Parameterized Dynamic Cluster Editing

Junjie Luo[1]

Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China
School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China
junjie.luo@campus.tu-berlin.de, luojunjie@amss.ac.cn

Hendrik Molter[2]

Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany
h.molter@tu-berlin.de

André Nichterlein

Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany
andre.nichterlein@tu-berlin.de

Rolf Niedermeier

Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany
rolf.niedermeier@tu-berlin.de

Abstract

We introduce a dynamic version of the NP-hard Cluster Editing problem. The essential point here is to take into account dynamically evolving input graphs: Having a cluster graph (that is, a disjoint union of cliques) that represents a solution for a first input graph, can we cost-efficiently transform it into a "similar" cluster graph that is a solution for a second ("subsequent") input graph? This model is motivated by several application scenarios, including incremental clustering, the search for compromise clusterings, or also local search in graph-based data clustering. We thoroughly study six problem variants (edge editing, edge deletion, edge insertion; each combined with two distance measures between cluster graphs). We obtain both fixed-parameter tractability as well as parameterized hardness results, thus (except for two open questions) providing a fairly complete picture of the parameterized computational complexity landscape under the perhaps two most natural parameterizations: the distance of the new "similar" cluster graph to (i) the second input graph and to (ii) the input cluster graph.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases graph-based data clustering, goal-oriented clustering, compromise clustering, NP-hard problems, fixed-parameter tractability, parameterized hardness

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.46

1 Introduction

The NP-hard Cluster Editing problem [6, 31], also known as Correlation Clustering [5], has developed into one of the most popular graph-based data clustering problems in algorithm theory. Given an undirected graph, the task is to transform it into a disjoint union of cliques (also known as cluster graph) by performing a minimum number of edge

[1] Supported by CAS-DAAD Joint Fellowship Program for Doctoral Students of UCAS. Work done while with TU Berlin.
[2] Supported by the DFG, project MATE (NI 369/17).

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 46; pp. 46:1–46:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

modifications (deletions or insertions). Being NP-hard, Cluster Editing gained high popularity in studies concerning parameterized algorithmics, e.g. [1, 4, 8, 9, 12, 18, 20, 22, 25]. To the best of our knowledge, to date these parameterized studies mostly focus on a "static scenario". Chen et al. [12] are an exception by also looking at temporal and multilayer graphs. In their work, the input is a set of graphs (multilayer) or an ordered list of graphs (temporal), in both cases defined over the same vertex set. The goal is to transform each input graph into a cluster graph such that, in the multilayer case, the number of vertices in which any two cluster graphs may differ is bounded, and in the temporal case, the number of vertices in which any consecutive (with respect to their position in the list) cluster graphs may differ is bounded. In this work, we introduce a dynamic view on Cluster Editing by, roughly speaking, assuming that the input graph changes. Thus we seek to efficiently and effectively adapt an existing solution, namely a corresponding cluster graph. In contrast to the work of Chen et al. [12], we do not assume that all future changes are known. We consider the scenario where given an input graph, we only know changes that lie immediately ahead, that is, we know the "new" graph that the input graph changes to. Motivated by the assumption that the "new" cluster graph should only change moderately but still be a valid representation of the data, we parameterize both on the number of edits necessary to obtain the "new" cluster graph and the difference between the "old" and the "new" cluster graph. We finally remark that there have been previous parameterized studies of dynamic (or incremental) graph problems, but they deal with coloring [23], domination [16, 2], or vertex deletion [3, 26] problems.

Mathematical model. In principle, the input for a dynamic version of a static problem X are two instances I and I' of X, a solution S for I, and an integer d. The task is to find a solution S' for I' such that the distance between S and S' is upper-bounded by d. Often, there is an additional constraint on the size of S'. Moreover, the symmetric difference between I and I' is used as a parameter for the problem many times. We arrive at the following "original dynamic version" of Cluster Editing (phrased as decision version).

> Original Dynamic Cluster Editing
> **Input:** Two graphs G_1 and G_2 and a cluster graph G_c over the same vertex set, and integers k and d such that $|E(G_1) \oplus E(G_c)| \leq k$.
> **Question:** Is there a cluster graph G' for G_2 such that $|E(G_2) \oplus E(G')| \leq k$ and $\text{dist}(G', G_c) \leq d$?

Herein, $\oplus$ denotes the symmetric difference between two sets and $\text{dist}(\cdot, \cdot)$ is a generic distance function for cluster graphs, which we discuss later. Moreover, G_c is supposed to be the "solution" given for the input graph G_1. However, since the question in this problem formulation is independent from G_1 we can remove this graph from the input and arrive at the following simplified version of the problem. For the remainder of this paper we focus on this simplified version of Dynamic Cluster Editing.

> Dynamic Cluster Editing
> **Input:** A graph G and a cluster graph G_c over the same vertex set, and two integers: a budget k and a distance upper bound d.
> **Question:** Is there a cluster graph G' for G such that $|E(G) \oplus E(G')| \leq k$ and $\text{dist}(G', G_c) \leq d$?

There are many different distance measures for cluster graphs [28, 29]. Indeed, we will study two standard ways of measuring the distance between two cluster graphs. One is called classification error distance, which measures the number of vertices one needs to move to make two cluster graphs the same – we subsequently refer to it as *matching-based distance*. The other is called disagreement distance, which is the symmetric distance between two edge sets – we subsequently refer to it as *edge-based distance*. Notably, the edge-based distance upper-bounds the matching-based distance. We give formal definitions in Section 2.

Motivation and related work. Beyond parameterized algorithmics and static Cluster Editing, dynamic clustering in general has been subject to many studies, mostly in applied computer science [32, 15, 14, 34, 33, 10]. We mention in passing that there are also close ties to reoptimization (e.g., [7, 30]) and parameterized local search (e.g., [17, 19, 21, 23, 27]).

There are several natural application scenarios that motivate the study of Dynamic Cluster Editing. Next, we list four of them.

Dynamically updating an existing cluster graph. Dynamic Cluster Editing can be interpreted to model a smooth transition between cluster graphs, reflecting that "customers" working with clustered data in a dynamic setting may only tolerate a moderate change of the clustering from "one day to another" since "revolutionary" transformations would require too dramatic changes in their work. In this spirit, when employing small parameter values, Dynamic Cluster Editing has kind of an evolutionary flavor with respect to the history of the various cluster graphs in a dynamic setting.

Editing a graph into a target cluster graph. For a given graph G, there may be many cluster graphs which are at most k edge modifications away. The goal then is to find one of these which is close to the given target cluster graph G_c since in a corresponding application one is already "used to" work with G_c. Alternatively, the editing into the target cluster graph G_c might be too expensive (that is, $|E(G) \oplus E(G')|$ is too big), and one has to find one with small enough modification costs but being still close to the target G_c.

Local search for an improved cluster graph. Here the scenario is that one may have found an initial clustering expressed by G_c, and one searches for another solution G' for G within a certain local region around G_c (captured by our parameter d).

Editing into a compromise clustering. When focusing on the edge-based distance, one may generalize the definition of Dynamic Cluster Editing by allowing G_c to be any graph (not necessarily a cluster graph). This may be used as a model for "compromise cluster editing" in the sense that the goal cluster graph then is a compromise for a cluster graph suitable for both input graphs since it is close to both of them.

Our results. We investigate the (parameterized) computational complexity of Dynamic Cluster Editing. We study Dynamic Cluster Editing as well as two restricted versions where only edge deletions ("Deletion") or edge insertions ("Completion") are allowed. We show that all problem variants (notably also the completion variants, whose static counterpart is trivially polynomial-time solvable) are NP-complete even if the input graph G is already a cluster graph. Table 1 surveys our main parameterized complexity results.

The general versions of Dynamic Cluster Editing all turn out to be parameterized intractable (W[1]-hard) by the single natural parameters "budget k" and "distance d"; however, when both parameters are combined, one achieves a polynomial kernel. We also derive a generic approach towards fixed-parameter tractability for several deletion and completion variants with respect to the budget k as well as with respect to the distance d. Proofs of results marked with ($\star$) are deferred to a full version of the paper.

Table 1 Result overview for DYNAMIC CLUSTER EDITING. We primarily categorize the problem variants by the distance measure (Matching, Edge) they use and secondarily by the allowed modification operation. NP-completeness for all problem variants (last column) even holds if the input graph G is a cluster graph. PK stands for polynomial kernel.

	Problem Variant	$k+d$		k		d			
Matching	Editing	FPT (PK)	Thm. 3	W[1]-h	Thm. 2	W[1]-h	Thm. 2	NP-c	Thm. 1
	Deletion	FPT (PK)		*open*		W[1]-h		NP-c	
	Completion	FPT (PK)		*open*		FPT	Thm. 4	NP-c	
Edge	Editing	FPT (PK)	Thm. 3	W[1]-h	Thm. 2	W[1]-h	Thm. 2	NP-c	Thm. 1
	Deletion	FPT (PK)		FPT	Thm. 4	W[1]-h		NP-c	
	Completion	FPT (PK)		FPT		FPT	Thm. 4	NP-c	

2 Preliminaries and Problems Variants

In this section we give a brief overview on concepts and notation of graph theory and parameterized complexity theory that are used in this paper. We also give formal definitions of the distance measures for cluster graphs we use and of our problem variants.

Graph-theoretic concepts and notations. Given a graph $G = (V, E)$, we say that a vertex set $C \subseteq V$ is a *clique in* G if $G[C]$ is a complete graph. We say that a vertex set $C \subseteq V$ is *isolated* in G if there is no edge $\{u, v\} \in E$ with $u \in C$ and $v \in V \setminus C$. A P_3 is a path with three vertices. We say that vertices $u, v, w \in V$ form an induced P_3 in G if $G[\{u, v, w\}]$ is a P_3. We say that an edge $\{u, v\} \in E$ is part of a P_3 in G if there is a vertex $w \in V$ such that $G[\{u, v, w\}]$ is a P_3. Analogously, we say that a non-edge $\{u, v\} \notin E$ is part of a P_3 in G if there is a vertex $w \in V$ such that $G[\{u, v, w\}]$ is a P_3. A graph $G = (V, E)$ is a *cluster graph* if for all $u, v, w \in V$ we have that $G[\{u, v, w\}]$ is not a P_3, or in other words, P_3 is a forbidden induced subgraph for cluster graphs.

Distance measures for cluster graphs. A cluster graph is simply a disjoint union of cliques. We use two basic distance measures for cluster graphs [28, 29]. The first one is called "matching-based distance" and counts how many vertices have to be moved from one cluster to another to make two cluster graphs the same. It is formally defined as follows.

▶ **Definition 1** (Matching-based distance). Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two cluster graphs defined over the same vertex set. Let $B(G_1, G_2) = (V_1 \uplus V_2, E, w)$ be a weighted complete bipartite graph, where each vertex $u \in V_1$ corresponds to a cluster in G_1, denoted by $C_u \subseteq V$, and each vertex $v \in V_2$ corresponds to a cluster of G_2, denoted by $C_v \subseteq V$. The weight of the edge between $u \in V_1$ and $v \in V_2$ is $w(\{u, v\}) = |C_u \cap C_v|$. Let W be the weight of a maximum-weight matching in $B(G_1, G_2)$. The *matching-based distance* d_M between G_1 and G_2 is $d_M(G_1, G_2) := |V| - W$.

The second distance measure is called "edge-based distance" and simply measures the symmetric distance between the edge sets of two cluster graphs.

▶ **Definition 2** (Edge-based distance). Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two cluster graphs defined over the same vertex set. The *edge-based distance* d_E between G_1 and G_2 is $d_E(G_1, G_2) := |E_1 \oplus E_2|$.

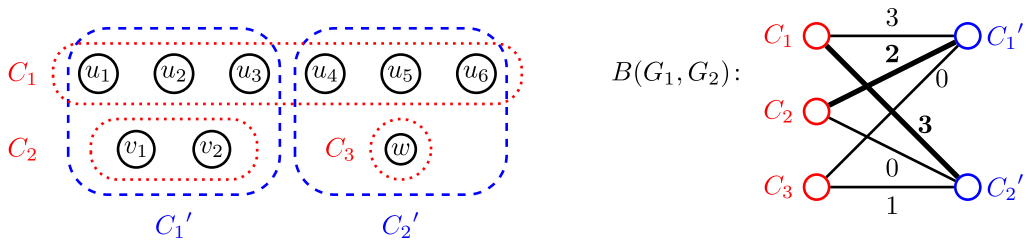

Figure 1 An illustration of the two distance measures. On the left side, red dotted boundaries represent cliques in cluster graph G_1, and blue dashed boundaries represent cliques in cluster graph G_2. The bipartite graph on the right side is the edge-weighted bipartite graph $B(G_1, G_2)$. The maximum-weight matching for $B(G_1, G_2)$ is formed by the two edges represented by the two bold lines.

See Figure 1 for an example illustration of two cluster graphs G_1 and G_2 defined over the same vertex set $V = \{u_1, u_2, u_3, u_4, u_5, u_6, v_1, v_2, w\}$. In G_1 there are three cliques (clusters) $C_1 = \{u_1, u_2, u_3, u_4, u_5, u_6\}$, $C_2 = \{v_1, v_2\}$ and $C_3 = \{w\}$. In G_2 there are two cliques $C_1{}' = \{u_1, u_2, u_3, v_1, v_2\}$ and $C_2{}' = \{u_4, u_5, u_6, w\}$. Then in $B(G_1, G_2)$ we have three vertices on the left side for the cliques in G_1 and two vertices on the right side for the cliques in G_2. A maximum-weight matching for $B(G_1, G_2)$ matches C_1 with C_2' and C_2 with C_1', and has weight $W = 5$. Thus we have $d_M(G_1, G_2) = |V| - W = 9 - 5 = 4$, while $d_E(G_1, G_2) = 3^2 + 2 \cdot 3 + 1 \cdot 3 = 18$.

Problem names and definitions. In the following we present the six problem variants we are considering. We use Dynamic Cluster Editing as a basis for our problem variants. In Dynamic Cluster Deletion we add the restriction that $E(G') \subseteq E(G)$ and in Dynamic Cluster Completion we add the restriction that $E(G) \subseteq E(G')$. For each of these three variants we distinguish a matching-based version and an edge-based version, where the generic "dist" in the problem definition of Dynamic Cluster Editing is replaced by d_M and d_E, respectively. This gives us a total of six problem variants. We use the following abbreviations for our problem names. The letters "DC" stand for "Dynamic Cluster", and "Matching Dist" is short for "Matching-Based Distance". Analogously, "Edge Dist" is short for "Edge-Based Distance". As an example, we abbreviate Dynamic Cluster Editing with Matching-Based Distance as DCEditing (Matching Dist). All other problem variants are abbreviated in an analogous way.

Parameterized complexity. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. We call the second component the *parameter* of the problem. A parameterized problem is *fixed-parameter tractable* (in the complexity class FPT) if there is an algorithm that solves each instance (I, r) in $f(r) \cdot |I|^{O(1)}$ time, for some computable function f. A parameterized problem L admits a *polynomial kernel* if there is a polynomial-time algorithm that transforms each instance (I, r) into an instance (I', r') such that $(I, r) \in L$ if and only if $(I', r') \in L$ and $|(I', r')| \leq f(r)$, for some computable function f. If a parameterized problem is hard for the parameterized complexity class W[1], then it is (presumably) not in FPT. The complexity class W[1] is closed under parameterized reductions, which may run in FPT-time and additionally set the new parameter to a value that exclusively depends on the old parameter.

3 Intractability Results

In this section we first show that all problem variants of Dynamic Cluster Editing are NP-complete even if the input graph G is already a cluster graph. Intuitively, this means that on top of the NP-hard task of transforming a graph into a cluster graph, it is computationally hard to improve an already found clustering (with respect to being closer to the target cluster graph). In particular, while the dynamic versions of Cluster Completion are NP-complete, it is simple to see that classical Cluster Completion is solvable in polynomial time. In a second part we show W[1]-hardness results both for budget parameter k and for distance parameter d for several variants of Dynamic Cluster Editing.

▶ **Theorem 1.** *All considered problem variants of* Dynamic Cluster Editing *are NP-complete, even if the input graph G is a cluster graph.*

Next, we present several parameterized hardness results showing that for certain problem variants we cannot hope for fixed-parameter tractability. Formally, we show the following.

▶ **Theorem 2.** DCEditing (Matching Dist) *and* DCEditing (Edge Dist) *are W[1]-hard with respect to the budget k. The following problems are W[1]-hard with respect to the distance d:* DCEditing (Matching Dist)*,* DCDeletion (Matching Dist)*,* DCEditing (Edge Dist)*, and* DCDeletion (Edge Dist)*.*

As a representative for the results of Theorem 2, we present a parameterized reduction showing that DCEditing (Matching Dist) is W[1]-hard when parameterized by the budget k. The remaining results are deferred to a full version of the paper.

▶ **Lemma 1.** DCEditing (Matching Dist) *is NP-complete and W[1]-hard with respect to the budget k, even if the input graph G is a cluster graph.*

Proof. We present a parameterized reduction from Clique, where given a graph G_0 and an integer ℓ, we are asked to decide whether G_0 contains a complete subgraph of order ℓ. Clique is W[1]-hard when parameterized by ℓ [13]. Given an instance (G_0, ℓ) of Clique, we construct an instance (G, G_c, k, d) of DCEditing (Matching Dist) as follows.

The construction is illustrated in Figure 2. Let $n = |V(G_0)|$. We first construct G. For every vertex v of G_0, create a clique C_v of size $\ell^7 + \ell^4 + \ell^2$. For every edge e of G_0, create a clique C_e of size $\ell^4 + 2$. Lastly, create a big clique C_B of size ℓ^8. Note that G is already a cluster graph. Next we construct G_c. We first create ℓ cliques D_i of size $n\ell^3$ for each $1 \leq i \leq \ell$. Every D_i contains ℓ^3 vertices in every C_v in G. In other words, every C_v in G contains ℓ^3 vertices in every D_i in G_c. Then create a big clique D_B which contains all vertices in C_B and ℓ^7 vertices in every C_v. For every vertex v of G_0, create clique D_v which contains ℓ^2 vertices in C_v and one vertex in every C_e for $v \in e$. Lastly, for every edge e create D_e which contains ℓ^4 vertices in C_e. Set $k = \binom{\ell}{2}(2\ell^4 + 1) + \ell\binom{\ell-1}{2}$ and set $d = d_0 - \ell(\ell - 1)$, where $d_0 = d_M(G, G_c)$ is the matching-based distance between G and G_c, which is computed as follows.

To compute $d_M(G, G_c)$, we need to find an optimal matching in $B(G, G_c)$, the weighted bipartite graph between G and G_c. First, in an optimal matching D_B must be matched with C_B since $|C_B \cap D_B| = \ell^8 > |C_v \cap D_B| = \ell^7$ for any $v \in V(G_0)$ and $C_B \subseteq D_B$. Similarly, D_e must be matched with C_e for every $e \in E(G_0)$. Then the remaining n cliques C_v in G need to be matched to ℓ cliques D_i and n cliques D_v in G_c. Since $|C_v \cap D_i| = \ell^3 > |C_v \cap D_v| = \ell^2$ for any $v \in V(G_0)$ and $1 \leq i \leq \ell$, it is always better to match C_v with some D_i. Since there are only ℓ cliques D_i, we can choose any ℓ cliques from $\{C_v \mid v \in V(G_0)\}$ to be matched

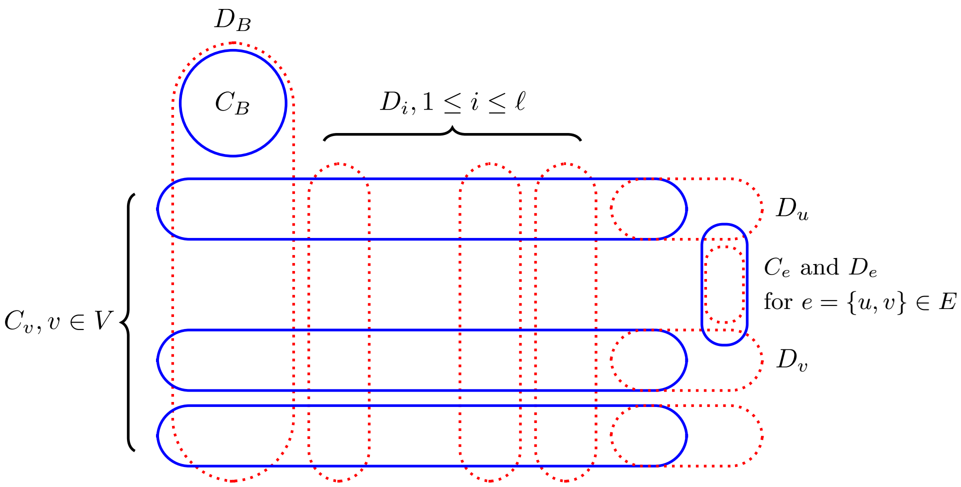

Figure 2 Illustration of the constructed instance for the proof of Lemma 1. Blue solid borders represent cliques in G and red dotted borders represent cliques in G_c. One horizontal long blue border represents a clique C_v in G. It has $\ell + 2$ parts and each part is contained in one clique of G_c. The first part contains ℓ^7 vertices which are contained in the big clique D_B of G_c. The following ℓ parts each contain ℓ^3 vertices which are contained in the ℓ cliques D_i of G_c, and the last part contains ℓ^2 vertices which are contained in D_v of G_c.

with D_i for $1 \leq i \leq \ell$ and the remaining $n - \ell$ cliques to be matched with D_v. Thus we have many different matchings in $B(G, G_c)$ which have the same maximum weight, and each of them corresponds to choosing ℓ different cliques from $\{C_v \mid v \in V(G_0)\}$ to be matched with D_i for $1 \leq i \leq \ell$. For each optimal matching, there are ℓ *free* cliques D_v in G_c which are not matched.

This reduction works in polynomial time. We show that there is a clique of size ℓ in G_0 if and only if there is a cluster graph $G' = (V, E')$ such that $|E(G') \oplus E(G)| \leq k$ and $d_M(G', G_c) \leq d$.

(⇒): Assume that there is a clique C^* of size ℓ in G_0. We modify the graph G as follows. First, for every edge e in the clique C^* partition the corresponding clique C_e in G into three parts; one part contains all vertices in D_e and the other two parts each have one vertex. After this we get $\ell(\ell - 1)$ single vertices. Since C^* is a clique, all these single vertices can be partitioned into ℓ groups such that each group has $\ell - 1$ vertices and all these $\ell - 1$ vertices are contained in the same D_v for some $v \in C^*$. Then for each $v \in C^*$, we combine the corresponding $\ell - 1$ vertices into one clique $C_v^{\ell-1}$. Denote the resulting graph as G'. For an illustration see Figure 3. Along the way to get G', we delete $\binom{\ell}{2}(2\ell^4 + 1)$ edges and add $\ell\binom{\ell-1}{2}$ edges, thus $|E(G) \oplus E(G')| = \binom{\ell}{2}(2\ell^4 + 1) + \ell\binom{\ell-1}{2} = k$. Next we show that $d_M(G', G_c) \leq d_0 - \ell(\ell - 1)$. Recall that an optimal matching in $B(G, G_c)$ can choose ℓ cliques from $\{C_v \mid v \in V(G_0)\}$ to be matched with D_i for $1 \leq i \leq \ell$. Now in $B(G, G_c)$ we can choose all cliques in $\{C_v \mid v \in C^*\}$ to be matched with D_i for $1 \leq i \leq \ell$, and then match $C_v^{\ell-1}$ with D_v for all $v \in C^*$. Then in the new matching we have ℓ additional edges between $C_v^{\ell-1}$ and D_v for $v \in C^*$, each with weight $\ell - 1$. Hence $d_M(G', G_c) \leq d_0 - \ell(\ell - 1)$.

(⇐): Assume that there is a cluster graph $G' = (V, E')$ such that $|E' \oplus E(G)| \leq k$ and $d_M(G', G_c) \leq d$. Note that $k < \ell^7$, thus $k < |C_v|$ and $k < |C_B|$. Consequently, we can only modify edges between vertices in C_e. It is easy to see that in any optimal matching in $B(G', G_c)$, we still have that clique C_B must be matched with D_B and clique C_e must be matched with D_e for every $e \in E(G_0)$. And we should choose ℓ cliques from $\{C_v \mid v \in V(G_0)\}$

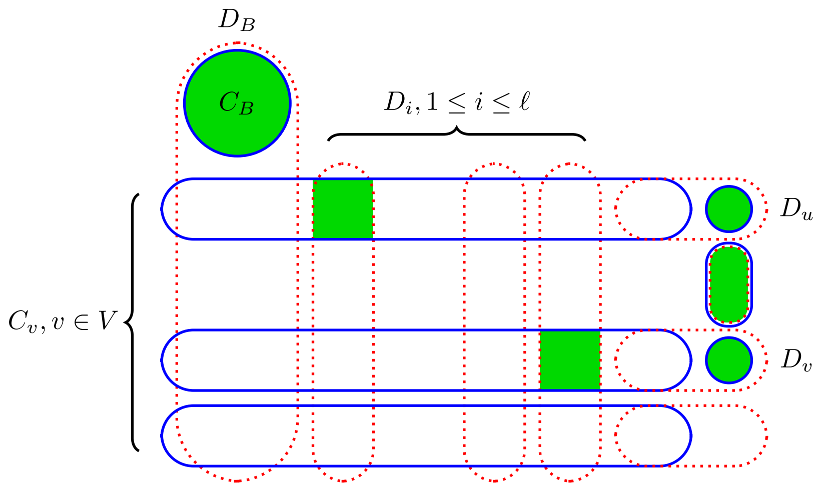

Figure 3 Illustration of a possible solution for the constructed instance (see Figure 2) in the proof of Lemma 1. Blue solid borders represent cliques in G' and red dotted borders represent cliques in G_c. Green shaded areas indicate how cliques of G' and G_c are matched. If two horizontal cliques of G' (blue) are matched with two of the ℓ vertical cliques of G_c, then the corresponding vertices are part of the clique and hence are adjacent. This means the cliques corresponding to the edge can be matched in the indicated way.

to be matched with D_i for $1 \leq i \leq \ell$, which creates ℓ free cliques D_v. Hence, to decrease the distance between G and G_c, or to increase the matching, we have to create new cliques to be matched with these ℓ free cliques D_v. Since for every D_v, except for vertices contained in C_v, it only contains single vertices from C_e with $v \in e$, to create new cliques we need to first separate D_e to get single vertices and then combine them. To decrease the distance by $\ell(\ell-1)$, we need to separate at least $\ell(\ell-1)$ single vertices from C_e. This will cost at least $\ell(\ell-1)(\ell^4+1) - \binom{\ell}{2} = \binom{\ell}{2}(2\ell^4+1)$ edge deletions if we always separate one C_e into three parts and get two single vertices. Then we need to combine these single vertices into at most ℓ cliques since there are at most ℓ free cliques D_v. This will cost at least $\ell\binom{\ell-1}{2}$ edge insertions if all these $\ell(\ell-1)$ single vertices can be partitioned into ℓ groups and each group has $\ell-1$ vertices. Since $k = \binom{\ell}{2}(2\ell^4+1) + \ell\binom{\ell-1}{2}$, we have that in the first step we have to choose $\binom{\ell}{2}$ cliques C_e and separate them into three parts and all these $\ell(\ell-1)$ single vertices are evenly distributed in ℓ free cliques D_v. This means that in G_0 we can select $\binom{\ell}{2}$ edges between ℓ vertices and each vertex has $\ell-1$ incident edges. Thus there is a clique of size ℓ in G_0. ◀

4 Fixed-Parameter Tractability Results

In this section we identify tractable cases for the considered variants of Dynamic Cluster Editing. We first show that all problem variants admit a polynomial kernel for the combination of the budget k and the distance d. Then we present further FPT-results with respect to single parameters.

4.1 Polynomial Kernels for the Combined Parameter $(k + d)$

In this section we present polynomial kernels with respect to the parameter combination $(k+d)$ for all considered variants of Dynamic Cluster Editing:

▶ **Theorem 3.** *The following problems admit an $O(k^2 + d^2)$-vertex kernel:* DCEditing (Matching Dist), DCDeletion (Matching Dist), *and* DCCompletion (Matching Dist). *The following problems admit an $O(k^2 + k \cdot d)$-vertex kernel:* DCEditing (Edge Dist), DCDeletion (Edge Dist), *and* DCCompletion (Edge Dist). *All kernels can be computed in $O(|V|^3)$ time.*

We describe data reduction rules that each take an instance $(G = (V, E), G_c = (V, E_c), k, d)$ as input and output a reduced instance that is a yes-instance if and only if the original instance is a yes-instance (of the corresponding problem variant). In the correctness proof of each reduction rule, we assume that all previous rules are not applicable.

We first use some well-known reduction rules for classical Cluster Editing [20] to get a graph which consists of isolated cliques plus one vertex set of size $k^2 + 2k$ that does not contain any isolated cliques. These rules remove edges that are part of $k + 1$ induced P_3s and add edges between non-adjacent vertex pairs that are part of $k + 1$ induced P_3s. We defer a formal description and correctness proofs of these rules to a full version of the paper. The reason we use these data reduction rules even though there are linear-vertex kernels for classical Cluster Editing [9, 11] is that they do not eliminate any possible solutions.

Now we introduce new reduction rules that are specific to our problem setting, allowing us to use $k + d$ to upper-bound the size of all remaining isolated cliques and their number to get a polynomial kernel. First, we observe that if there is a vertex set that forms an isolated clique both in G and G_c, then we can remove it since it has no influence on k or d in any problem variant. This is formalized in the next rule. We omit a formal correctness proof.

▶ **Reduction Rule 1.** If there is a vertex set $C \subseteq V$ that is an isolated clique in G and G_c, then remove all vertices in C from G and G_c.

The next rules deal with large cliques and allow us to either remove them or conclude that we face a no-instance.

▶ **Reduction Rule 2a** (Matching-based distance)**.** If there is a vertex set $C \subseteq V$ with $|C| > k + 2d + 1$ that is an isolated clique in G, then

- if for each vertex set $C' \subseteq V$ that is an isolated clique in G_c we have that $|C \cap C'| \leq d$, then answer NO,
- otherwise, if there is a vertex set $C' \subseteq V$ that is an isolated clique in G_c and $|C \cap C'| > d$, then we remove vertices in C from G and G_c and decrease d by $|C \setminus C'|$. Furthermore, if $d \geq 0$, then add a set C_d of $k+d+1$ fresh vertices to V. Add all edges between vertices in C_d to E and add all edges between vertices in $C_d \cup (C' \setminus C)$ to G_c (if not already present).

▶ **Reduction Rule 2b** (Edge-based distance)**.** If there is a vertex set $C \subseteq V$ with $|C| > k$ that is an isolated clique in G, then decrease d by $|E_c| + \binom{|C|}{2} - 2|E(G_c[C])| - |E(G_c[V \setminus C])|$ and remove vertices in C from G and G_c.

If none of the previous rules are applicable, then we know that there are no large cliques left in the graph. The next rule allows us to conclude that we face a no-instance if there are too many small cliques left.

▶ **Reduction Rule 3.** If there are more than $2(k + d)$ isolated cliques in G, then output NO.

In the following we show that the rules we presented decrease the number of vertices of the instance to a number polynomial in $k + d$.

▶ **Lemma 2.** *Let $(G = (V, E), G_c = (V, E_c), k, d)$ be an instance of any one of the considered problem variants of* Dynamic Cluster Editing *that uses the matching-based distance. If none of the appropriate reduction rules applies, then $|V| \in O(k^2 + d^2)$.*

▶ **Lemma 3.** *Let $(G = (V, E), G_c = (V, E_c), k, d)$ be an instance of any one of the considered problem variants of* Dynamic Cluster Editing *that uses the edge-based distance. If none of the appropriate reduction rules applies, then $|V| \in O(k^2 + k \cdot d)$.*

Finally, we can apply all data reduction rules exhaustively in $O(|V|^3)$ time.

▶ **Lemma 4.** *Let $(G = (V, E), G_c = (V, E_c), k, d)$ be an instance of any one of the considered problem variants of* Dynamic Cluster Editing*. Then the respective reduction rules can be exhaustively applied in $O(|V|^3)$ time.*

It is easy to see that Theorem 3 directly follows from Lemma 2, Lemma 3, and Lemma 4. We remark that the number of edges that are not part of an isolated clique can be bounded by $O(k^3)$ [20].

4.2 Fixed-Parameter Tractable Cases for Single Parameters

In this section we show that several variants of Dynamic Cluster Editing are fixed-parameter tractable with respect to either the budget k or the distance d.

▶ **Theorem 4.** DCDeletion (Edge Dist) *and* DCCompletion (Edge Dist) *are in FPT with respect to the budget k.* DCCompletion (Matching Dist) *and* DCCompletion (Edge Dist) *are in FPT with respect to the distance d.*

All our FPT results are using the same approach: We reduce (in FPT time) the input to an instance of Multi-Choice Knapsack (MCK), formally defined as follows.

> Multi-Choice Knapsack (MCK)
> **Input:** A family of ℓ mutually disjoint sets $S_1, \ldots, S_\ell$ of items, a weight $w_{i,j}$ and a profit $p_{i,j}$ for each item $j \in S_i$, and two integers W and P.
> **Question:** Is it possible to select one item from each set S_i such that the profit sum is at least P and the weight sum is at most W?

MCK is solvable in pseudo-polynomial time by dynamic programming [24]:

▶ **Lemma 5** ([24, Section 11.5])**.** *MCK can be solved in $O(W \cdot \sum_{i=1}^{\ell} |S_i|)$ time.*

As our approach is easier to explain with the edge-based distance, we start with this case and afterwards show how to extend it to the matching-based distance. As already exploited in our reductions showing NP-hardness (see Theorem 1), all variants of Dynamic Cluster Editing carry some number-problem flavor. Our generic approach will underline this flavor: We will focus on cases where we can partition the vertex set of the input graph into parts such that we will neither add nor delete an edge between two parts. Moreover, we require that the parts are "easy" enough to list all Pareto-optimal (with respect to k and d) solutions in FPT-time (this is usually achieved by some kernelization arguments). However, even with these strict requirements we cannot solve the parts independently from each other: The challenge is that we have to select for each part an appropriate Pareto-optimal solution. Finding a feasible combination of these part-individual solutions leads to a knapsack-type problem (in this case MCK). Indeed, this is common to all studied variants of Dynamic Cluster Editing.

The details for our generic approach (for edge-based distance) are as follows:

1. When necessary, apply data reduction rules from Section 4.1. Partition the input graph $G = (V, E)$ into different parts $G_1, G_2, \ldots, G_{\ell+1}$ such that there exists a solution (if there is a solution) where no edge between two parts will be inserted or deleted. (In particular, this implies that in G there is no edge between the parts.)
2. Compute for each part $G_i = (V_i, E_i)$, $1 \leq i \leq \ell$, a set $S_i \subseteq \mathbb{N}^2$ encoding "cost" and "gain" of all "representative" solutions for G_i. The size of the set S_i has to be upper-bounded in a function of the parameter p. (Here, p will be either k or d.)
   More precisely, select a family $\mathcal{E}_i$ of $f(p)$ edge sets such that for each edge set $E'_i \subseteq \binom{V_i}{2}$ in $\mathcal{E}_i$ the graph $G'_i = (V_i, E'_i \oplus E_i)$ is a cluster graph achievable with the allowed number of modification operations (edge deletions or edge insertions). For each such edge set E'_i, add to S_i a tuple containing the cost $(= |E'_i|)$ and "decrease" of the distance from G_i to the target cluster graph G_c. More formally, add $(|E'_i|, |E'_i \cap E_c| - |E'_i \setminus E_c|)$ to S_i, where E_c is the edge set of G_c. Note that we allow $E'_i = \emptyset$, that is, if G_i is a cluster graph, then S_i contains the tuple $(0, 0)$.
   The set S_i has to fulfill the following property: If there is a solution, then there is a solution G' such that restricting G' to V_i yields a tuple in S_i. More precisely, we require that $(|E(G'[V_i]) \oplus E_i|, |(E(G'[V_i]) \oplus E_i) \cap E_c| - |(E(G'[V_i]) \oplus E_i) \setminus E_c|) \in S_i$.
3. Create an MCK instance I with $W = k$, $P = |E \oplus E_c| - d$, and the sets $S_1, S_2, \ldots, S_\ell$ where the tuples in the sets correspond to the items with the first number in the tuple being its weight and the second number being its profit.
4. Return true if and only if I is a yes-instance.

Note that the requirement in Step 1 implies that a part is a collection of connected components in G. Furthermore, note that the part $G_{\ell+1}$ will be ignored in the subsequent steps. Thus $G_{\ell+1}$ contains all vertices which are not contained in an edge of the edge modification set. Observe that $\ell \leq n$. Hence, we have $\sum_{i=1}^{\ell} |S_i| \in O(f(p) \cdot n)$. (The parameter p will be either k or d.) Moreover, as k and d are smaller than n^2, it follows that $W < n^2$ and thus, by Lemma 5, the MCK instance I created in Step 3 can be solved in $O(f(p) \cdot n^3)$ time in Step 4. This yields the following.

▶ **Observation 1.** *If the partition in Step 1 and the sets S_i in Step 2 can be computed in FPT-time with respect to p, then the above four-step-approach runs in FPT-time with respect to p.*

Note that Steps 1 and 2 are different for every problem variant we consider. There are, however, some similarities between the variants where only edge insertions are allowed. Note that the requirements of Steps 1 and 2 seem impossible to achieve in FPT-time when allowing edge insertions and deletions. Indeed, as shown in Theorem 2, the corresponding edge-edit variants are W[1]-hard with respect to the studied (single) parameters k and d respectively.

Next, we use the above approach to show that DCDeletion (Edge Dist) is fixed-parameter tractable with respect to k. The fixed-parameter tractability of DCCompletion (Edge Dist) with respect to k and with respect to d is deferred to a full version of the paper.

▶ **Lemma 6.** DCDeletion (Edge Dist) *is FPT with respect to k.*

Proof (Sketch). We first apply the known reduction rules for Cluster Editing (see discussion after Theorem 3). As a result, we end up with a graph where at most $k^2 + 2k$ vertices are contained in an induced P_3; all other vertices form a cluster graph with cliques containing at most k vertices each. We define the parts $G_1, G_2, \ldots, G_\ell, G_{\ell+1}$ of Step 1 as follows: The first part $G_1 = (V_1, E_1)$ contains the graph induced by all vertices contained in a P_3. Each of the cliques in the cluster graph $G[V \setminus V_1]$ forms another part G_i, $2 \leq i \leq \ell$. Finally, set $G_{\ell+1} = (\emptyset, \emptyset)$, that is, we include all vertices in the subsequent steps of our generic approach. Clearly, each part contains less than $2k^2$ vertices. Moreover, observe that there are no edges between the parts.

As to Step 2, we add, for every edge set $E'_i \subseteq E_i$ such that $G'_i = (V_i, E'_i \setminus E_i)$ is a cluster graph, a tuple $(|E'_i|, |E'_i \cap E_c| - |E'_i \setminus E_c|)$ to S_i. As this enumerates all possible solutions for G_i, the requirement in Step 2 is fulfilled. Together with Observation 1 we get the statement of the lemma. ◀

We next discuss how to adjust our generic four-step approach for DCCompletion (Matching Dist). The main difference to the edge-based distance variants is an additional search tree of size $O(d^{d+2})$ in the beginning. Each leaf of the search tree then corresponds to a simplified instance where we have additional knowledge on the matching defining the distance of a solution to G_c. With this additional knowledge, we can apply our generic four-step approach in each leaf, yielding the following.

▶ **Lemma 7.** DCCompletion (Matching Dist) *is FPT with respect to d.*

Proof. We apply our generic four-step approach and thus need to provide the details how to implement Steps 1 and 2.

We can assume that our input graph is a cluster graph. Let $\mathcal{C}$ be the set of all cliques in G and $\mathcal{D} = \{D_1, D_2, \ldots, D_q\}$ the set of all cliques in G_c. Then we classify all cliques in $\mathcal{C}$ into two classes $\mathcal{C}_1$ and $\mathcal{C}_2$, where every clique in $\mathcal{C}_1$ has the property that all its vertices are contained in one clique in $\mathcal{D}$ and every clique in $\mathcal{C}_2$ contains vertices from at least two different cliques in $\mathcal{D}$. Observe that $|\mathcal{C}_2| \leq d$ as otherwise the input is a no-instance. Similarly, every clique in $\mathcal{C}_2$ contains vertices from at most $d+1$ different cliques in $\mathcal{D}$ as otherwise the input is a no-instance.

This allows us to do the following branching step. For each clique in $\mathcal{C}_2$ we try out all "meaningful" possibilities to match it to a clique in $\mathcal{D}$, where "meaningful" means that the cliques in $\mathcal{C}_2$ and $\mathcal{D}$ should share some vertices or we decide to not match the clique of $\mathcal{C}_2$ to any clique in $\mathcal{D}$. For each clique this gives us $d+2$ possibilities and hence we have at most d^{d+2} different cases each of which defines a mapping $M : \mathcal{C}_2 \to \mathcal{D} \cup \{\emptyset\}$ that maps a clique in $\mathcal{C}_2$ to the clique in $\mathcal{D}$ it is matched to.

Given the mapping M from cliques in $\mathcal{C}_2$ to cliques $\mathcal{D}$ or $\emptyset$, we partition G into $q+1$ groups $G_1, G_2, \ldots, G_q, G_{q+1}$ with $G_i = G[V_i]$, where $V_i = \{C \in \mathcal{C}_1 \mid C \subseteq D_i\} \cup \{C \in \mathcal{C}_2 \mid M(C) = D_i\}$ and $V_{q+1} = \{C \in \mathcal{C}_2 \mid M(C) = \emptyset\}$.

If there is a solution with a matching that uses the matches given by M, then there is a solution only combining cliques within every group G_i, $1 \leq i \leq q$, since all cliques in G_i that are not matched by M are completely contained in D_i and hence would not be merged with cliques in G_j for some $i \neq j$. This shows that with $\ell = q$ the requirements of Step 1 of our generic approach are met.

Next we describe Step 2, that is, for every part G_i, we show how to compute a set S_i corresponding to all "representative" solutions. Note that all except at most d cliques from G_i need to be merged into one clique that is then matched with D_i, otherwise the matching distance would be too large. For each clique in G_i that is not completely contained in D_i we already know that it is matched to D_i, hence we need to merge all cliques of this kind to one clique $C_i^\star$. Each clique in G_i that is completely contained in D_i and has size at least $d+1$ also needs to be merged to $C_i^\star$, otherwise the matching distance would be too large. For all cliques of G_i that are completely contained in D_i with size x for some $1 \leq x \leq d$ we merge all but d cliques to $C_i^\star$. This leaves us with one big clique $C_i^\star$ and d^2 cliques of size at most d

each. Now we can brute-force all possibilities to merge some of the remaining cliques to $C_i^\star$. There are less than d^d possibilities to do so and for each possibility we add to S_i a tuple representing the cost and gain of merging the cliques according to the partition. ◀

5 Conclusion

Our work provides a first thorough (parameterized) analysis of Dynamic Cluster Editing, addressing a natural dynamic setting for graph-based data clustering. We deliver both (parameterized) tractability and intractability results. Our positive algorithmic results (fixed-parameter tractability and kernelization) are mainly of classification nature. To get practically useful algorithms, one needs to further improve our running times.

The main difference to static Cluster Editing seems to come from the fact that all six variants of Dynamic Cluster Editing remain NP-hard when the input graph is a cluster graph (see Theorem 1). Moreover, Dynamic Cluster Editing (both matching- and edge-based distance) is W[1]-hard with respect to the budget k (see Theorem 2) whereas Cluster Editing is FPT with respect to k. The obvious approach to solve Dynamic Cluster Editing is to compute (almost) all cluster graphs achievable with at most k edge modifications, then from this set of cluster graphs pick one at distance at most d to the target cluster graph. However, listing these cluster graphs is computationally hard. Indeed, our W[1]-hardness results indicate that we might not do much better than using this simple approach.

We mention in passing that our results can also be used to show fixed-parameter tractability for the case when both input graphs are arbitrary graphs and one wants to find a "compromise" cluster graph being close enough (in terms edge-based distance) to both input graphs. The parameter herein is the symmetric distance of the edge sets.

We conclude with few open questions. First, we left open the parameterized complexity of Dynamic Cluster Editing (deletion variant and completion variant) with matching-based distance when parameterized by the budget k, see Table 1 in Section 1. Moreover, the existence of polynomial-size problem kernels for our fixed-parameter tractable cases in case of single parameters (budget k or distance d) is open.

References

1 Faisal N. Abu-Khzam. On the complexity of multi-parameterized cluster editing. *Journal of Discrete Algorithms*, 45:26–34, 2017.

2 Faisal N. Abu-Khzam, Shaowei Cai, Judith Egan, Peter Shaw, and Kai Wang. Turbo-Charging Dominating Set with an FPT Subroutine: Further Improvements and Experimental Analysis. In *Proceedings of the 14th Annual Conference on Theory and Applications of Models of Computation, TAMC 2017*, volume 10185 of *LNCS*, pages 59–70. Springer, 2017.

3 Faisal N Abu-Khzam, Judith Egan, Michael R Fellows, Frances A Rosamond, and Peter Shaw. On the parameterized complexity of dynamic problems. *Theoretical Computer Science*, 607:426–434, 2015.

4 Faisal N. Abu-Khzam, Judith Egan, Serge Gaspers, Alexis Shaw, and Peter Shaw. Cluster Editing with Vertex Splitting. In *Proceeding of the 5th International Symposium on Combinatorial Optimization, ISCO 2018*, volume 10856 of *LNCS*, pages 1–13. Springer, 2018.

5 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation Clustering. *Machine Learning*, 56:89–113, 2004.

6 Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering Gene Expression Patterns. *Journal of Computational Biology*, 6(3-4):281–297, 1999.

7 Hans-Joachim Böckenhauer, Juraj Hromkovič, Tobias Mömke, and Peter Widmayer. On the Hardness of Reoptimization. In *Proceedings of the 34th Conference on Theory and Practice of Computer Science, SOFSEM 2008*, volume 4910 of *LNCS*, pages 50–65. Springer, 2008.

8 Sebastian Böcker and Jan Baumbach. Cluster Editing. In *Proceedings of the 9th Conference on Computability in Europe, CiE 2013*, volume 7921 of *LNCS*, pages 33–44. Springer, 2013.

9 Yixin Cao and Jianer Chen. Cluster Editing: Kernelization Based on Edge Cuts. *Algorithmica*, 64(1):152–169, 2012.

10 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

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

12 Jiehua Chen, Hendrik Molter, Manuel Sorge, and Ondrej Suchý. Cluster Editing in Multi-Layer and Temporal Graphs. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC '18)*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. To appear.

13 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

14 Tamal K. Dey, Alfred Rossi, and Anastasios Sidiropoulos. Temporal Clustering. In *Proceedings of the 25th Annual European Symposium on Algorithms, ESA 2017*, volume 87 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

15 X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov. Clustering With Multi-Layer Graphs: A Spectral Perspective. *IEEE Transactions on Signal Processing*, 60(11):5820–5831, 2012.

16 Rodney G Downey, Judith Egan, Michael R Fellows, Frances A Rosamond, and Peter Shaw. Dynamic dominating set and turbo-charging greedy heuristics. *Tsinghua Science and Technology*, 19(4):329–337, 2014.

17 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012.

18 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Yngve Villanger. Tight Bounds for Parameterized Complexity of Cluster Editing with a Small Number of Clusters. *Journal of Computer and System Sciences*, 80(7):1430–1447, 2014.

19 Serge Gaspers, Eun Jung Kim, Sebastian Ordyniak, Saket Saurabh, and Stefan Szeider. Don't Be Strict in Local Search! In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 486–492. AAAI Press, 2012.

20 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-Modeled Data Clustering: Exact Algorithms for Clique Generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

21 Jiong Guo, Sepp Hartung, Rolf Niedermeier, and Ondrej Suchý. The Parameterized Complexity of Local Search for TSP, More Refined. *Algorithmica*, 67(1):89–110, 2013.

22 Sepp Hartung and Holger H. Hoos. Programming by Optimisation Meets Parameterised Algorithmics: A Case Study for Cluster Editing. In *Proceedings of the 9th International Conference on Learning and Intelligent Optimization, LION 2015*, volume 8994 of *LNCS*, pages 43–58. Springer, 2015.

23 Sepp Hartung and Rolf Niedermeier. Incremental list coloring of graphs, parameterized by conservation. *Theoretical Computer Science*, 494:86–98, 2013.

24 Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004.

25 Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160(15):2259–2270, 2012.

26 R Krithika, Abhishek Sahu, and Prafullkumar Tale. Dynamic parameterized problems. *Algorithmica*, 80(9):2637–2655, 2018.

27 Dániel Marx and Ildikó Schlotter. Parameterized Complexity and Local Search Approaches for the Stable Marriage Problem with Ties. *Algorithmica*, 58(1):170–187, 2010.

28 Marina Meilă. Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 577–584. ACM, 2005.

29 Marina Meilă. Local equivalences of distances between clusterings—a geometric perspective. *Machine Learning*, 86(3):369–389, 2012.

30 Baruch Schieber, Hadas Shachnai, Gal Tamir, and Tami Tamir. A theory and algorithms for combinatorial reoptimization. *Algorithmica*, 80(2):576–607, 2018.

31 Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004.

32 W. Tang, Z. Lu, and I. S. Dhillon. Clustering with Multiple Graphs. In *Proceedings of the Ninth IEEE International Conference on Data Mining, ICDM 2009*, pages 1016–1021, 2009.

33 C. Tantipathananandh and T. Y. Berger-Wolf. Finding Communities in Dynamic Social Networks. In *Proceedings of the IEEE 11th International Conference on Data Mining, ICDM 2011*, pages 1236–1241, 2011.

34 Chayant Tantipathananandh, T. Y. Berger-Wolf, and David Kempe. A Framework for Community Identification in Dynamic Social Networks. In *Proceedings of the 13th SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2007*, pages 717–726. ACM, 2007.