 0$, otherwise. Two elements $v, w \in D \setminus K$ are either co-clustered in both C and C' or anti-clustered in both partitions. Hence,

$$\operatorname{dist}(C, C') = \sum_{v \in K} \sum_{w \in K \setminus \{v\}} \delta(v, w)/2 + \sum_{v \in K} \sum_{w \in D \setminus K} \delta(v, w).$$

The first term of above equation is clearly $k \cdot (k-1)/2$. The second term is exactly $k \cdot (k-1)/2 - 1$: For each $v \in K \setminus \{u\}$, the number of elements $w \in D \setminus K$ for which $\delta(v, w) = 1$ is $(k-1)/2$, because v is in C in a cluster with the $(k-1)/2$ vertices from S_v. Likewise, for u the number of elements $w \in D \setminus K$ for which $\delta(v, w) = 1$ is $(k-1)/2 - 1$, because u is in C in a cluster with the $(k-1)/2 - 1$ vertices from S_u. Summarizing, $\operatorname{dist}(C, C') = k \cdot (k-1)/2 + k \cdot (k-1)/2 - 1 = d$.

It remains to show that $\operatorname{dist}(C', \mathcal{C}) < \operatorname{dist}(C, \mathcal{C})$. As argued above, we only need to consider elements in $D = K \cup \bigcup_{v \in K} S_v$ since for all other elements the clusters have not changed. Furthermore, the pairs of elements that are clustered differently in C and C' contain at least one element from K, and for those that contain exactly one element $v \in K$ the other element is from S_v. More precisely, we can express $\operatorname{dist}(C, \mathcal{C}) - \operatorname{dist}(C', \mathcal{C})$ as

$$\begin{aligned}
&\sum_{v \in K} \sum_{w \in K \setminus \{v\}} (\operatorname{co}(v, w) - \operatorname{anti}(v, w))/2 + \sum_{v \in K} \sum_{w \in S_v} (\operatorname{anti}(v, w) - \operatorname{co}(v, w)) \\
=&\sum_{v \in K} \sum_{w \in K \setminus \{v\}} 1/2 + \Big(\sum_{v \in K \setminus \{u\}} -(k-1)/2 \Big) - ((k-1)/2 - 1) \\
=&k \cdot (k-1)/2 - (k \cdot (k-1)/2 - 1) \\
>&0.
\end{aligned}$$

The first equality can be seen as follows. First, since K is a clique, all pairs of elements in K are co-clustered in $|E| + 1$ partitions in $\mathcal{C}$. Second, each pair of $v \in K$ and $w \in S_v$ is also co-clustered in $|E|+1$ partitions and for $v \in K \setminus \{u\}$, we have $|S_v| = (k-1)/2$, and $|S_u| = (k-1)/2 - 1$ (recall that we assume that $u \in K$).

Summarizing, the partition C' fulfills both $\text{dist}(C, C') \leq d$ and $\text{dist}(C', \mathcal{C}) < \text{dist}(C, \mathcal{C})$. Hence, $(\mathcal{C}, C, d)$ is a yes-instance of CCML.

"$\Leftarrow$": Let C' be a partition of S with $\text{dist}(C', \mathcal{C}) < \text{dist}(C, \mathcal{C})$, and $\text{dist}(C, C') \leq d$. First, we show that for each $v \in V$ we can assume that C' contains either the cluster $S_v \cup \{v\}$ or the cluster S_v. Using this property, we then show that C' contains a cluster $K^* \subseteq V$ such that $G[K^*]$ is a clique of size k.

We now show that we can assume that, for each $v \in V$, C' contains either the cluster $S_v \cup \{v\}$ or the cluster S_v. Suppose that this is not the case, that is, there is some $v \in V$ such that C' either contains a cluster K with $K \cap S_v \neq \emptyset$ and with $K \setminus (S_v \cup \{v\}) \neq \emptyset$ or it contains two nonempty clusters $K_1 \subseteq S_v$ and $K_2 \subseteq S_v$. We show that in both cases C' can be transformed into a new partition C^* such that $\text{dist}(C^*, C) \leq d$, $\text{dist}(C^*, \mathcal{C}) \leq \text{dist}(C', \mathcal{C})$, and C^* satisfies our assumption.

Case 1: $K \cap S_v \neq \emptyset$ and $K \setminus (S_v \cup \{v\}) \neq \emptyset$. Let C^* denote the partition resulting from C' by replacing K by the two clusters $K \cap S_v$ and $K \setminus S_v$. Note that each pair of elements $x \in K \cap S_v$ and $y \in K \setminus (S_v \cup \{v\})$ is anti-clustered in C. Furthermore, at most one vertex in $K \setminus S_v$, namely v, is co-clustered in C with the vertices from $K \cap S_v$. Therefore, we have $\text{dist}(C^*, C) \leq \text{dist}(C', C)$.

Moreover, for each pair $x \in K \cap S_v$ and $y \in K \setminus (S_v \cup \{v\})$ it also holds that $\text{anti}(x, y) > \text{co}(x, y)$. Also, there is at most one further vertex, namely v, in $K \setminus S_v$. Since $|K \setminus (S_v \cup \{v\})| \geq 1$, we thus have $\text{dist}(C^*, \mathcal{C}) \leq \text{dist}(C', \mathcal{C})$.

By repeatedly applying the modification described above, one can obtain a solution C^* such that for each $v \in V$, every cluster that contains an element from S_v is a subset of $S_v \cup \{v\}$.

Case 2: C' contains two clusters $K_1 \subset S_v$ and $K_2 \subset S_v \cup \{v\}$. Then, we obtain a partition C^* from C' by replacing K_1 and K_2 by the cluster $K_1 \cup K_2$. Both $\text{dist}(C^*, C) < \text{dist}(C', C)$ and $\text{dist}(C^*, \mathcal{C}) < \text{dist}(C', \mathcal{C})$ hold, which can be seen as follows. All pairs of elements $x \in K_1$ and $y \in K_2$ are co-clustered in C; this implies $\text{dist}(C^*, C) < \text{dist}(C', C)$, since for all other elements the clusters do not change. Furthermore, $\text{co}(x, y) > \text{anti}(x, y)$; this implies $\text{dist}(C^*, \mathcal{C}) < \text{dist}(C', \mathcal{C})$. By repeatedly applying this modification, we arrive at a solution for which there is no pair of clusters that are both subsets of $S_v \cup \{v\}$ for some $v \in V$.

Summarizing, we can thus assume that there is a solution C' in which for each $v \in V$, there is a cluster K with either $K = S_v$ or $K = S_v \cup \{v\}$. Thus, all other clusters in C' are subsets of V.

We now show that C' contains one cluster $K \subseteq V$ of size k such that $G[K]$ is a clique. Let $\mathcal{A} := \{A_1, \ldots, A_\ell\}$ denote the clusters in C' that are subsets of V. We can express $\text{dist}(C, \mathcal{C}) - \text{dist}(C', \mathcal{C})$ by summing over the values of $\text{co}(a, b)$ and $\text{anti}(a, b)$ for each pair of elements $a \in A_i$, $1 \leq i \leq \ell$, and $b \in S$, since for

all other element pairs, C and C' have the same partitioning. More precisely,

$$\operatorname{dist}(C,\mathcal{C}) - \operatorname{dist}(C',\mathcal{C}) =$$
$$\sum_{i=1}^{\ell}\Big(\sum_{v\in A_i}\sum_{w\in A_i\setminus\{v\}}(\operatorname{co}(v,w)-\operatorname{anti}(v,w))/2+\sum_{v\in A_i}\sum_{w\in S_v}(\operatorname{anti}(v,w)-\operatorname{co}(v,w))\Big).$$

Since $\operatorname{dist}(C,\mathcal{C}) > \operatorname{dist}(C',\mathcal{C})$, there must be at least one A_i for which the summation above is at least 1, that is, there is an i, $1 \leq i \leq \ell$, such that

$$\sum_{v\in A_i}\sum_{w\in A_i\setminus\{v\}}(\operatorname{co}(v,w)-\operatorname{anti}(v,w))/2 > \sum_{v\in A_i}\sum_{w\in S_v}(\operatorname{co}(v,w)-\operatorname{anti}(v,w)). \quad (1)$$

We show that $G[A_i]$ is a clique of size k in G. First, we show that $|A_i| = k$. Observe that the elements of A_i are all anti-clustered in C and in C each element $v \in A_i$ is co-clustered with all elements of S_v. Hence, $\operatorname{dist}(C,C') \geq |A_i|\cdot(|A_i|-1)/2+|A_i|(k-1)/2-1$. If $|A_i| > k$, then this term is clearly larger than $d = k(k-1)-1$. Hence, $|A_i| \leq k$.

Next, assume towards a contradiction that $|A_i| < k$. Clearly, $\operatorname{co}(v,w) - \operatorname{anti}(v,w) \leq 1$ for each $v,w \in A_i$, and $\operatorname{co}(v,w) - \operatorname{anti}(v,w) = 1$ for $v \in A_i$ and $w \in S_v$. Hence, for all $k > 3$ we have

$$\sum_{v\in A_i}\sum_{w\in A_i\setminus\{v\}}(\operatorname{co}(v,w)-\operatorname{anti}(v,w))/2 < |A_i|(k-1)/2-1$$
$$\leq \sum_{v\in A_i}\sum_{w\in S_v}(\operatorname{co}(v,w)-\operatorname{anti}(v,w)).$$

which contradicts Inequality (1). Consequently, we have $|A_i| = k$. This implies

$$\sum_{v\in A_i}\sum_{w\in S_v}(\operatorname{co}(v,w)-\operatorname{anti}(v,w)) \geq k\cdot(k-1)/2-1$$

and the equality holds only if $u \in A_i$. On the other hand, this implies

$$\sum_{v\in A_i}\sum_{w\in A_i\setminus\{v\}}(\operatorname{co}(v,w)-\operatorname{anti}(v,w))/2 \leq k(k-1)/2.$$

Clearly, equality is obtained only if $\operatorname{co}(v,w) - \operatorname{anti}(v,w) = 1$ for all $v,w \in A_i$.

Therefore, Inequality (1) enforces that $u \in A_i$ and that $\operatorname{co}(v,w)-\operatorname{anti}(v,w) = 1$ for all $v,w \in A_i$. Hence, each pair of elements from A_i is adjacent in G and thus $G[A_i]$ is a size-k clique. □

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. By modifying the construction above, we can then show the hardness of the following local search variant of 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 $\text{dist}(G, C') < \text{dist}(G, C)$ and $\text{dist}(C, C') \leq k$?

Herein, a cluster graph is a disjoint union of complete graphs and $\text{dist}(G = (V, E), H = (V, E')) := |(E \setminus E') \cup (E' \setminus E)|$ denotes the number of edge modifications needed to transform a graph G into a graph H. In complete analogy to Theorem 4, we obtain 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 [5] 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] E. H. L. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. Princeton University Press, 1997.

[2] L. AbedAllah and I. Shimshoni. k Nearest neighbor using ensemble clustering. In *Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery (DaWaK '12)*, volume 7448 of *Lecture Notes in Computer Science*, pages 265–278. Springer, 2012.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1): 89–113, 2004.

[4] M. Bertolacci and A. Wirth. Are approximation algorithms for consensus clustering worthwhile? In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM '07)*, pages 437–442. SIAM, 2007.

[5] N. Betzler, J. Guo, C. Komusiewicz, and R. Niedermeier. Average parameterization and partial kernelization for computing medians. *Journal of Computer and System Sciences*, 77(4):774–789, 2011.

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

[7] P. Bonizzoni, G. Della Vedova, and R. Dondi. A randomized PTAS for the minimum consensus clustering with a fixed number of clusters. *Theoretical Computer Science*, 429:36–45, 2012.

[8] C. Carpineto and G. Romano. Consensus clustering based on a new probabilistic rand index with application to subtopic retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2012.

[9] T. Coleman and A. Wirth. A polynomial time approximation scheme for k-consensus clustering. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '10)*, pages 729–740. SIAM, 2010.

[10] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[11] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[12] R. Fa, A. Nandi, and L.-Y. Gong. Clustering analysis for gene expression data: A methodological review. In *Proceedings of the 5th International Symposium on Communications Control and Signal Processing (ISCCSP '12)*, pages 1–6, 2012.

[13] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, and Y. Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78 (3):707–719, 2012.

[14] V. Filkov and S. Skiena. Integrating microarray data by consensus clustering. *International Journal on Artificial Intelligence Tools*, 13(4):863–880, 2004.

[15] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[16] F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Fast local search algorithm for weighted feedback arc set in tournaments. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI '10)*. AAAI Press, 2010.

[17] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[18] A. Goder and V. Filkov. Consensus clustering algorithms: Comparison and refinement. In *Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX '08)*, pages 109–117. SIAM, 2008.

[19] J. Guo, S. Hartung, R. Niedermeier, and O. Suchý. The parameterized complexity of local search for TSP, more refined. *Algorithmica*, 67(1):89–110, 2013.

[20] S. Hartung and R. Niedermeier. Incremental list coloring of graphs, parameterized by conservation. *Theoretical Computer Science*, 494:86–98, 2013.

[21] N. Iam-On, T. Boongeon, S. Garrett, and C. Price. A link-based cluster ensemble approach for categorical data clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):413–425, 2012.

[22] M. Karpinski and W. Schudy. Linear time approximation schemes for the Gale-Berlekamp game and related minimization problems. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC '09)*, pages 313–322. ACM, 2009.

[23] A. Kelarev, A. Stranieri, J. Yearwood, and H. Jelinek. Empirical investigation of consensus clustering for large ecg data sets. In *Proceedings of the 25th International Symposium on Computer-Based Medical Systems (CBMS '12)*, pages 1–4, 2012.

[24] S. Khuller, R. Bhatia, and R. Pless. On local search and placement of meters in networks. *SIAM Journal on Computing*, 32(2):470–487, 2003.

[25] C. Komusiewicz. *Parameterized Algorithmics for Network Analysis: Clustering & Querying*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2011.

[26] A. Krokhin and D. Marx. On the hardness of losing weight. *ACM Transactions on Algorithms*, 8(2):19:1–19:18, 2012.

[27] M. Křivánek and J. Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986.

[28] A. Lancichinetti and S. Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2, 2012.

[29] A. Liu and D. Lam. Using consensus clustering for multi-view anomaly detection. In *Proceedings on the IEEE Symposium on Security and Privacy Workshops (SPW '12)*, pages 117–124, 2012.

[30] B. Long, Z. M. Zhang, and P. S. Yu. Combining multiple clusterings by soft correspondence. In *Proceedings of the 5th IEEE International Conference on Data Mining*

(ICDM '05), pages 282–289. IEEE Computer Society, 2005.

[31] D. Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Operations Research Letters*, 36(1):31–36, 2008.

[32] D. Marx and I. Schlotter. Stable assignment with couples: Parameterized complexity and local search. *Discrete Optimization*, 8(1):25–40, 2011.

[33] W. Michiels, E. H. L. Aarts, and J. Korst. *Theoretical Aspects of Local Search*. Springer, 2007.

[34] 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. *Machine Learning*, 52(1–2):91–118, 2003.

[35] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[36] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004.

[37] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research*, 3:583–617, 2003.

[38] Y. Wakabayashi. The complexity of computing medians of relations. *Resenhas*, 3(3): 323–350, 1998.

[39] J. Wu. K-means based consensus clustering. In *Advances in K-means Clustering*, Springer Theses, pages 155–175. Springer Berlin Heidelberg, 2012.