Fully-Dynamic Hierarchical Graph Clustering Using Cut Trees

Christof Doll, Tanja Hartmann, and Dorothea Wagner

Department of Informatics, Karlsruhe Institute of Technology (KIT)*
christof@doll.de.com, {t.hartmann,dorothea.wagner}@kit.edu

Abstract. Algorithms or target functions for graph clustering rarely admit quality guarantees or optimal results in general. However, a hierarchical clustering algorithm by Flake et al., which is based on minimum *s*-*t*-cuts whose sink sides are of minimum size, yields such a provable guarantee. We introduce a new degree of freedom to this method by allowing arbitrary minimum *s*-*t*-cuts and show that this unrestricted algorithm is complete, i.e., any clustering hierarchy based on minimum *s*-*t*-cuts can be found by choosing the right cuts. This allows for a more comprehensive analysis of a graph's structure. Additionally, we present a dynamic version of the unrestricted approach which employs this new degree of freedom to maintain a hierarchy of clusterings fulfilling this quality guarantee and effectively avoid changing the clusterings.

1 Introduction

Graph clustering has become a central tool for the analysis of networks in general, with applications ranging from the field of social sciences to biology and to the growing field of complex systems. The general aim of *graph clustering* is to identify dense subgraphs (*clusters*) that are sparsely connected in networks, i.e., a good clustering conforms to the paradigm of *intra-cluster density* and *inter-cluster sparsity*. Countless formalizations thereof exist, however, the overwhelming majority of algorithms for graph clustering relies on heuristics and do not allow for any structural guarantee on their outputs [1, 2]. Inspired by the work of Kannan et al. [6], Flake et al. [3] recently presented a hierarchical clustering algorithm that does guarantee a very reasonable bottleneck-property based on an input parameter and returns clusterings at different levels of granularity. Their elegant approach exploits properties of *cut trees*, pioneered by Gomory and Hu [4]. It partially constructs those trees using minimum *s*-*t*-cuts whose sink sides are of minimum size. Due to this restriction the returned clusterings are unique. However, the algorithm possibly misses convenient clusterings in graphs where minimum *s*-*t*-cuts and cut trees are not unique (see Appendix A for a small example).

We show that a restriction to specific cuts is not necessary, i.e., permitting arbitrary minimum *s*-*t*-cuts is a feasible degree of freedom. This makes the method more powerful since construction may actually use the most appropriate cut, depending on the application. We further prove that the unrestricted approach is even complete, i.e., any clustering hierarchy based on minimum *s*-*t*-cuts can be returned by choosing the right

* This work was partially supported by the DFG under grant WA 654/15-2.

cuts. Additionally, we develop the first update algorithm that efficiently and dynamically maintains a whole hierarchy of clusterings, as found by our unrestricted method, for a dynamically changing graph. This algorithm allows arbitrary atomic changes, and employs the new degree of freedom to save costs and keep consecutive clusterings on the same level similar (a notion we call *temporal smoothness*).

We briefly give our notational conventions and two fundamental insights in Sec. 2. Then, in Sec. 3, we revisit the static hierarchical algorithm by Flake et al. [3] and prove correctness and completeness of this approach when using arbitrary minimum cuts. In Sec. 4 we present our new update algorithm and its analysis, concluding in Sec. 5.

2 Preliminaries and Notation.

Throughout this work we consider an undirected, weighted graph $G = (V, E, c)$ with vertex set V, edge set E and a non-negative edge weight function c. We write $c(u, v)$ as a shorthand for $c(\{u, v\})$ with $u \sim v$, i.e., $\{u, v\} \in E$. We reserve the term *node* (or *supernode*) for compound vertices of abstracted graphs, which may contain several basic vertices; however, we identify singleton nodes with the contained vertex without further notice. Dynamic modifications of G will solely concern edges as vertex insertions and deletions are trivial for a disconnected vertex. Thus, a modification of G always involves an edge $\{b, d\}$, yielding $G^{\oplus}$ if $\{b, d\}$ is newly inserted into G, and $G^{\ominus}$ if it is deleted from G. We write $G^{\oplus\ominus}$ as a shorthand for $G^{\oplus}$ *or* $G^{\ominus}$. Decreasing edge weights can be handled by the same method as deletions, the techniques for edge insertions also apply for increasing weights. We further assume G to be connected; otherwise one can work on each connected component independently and the results still apply.

An edge $e_T = \{u, v\}$ of a tree $T(G) = (V, E_T, c_T)$ on V induces a cut in G by decomposing $T(G)$ into two connected components. A weighted tree $T(G)$ is called a *cut tree* [4, 5] if edge weights correspond to cut weights and if for any vertex pair $\{u, v\} \in \binom{V}{2}$ the cheapest edge on the unique path between u and v induces a minimum u-v-cut in G. Neither must this edge be unique, nor $T(G)$. Note that we sometimes identify e_T with the cut it induces in G.

A *contraction* of G by $N \subseteq V$ means replacing the set N in G by a single node, denoted by $[N]$, and leaving this node adjacent to all former adjacencies u of vertices of N, with edge weight equal to the sum of all former edges between N and u.

Our understanding of a *clustering* $\mathcal{C}(G)$ of G is a partition of V into subsets C, which define vertex-induced subgraphs, called *clusters*. In the context of dynamic graphs and edge modifications of $\{b, d\}$ we particularly designate C^b, C^d and $C^{b,d}$ containing b and d, respectively. A *hierarchy of clusterings* is a sequence $\mathcal{C}_1(G) \leq \cdots \leq \mathcal{C}_r(G)$ of clusterings such that $\mathcal{C}_i(G) \leq \mathcal{C}_j(G)$ implies that each cluster in $\mathcal{C}_i(G)$ is a subset of a cluster in $\mathcal{C}_j(G)$. We say $\mathcal{C}_i(G) \leq \mathcal{C}_j(G)$ are *hierarchically nested*.

We start by giving two fundamental insights about cuts in static and dynamic graphs. Lemma 1 results from the basic properties of cut trees and is proven in Appendix B. We will use Observation 2 without further notice.

Lemma 1. *Let $(U, V \setminus U)$ denote a minimum u-v-cut in G, $u \in U$ and $x \in U$. Then there exists a minimum x-v-cut $(X, V \setminus X)$ in G, $x \in X$, such that $X \subseteq U$.*

Observation 2. Suppose edge $\{b,d\}$ changes in G yielding $G^{\oplus\ominus}$. Let θ denote a minimum u-v-cut in $G^{\oplus\ominus}$ and $\hat{\theta}$ a min-u-v-cut in G, both not separating b and d. Then $c^{\oplus\ominus}(\theta) = c(\theta) = c(\hat{\theta}) = c^{\oplus\ominus}(\hat{\theta})$, i.e., $\hat{\theta}$ is a minimum u-v-cut in $G^{\oplus\ominus}$.

3 The Static Hierarchical Clustering Algorithm

Flake et al. [3] propose and evaluate a hierarchical algorithm, which clusters instances in a way that yields a certain guarantee on the quality of the clusters. This quality guarantee is inherited from a basic clustering procedure, which computes one clustering. Applying this procedure iteratively to instances obtained by contracting foregoing clusters yields a clustering hierarchy.

The Basic Clustering Procedure. The quality measure of the basic clustering procedure bases on the *expansion* of a cut $(S,\bar{S})$ due to Kannan et al. [6]:

$$\Psi = \frac{\sum_{u\in S, v\in\bar{S}} c(u,v)}{\min\{|S|,|\bar{S}|\}} \qquad (\textit{expansion} \text{ of cut } (S,\bar{S}))$$

Inspired by a bicriterial approach for good clusterings by Kannan et al. [6], which bases on the related measure *conductance*[1], Flake et al. [3] design a basic clustering procedure that, given parameter α, asserts:[2]

$$\underbrace{\frac{c(C,V\setminus C)}{|V\setminus C|}}_{\text{inter-cluster cuts}} \leq \alpha \leq \underbrace{\frac{c(P,Q)}{\min\{|P|,|Q|\}}}_{\text{intra-cluster cuts}} \quad \forall C\in\mathcal{C}(G) \quad \forall P,Q\neq\emptyset \quad P\dot{\cup}Q = C$$

This quality guarantee is due to special properties of cut trees, which are used by the procedure: Given a graph G and parameter $\alpha > 0$, augment G by inserting an artificial vertex t and connecting t to each vertex in G by an edge of weight α. Then compute a cut tree $T(G_\alpha)$ of the resulting graph G_α. Finally, remove t from $T(G_\alpha)$, which decomposes $T(G_\alpha)$ into connected components, which are returned as clusters in $\mathcal{C}(G)$. In the following we call a clustering that can be computed by this procedure a *cut-clustering*, and we denote by $G_\alpha^{\ominus}$ and $G_\alpha^{\oplus}$ the augmented *and* modified graphs.

Flake et al. further point out that, instead of constructing a whole cut tree, only knowing the edges of $T(G_\alpha)$ incident to t would suffice. According to Lemma 3, which directly follows from a lemma introduced by Gusfield [5], Alg. 1 (SCC), with $S = \emptyset$, returns a cut-clustering by constructing such a partial cut tree, which is in fact a *star* with center t (not to be confused with *Strongly Connected Components*). The parameter S will be used later for the dynamic approach. The number of cuts calculated in SCC depends on the sequence of chosen sinks and the shape of the returned cuts. Already known cuts might be covered by later cuts in line 7, i.e., possibly computed without need.

[1] *conductance* is similar to *expansion* but normalizes cuts by total incident edge weight.

[2] The disjoint union $A\cup B$ with $A\cap B = \emptyset$ is denoted by $A\dot{\cup}B$.

Algorithm 1: SIMPLE CUT-CLUSTERING (SCC as a shorthand)

Input: Graph $G_\alpha = (V_\alpha, E_\alpha, c_\alpha)$, set S

1 $\mathcal{C}(G) \leftarrow S$, $V \leftarrow V_\alpha \setminus (\{t\} \cup \bigcup_{C \in S} C)$
2 **while** $\exists\, u \in V$ **do**
3 $\quad (U, V_\alpha \setminus U) \leftarrow$ min-t-u-cut in G_α, with $u \in U$ `// new degree of freedom`
4 $\quad C^u := U$, $r(C^u) := u$
5 $\quad$ **forall** $C^i \in \mathcal{C}(G)$ **do**
6 $\quad\quad$ **if** $r(C^i) \in C^u$ **then** `//` $C^u =: H$ `covers` C^i
7 $\quad\quad\quad C^u \leftarrow C^u \cup C^i$, $\mathcal{C}(G) \leftarrow \mathcal{C}(G) \setminus \{C^i\}$ `// reshaping by Lem.3`
8 $\quad\quad$ **else** $C^u \leftarrow C^u \setminus C^i$ `// reshaping by Lem.3,` $C^u =: V_\alpha \setminus H$
9 $\quad \mathcal{C}(G) \leftarrow \mathcal{C}(G) \cup \{C^u\}$, $V \leftarrow V \setminus C^u$

Lemma 3 (Gusfield [5], Lemma 1). *Let $(C^i, V_\alpha \setminus C^i)$ be a min-t-$r(C^i)$-cut in G, with $r(C^i) \in C^i$. Let $(H, V_\alpha \setminus H)$ be a min-t-u-cut, with $t, u \in V_\alpha \setminus C^i$ and $r(C^i) \in H$. Then the cut $(C^i \cup H, (V_\alpha \setminus C^i) \cap (V_\alpha \setminus H))$ is also a min-t-u-cut.*

Line 3 in SCC represents the new degree of freedom. Whenever used in a hierarchical context, Flake et al. restricted this to minimum t-u-cuts whose sink sides are of minimum size and called the minimum sink side the *community* of u and u a *representative* of its community. Analogously, we call U a *cut side* with *representative* $r(U)$ if $(U, V_\alpha \setminus U)$ is a minimum t-u-cut in G_α, with $u \in U$. We assume, that the final clustering $\mathcal{C}(G)$ found by SCC stores at least one representative per cluster. In the following we identify t-u-cuts $(U, V_\alpha \setminus U)$ with vertex sets U, $u \in U$ and $t \notin U$.

The Hierarchical Algorithm. Flake et al. developed a hierarchical clustering approach (HSCC), which uses SCC iteratively (see Alg. 2). On each level the returned hierarchy provides a cut-clustering $\mathcal{C}_i(G)$ of G with respect to a particular α_i, i.e., $\mathcal{C}_i(G)$ holds the quality guarantee. We call such a hierarchy a *cut-clustering hierarchy*. Iterating a cut-clustering hierarchy bottom-up the α_i-values decrease, i.e., $\alpha_i > \alpha_j$ for $i < j$. For the proof of correctness of Alg. 2 Flake et al. employed spezial nesting properties of communities. These properties guarantee that communities do not change in line 7 and 8 and that communities in the contracted graph (line 4) correspond to communities in the original graph. Thus, the restricted SCC applied to the contracted graph also returns a valid cut-clustering for G, and the resulting hierarchy is a cut-clustering hierarchy.

Algorithm 2: HIERARCHICAL SCC

Input: $G = (V, E, c)$, $\alpha_1 > \cdots > \alpha_r$

1 $\mathcal{C}_0(G) \leftarrow \{\{v\} \mid v \in V\}$, $r(\{v\}) \leftarrow v$
2 **for** $i = 1, \ldots, r$ **do**
3 $\quad$ **forall** $C \in \mathcal{C}_{i-1}(G)$ **do**
4 $\quad\quad$ contract C in G_{α_i}
5 $\quad\quad$ associate $[C]$ with $r(C)$
6 $\quad \mathcal{C}_i(G) \leftarrow \mathrm{SCC}(G_{\alpha_i}, \emptyset)$

Correctness and Completeness of Unrestricted HSCC In the following we show that HSCC remains correct if we apply SCC with arbitrary minimum t-u-cuts, and that

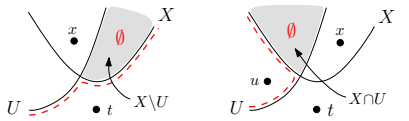
this unrestricted approach is complete. We further characterize the set of cut-clustering hierarchies.

Theorem 4. *Unrestricted HSCC is correct and complete.*

In order to prove the correctness of HSCC independently from special nesting properties of communities, we state the following lemma and show that arbitrary minimum t-u-cuts in the contracted graph (Alg. 2,line 4) are also cut sides in the original graph. Otherwise, SCC applied to the contracted graph would possibly not return a valid cut-clustering for G.

Lemma 5. *Let $(U, V_{\alpha_j} \setminus U)$ denote a min-t-u-cut in G_{α_j} with $u \in U$, and for $\alpha_i > \alpha_j$ let $(X, V_{\alpha_i} \setminus X)$ denote a minimum t-x-cut in G_{α_i} with $x \in X$. Then it holds (a) $X \subseteq U$ if $x \in U$ and (b) $X \cap U = \emptyset$ if $x \notin U$ and $u \notin X$.*

Figure 1 sketches X and U and the conclusions (dashed cuts) proven by contradiction in App. C. Note that for our purpose the case $x \notin U$ but $u \in X$ is irrelevant. Lemma 5 tells us the following: Consider a minimum t-$r(C)$-cut θ in the original graph G_{α_j} with $r(C)$ a representative of a designated node $[C]$ in the contracted graph (line 4, Alg. 2), and let $[C']$ denote an arbitrary node in the contracted graph. If $r(C')$ is in θ then θ also contains $[C']$; in particular, θ contains $[C]$. If $r(C')$ is not in θ then $\theta \cap C' = \emptyset$. Thus, θ is a proper cut in the contracted graph and contains $[C]$. Conversely, each minimum t-$[C]$-cut in the contracted graph is a proper cut in G_{α_j} and contains $r(C)$. Consequently, there exists a 1-1-correspondence between minimum t-$r(C)$-cuts in the original graph and minimum t-$[C]$-cuts in the contracted graph, and SCC applied to the contracted graph returns a valid cut-clustering for G.



(a) It is $X \subseteq U$ if $x \in U$.

(b) $X \cap U = \emptyset$ if $x \notin U$ and $u \notin X$.

Fig. 1. Sketch to proof of Lem. 5

According to the proof of correctness, by choosing the right cuts HSCC is capable to return any cut-clustering hierarchy where the representatives of clusters on one level are a subset of the representatives on the level below. The following lemma shows that this property holds for any cut-clustering hierarchy. Thus, Lemma 5 and Lemma 6 together witness the completeness of HSCC. The proof of Lemma 6 is in App. C.

Lemma 6. *Let $\mathcal{C}_i(G)$ and $\mathcal{C}_j(G)$ denote two cut-clusterings with respect to $\alpha_i > \alpha_j$ and let $C' \in \mathcal{C}_i(G)$ and $C \in \mathcal{C}_j(G)$ denote two clusters with $r(C') \neq r(C)$ but $r(C) \in C'$. Then it holds $C' \subseteq C$ and $r(C')$ is a representative of C in $\mathcal{C}_j(G)$.*

We further give the following simple characterization of all cut-clustering hierarchies and present Corollary 8, which we will apply later to prove temporal smoothness and the feasibility of certain vertex contractions. For a proof of Theorem 7 see App. C.

Theorem 7. *Given a sequence $\alpha_1 > \cdots > \alpha_r$ of parameter values each set of cut-clusterings $\mathcal{C}_1(G), \ldots, \mathcal{C}_r(G)$ forms a hierarchy.*

Corollary 8. *A cluster $C \in \mathcal{C}_j(G)$ separates G into C and $V \setminus C$ such that both parts are clustered independently with respect to $\alpha_i > \alpha_j$, i.e., minimum cuts in G_{α_i} with representatives in C do not cover any vertex in $V \setminus C$ and vice versa.*

Otherwise there would exists a cut-clustering $\mathcal{C}_i(G)$ that is not hierarchically nested in $\mathcal{C}_j(G)$ contradicting Theorem 7.

4 Update Algorithm for Dynamic Clustering Hierarchies

The second part of this work addresses a dynamic version of HSCC. We give a method that employes the new degree of freedom for consecutively updating cut-clustering hierarchies with respect to a given sequence of α's. Based on Theorem 7 this can be already done by simply updating each level independently using a dynamic approach of the basic clustering procedure given by Hartmann et al. [7]. Since the basic non-hierarchical clustering procedure introduced by Flake et al. [3] is not restricted to communities, the basic dynamic approach by Hartmann et al. also allows for the use of arbitrary cuts, and thus, already achieves good temporal smoothness and some cost savings. However, in the following we present a more efficient algorithm, which also exploits the hierarchical structure to save costs and provide high temporal smoothness.

The Basic Clustering Procedure in a Dynamic Scenario. Hartmann et al. [7] developed an algorithm for dynamically updating single cut-clusterings. We will refer to this algorithm by LU (for level update). Given a cut-clustering $\mathcal{C}(G)$, we distinguish four cases of edge modification: inter-cluster deletion (*inter-del*), where the deleted edge is incident to vertices in different clusters, intra-cluster deletion (*intra-del*), i.e., an edge within a cluster is removed, and analogously, inter- and intra-cluster insertion (*inter-ins, intra-ins*). LU reshapes cuts in order to prevent previous clusters from splitting. In this way some clusters are guaranteed to remain clusters or at least subsets of clusters after a change. Regarding different modification cases the following facts hold [7]:

- **a)** all clusters in $\mathcal{C}(G) \setminus \{C^b, C^d\}$ (for *inter-ins*) and in $\{C^b, C^d\}$ (for *inter-del*) are still cut sides in $G_\alpha^{\oplus\ominus}$ with respect to their previous representatives.
- **b)** if $C^{b,d}$ (for *intra-del*) or C^b and C^d (for *inter-ins*) are still cut sides with respect to any representative after the change, $\mathcal{C}(G)$ is still a cut-clustering for $G^{\oplus\ominus}$. We call this the *copy-property* of $\mathcal{C}(G)$. However, the previous representatives of $C^{b,d}, C^b, C^d$ possibly become invalid.
- **c)** for *intra-ins*, $\mathcal{C}(G)$ fulfills the copy-property retaining all representatives.
- **d)** for *inter-del*, LU computes at most $|\mathcal{C}(G)| - 2$ minimum cuts, and updating $\mathcal{C}(G)$ by LU yields $\mathcal{C}(G) = \mathcal{C}(G^{\ominus})$ with valid representatives if $\mathcal{C}(G)$ fulfills the copy-property.
- **e)** for any *deletion*, consider $C \in \mathcal{C}(G)$ with $b, d \notin C$. There exists a minimum t-$r(C)$-cut X in $G_\alpha^{\ominus}$ with $C \subseteq X$.

An Intelligent Hierarchical Approach from Scratch. The naive way to compute a new hierarchy after a change in G is to apply HSCC from scratch. In Sec. 3 we showed that HSCC allows for the use of arbitrary cuts, i.e., construction may use the most appropriate cut, depending on the application. Given an appropriate initial hierarchy we present a hierarchical approach that still calculates a new hierarchy from scratch

but adopts appropriate cuts applied before. To this end we modify HSCC by improving SCC: When computing a new min-t-u-cut θ (u may be a node) let C denote the cluster that contains $r(u)$ in the old clustering on the same level. If $c^{\oplus\ominus}(\theta) = c^{\oplus\ominus}(C)$ in $G_\alpha^{\oplus\ominus}$, SCC takes C as new minimum t-u-cut.

Lemma 9. *In the situation described above it is $u \subseteq C$ and C is a minimum t-u-cut in the contracted graph (Alg. 2, line 4) resulting from $G_\alpha^{\oplus\ominus}$. Thus, the intelligent hierarchical approach is correct.*

For a proof see App. D. In the following we will refer to the improved SCC by *intelligent* SCC (or ISCC). We will further express the costs of our new update algorithm in terms of costs of the intelligent hierarchical approach: Given a hierarchy for G and a new hierarchy from level 1 to level $i-1$ we denote the costs for extending the hierarchy to level j by $\mathcal{T}([i,j], G^{\oplus\ominus})$.

For one level, $\mathcal{T}([i,i], G^{\oplus\ominus})$ consists of the costs for contracting the clusters on level $i-1$ and the costs of ISCC applied to the contracted graph. The latter depend on the size of the contracted graph, which influences the runtime of the cut computations, and the number of calculated cuts.

Reusable Parts of the Hierarchy in a Dynamic Scenario. Given an edge modification a cut-clustering hierarchy decomposes into two parts. Levels where the modification induces an inter-cluster event form the lower part, intra-event levels build the upper part. The first idea in this paragraph considers levels of intra-cluster events. According to Fact c) each intra-ins level can be copied to a new hierarchy. An intra-del level can be copied if $C^{b,d}$ remains a cut side, cf. Fact b). The following lemma gives a further indicator for an intra-del level fulfilling the copy property. We sketch the proof in App. D.

Lemma 10. *Let $\mathcal{C}(G)$ denote an intra-del cut-clustering with $b,d \in C^{b,d}$. If no cut-clustering $\mathcal{C}(G^\ominus)$ exists with b,d in different clusters, $\mathcal{C}(G)$ fulfills the copy-property. If there exists a cut-clustering $\mathcal{C}_i(G^\ominus)$ with $b,d \in C_i^{b,d}$, each cut-clustering $\mathcal{C}_j(G)$ with $\alpha_i > \alpha_j$ fulfills the copy-property.*

According to Lemma 10 and Fact c) we get the following:

Theorem 11. *Given a cut-clustering hierarchy, let k denote the lowest intra-del level that fulfills the copy-property (deletion) or just the lowest intra-ins level (insertion). Then all levels $i \geq k$ can be reused as part of a new hierarchy (however, in case of deletion some repres. possibly become invalid, cf. Fact b)).*

A second idea is to consider subtrees of clusters. A *subtree* of a cluster C on level i consists of C and all clusters on lower levels in the hierarchy that are nested in C. Lemma 12 (proof in App. D) and Theorem 13 attest that in some cases we can preserve the whole subtree of a cluster after a change in G.

Lemma 12. *Let $C \not\ni b,d$ denote a cluster in $\mathcal{C}_j(G)$ that remains a cut side for $r(C)$ (which is equivalent to any representative) in $G_{\alpha_j}^{\oplus\ominus}$. Let further denote $C' \subseteq C$ a cluster in $\mathcal{C}_i(G)$, $i < j$. Then C' remains a cut side for $r(C')$ in $G_{\alpha_i}^{\oplus\ominus}$.*

Algorithm 3: UPDATE INTRA-DEL LEVEL

Input: Graph $D_\alpha^\ominus$, cut-clustering $\mathcal{C}(G) \ni C^{b,d}$

1 **if** $\exists\, C \in \mathcal{C}(G)$ *that is not a proper union of nodes in* V **then** `// V := V(D_α^⊖)`
2 $\quad \mathcal{C}(G^\ominus) \leftarrow \text{ISCC}(D_\alpha^\ominus, \emptyset)$ `// ISCC takes nodes containing...`
3 $\quad$ **return** $(\mathcal{C}(G^\ominus), \text{false})$ `// ...representatives in C(G) first`
4 **while** $\exists\, u \in V$ *with* $u \subseteq C^{b,d}$ **do** `// start with u ∋ r(C^{b,d})`
5 $\quad U \leftarrow$ community of u in $D_\alpha^\ominus$
6 $\quad$ **if** $\exists\, x \in U$ *with* $x \not\subseteq C^{b,d}$ **then** apply line 4 to 9 of Algo 1, goto line 9
7 $\quad$ **if** $c^\ominus(U) = c(C^{b,d})$ **then** $\mathcal{C}(G^\ominus) \leftarrow \mathcal{C}(G)$, $r(C^{b,d}) \leftarrow u$, **return** $(\mathcal{C}(G^\ominus), \text{true})$
8 $\quad$ apply line 4 to 9 of Algo 1 (ISCC)
9 **while** $\exists\, u \in V$ **do** `// ISCC takes nodes containing rep. in C(G) first`
10 $\quad$ apply line 3 to 9 of Algo 1 (ISCC)
11 **return** $(\mathcal{C}(G^\ominus), \text{false})$

If C in Lemma 12 even remains a cluster in a new cut-clustering $\mathcal{C}_j(G^{\oplus\ominus})$, according to Corollary 8 the following holds:

Theorem 13. *In a cut-clustering hierarchy let $C \not\ni b,d$ denote a cluster in $\mathcal{C}_j(G)$ that is also a cluster in a cut-clustering $\mathcal{C}_j(G^{\oplus\ominus})$. Then the whole subtree of C can be used as part of a new hierarchy (representatives remain valid).*

We define the root of a (inclusion-) maximal reusable subtree as a *highest root*.

Our New Update Approach. Our new update approach treats the two parts of inter and intra-event levels of the hierarchy differently. We start by applying Theorem 11 and Theorem 13 to intra-event levels and estimate the costs in terms of costs of the intelligent HSCC.

In case of insertion Theorem 11 tells us that we can just copy each intra-ins level to a new hierarchy without further costs (cf. upper shaded area in Fig. 2).

In case of deletion we search for the lowest intra-del level k that fulfills the copy property. To this end, beginning at the lowest intra-del level ℓ we iteratively apply Alg. 3 until the first copy-property level k is found. Alg. 3 takes an intra-del clustering $\mathcal{C}_i(G)$ and a graph $D_{\alpha_i}^\ominus$ obtained from $G_{\alpha_i}^\ominus$ by contracting clusters on level $i-1$. Line 2 catches a case where $\mathcal{C}_i(G)$ obviously does not fulfill the copy-property and applies ISCC in this case. If $\mathcal{C}_i(G)$ fulfills the copy-property, according to Fact b) it suffices to find a valid representative for $C_i^{b,d}$. Thus, line 4 ff. search for such a representative and return $\mathcal{C}_i(G)$ together with the representative if one is found and continue ISCC otherwise. Lemma 17 in App. D shows that Alg. 3 finds a valid representative of $C_i^{b,d}$ if there is one. The costs for updating level ℓ to $k-1$ are about $\mathcal{T}([\ell, k-1], G^\ominus)$ since Alg. 3 is just a modified SCC (see Fig. 2, area (1)).

After we found level k we can actually copy all levels $i > k$ according to Theorem 11, apart from the representatives of $C_i^{b,d}$, $i = k+1, \ldots, r$. Hence, we apply the while-loop in line 4 of Alg. 3 instead of copying the levels, since this additionally returns valid representatives. This costs about $\sum_{i=k}^{r} \mathcal{T}([i,i], C_i^{b,d})$ also including the costs for level k (see area (2), Fig.2).

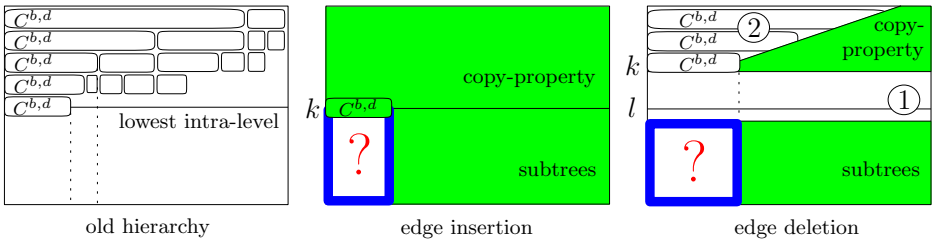


Fig. 2. Sketch of costs for updating a hierarchy using our first update approach. Shaded areas represent saved costs compared to a hierarchical construction from scratch.

However, in order to apply Alg. 3 the first time on level ℓ we need to compute a clustering $\mathcal{C}_{\ell-1}(G^{\ominus})$ on the highest inter-del level acting as a base for contracting the initial instance. To this end, we contract $C^b_{\ell-1}$ and $C^d_{\ell-1}$ in $G^{\ominus}_{\alpha_{\ell-1}}$ and associate the nodes with $r(C^b)$ and $r(C^d)$. Then we apply LU to the obtained graph, which is feasible and costs about $\mathcal{T}([l-1,l-1],G^{\ominus})$; see App. E.

In both cases, insertion and deletion, we can further reuse the subtrees of all clusters $C \in \mathcal{C}_k(G) \setminus \{C^{b,d}_k\}$ by Theorem 13 (see lower shaded area in Fig. 2). This already updates parts of inter-event levels. In case of deletion the clusters of subtrees overlapping levels $\ell-1$ to $k-1$ already exist in $\mathcal{C}_{\ell-1}(G^{\ominus}),\dots,\mathcal{C}_{k-1}(G^{\ominus})$ since Alg. 3 and LU construct reusable subtrees, apart from highest roots, by default according to Corollary 8 (see also Lemma 18 and Lemma 19 in App. D).

Observation 14. Each level $i \geq \ell$ (intra-del) or $i \geq k$ (intra-ins) that fulfills the copy-property and each reusable subtree that is rooted on level $i \geq k$ is part of the new hierarchy (with valid repres.) resulting from our update approach.

By updating the intra-event levels with this approach, we reduce the problem of updating a cut-clustering hierarchy of r levels to an update of $k-1$ levels (insertion) or $\ell-2$ levels (deletion), regarding an instance just as big as $C^{b,d}_k$ (cf. boxed question mark in Fig. 2).

Strategies for Completing the Hierarchy on Inter-Event Levels. After our first update step we still need to fill in the question marks in Fig. 2, i.e., construct a hierarchy based on the vertices in $C^{b,d}_k$. According to Corollary 8, $C^{b,d}_k$ and $V \setminus C^{b,d}_k$ in G are clustered independently on the missing levels. Thus, when updating level i in the following, we consider $G^{\oplus\ominus}_{\alpha_i}$ with $V \setminus C^{b,d}_k$ contracted into a node representing the subtrees already used.

In case of insertion we iterate the missing levels bottom-up contracting $G^{\oplus}_{\alpha_i}$ as the hierarchical approach does. On each level we apply Alg. 4, which is a modified SCC. It takes an inter-ins clustering $\mathcal{C}_i(G)$ and a graph $G^{\oplus}_{\alpha_i}$ contracted as described above. Line 1 further contracts $G^{\oplus}_{\alpha_i}$, which, together with line 2, enables the algorithm to save the costs for explicitly constructing reusable trees, as we will see later. The contraction is as follows: Contract each $C \in \mathcal{C}_i(G) \setminus \{C^b, C^d\}$ that is a proper union of nodes in the current instance $G^{\oplus}_{\alpha_i}$. Associate a new node $[C]$ with $r(X)$, where $X \in \mathcal{C}_{i-1}(G^{\oplus})$

Algorithm 4: UPDATE INTER-INS LEVEL

Input: Graph $G_\alpha^\oplus$, partial clustering $P := \{C \in \mathcal{C}(G) \mid C \subseteq C_k^{b,d}\} \supseteq \{C^b, C^d\}$

1 $D_\alpha^\oplus \leftarrow$contract some $C \in P \setminus \{C^b, C^d\}$ in $G_\alpha^\oplus$ according to text description

2 $S \leftarrow \{C \in P \mid [C]$ in $D_\alpha^\oplus$ formed in line 1$\}$ `//identify` $V := V(G_\alpha^\oplus)$ `with` $C_k^{b,d}$

3 **if** $\exists\, C \in P$ *that is not a proper union of nodes in* V **then**

4 $\quad \mathcal{C}(G^\oplus) \leftarrow$ ISCC($D_\alpha^\oplus$, S), **return** ($\mathcal{C}(G^\oplus)$,false)

5 $\mathcal{C}(G^\oplus) \leftarrow S$, $V \leftarrow V \setminus \bigcup_{C \in S} C$, $b \leftarrow$ false, $d \leftarrow$ false `//` $V = C^b \cup C^d$

6 **while** $\exists\, u \in V$ **do** `//start with` $u_b \ni r(C^b)$ `and` $u_d \ni r(C^d)$

7 $\quad U \leftarrow$ community of u in $D_\alpha^\oplus$

8 $\quad$ **if** $\exists\, x \in U$ *with* $x \not\subseteq C^b$ *or* $x \not\subseteq C^d$ **then** skip line 9 in later iterations

9 $\quad$ **if** $c^\oplus(U) = c(C^b)$ *or* $c^\oplus(U) = c(C^d)$ **then**

10 $\quad\quad b \leftarrow$true, $r(C^b) \leftarrow u$, $U \leftarrow C^b$ (if currently $u \subseteq C^b =: Z$)

11 $\quad\quad d \leftarrow$true, $r(C^d) \leftarrow u$, $U \leftarrow C^d$ (if currently $u \subseteq C^d =: Z$)

12 $\quad\quad$ in later iterations only consider $u \not\subseteq Z$, in line 6

13 $\quad\quad$ **if** b *and* d **then** $\mathcal{C}(G^\oplus) \leftarrow P$, **return** ($\mathcal{C}(G^\oplus)$, true)

14 $\quad$ apply line 4 to 9 of Algo 1 (ISCC)

15 **return** ($\mathcal{C}(G^\oplus)$,false)

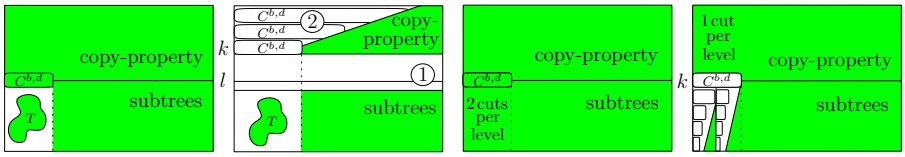
contains $r(C)$. In Lemma 21, found in App. E, we prove that applying ISCC to the obtained graph $D_{\alpha_i}^\oplus$ is correct, i.e., returns a cut-clustering for $G^\oplus$. Line 9 catches a case where $\mathcal{C}_i(G)$ obviously does not fulfill the copy-property and applies ISCC in this case. If $\mathcal{C}_i(G)$ fulfills the copy-property according to Fact b) it suffices to find a valid representative for C_i^b and C_i^d. Thus, line 6 ff. search for those representatives and return the part of $\mathcal{C}_i(G)$ that is nested in $C_k^{b,d}$ together with the representatives if some are found or continue ISCC otherwise. The proof that Alg. 4 finds valid representatives of C_i^b and C_i^d if some exist is analog to Alg. 3. Although Alg. 4 detects each level that fulfills the copy-property, when updating inter-ins levels we can not directly benefit from their copy-property. Thus, applying Alg. 4 to $k-1$ inter-ins levels costs about $\mathcal{T}([1,k-1], C_k^{b,d})$; see App. E.

Furthermore, the bottom-up iteration makes the reuse of subtrees impossible. However, Alg. 4 counterbalances the missing subtree conservation. Using the same techniques as Alg. 3, Alg. 4 returns all reusable subtrees by default, apart from highest roots. It even saves the costs for explicitly constructing such trees, due to lines 1 and 2 as follows: By Corollary 8 each cluster of a reusable subtree is contracted in line 1 and added to S in line 2. Due to Fact a) the nodes in S are considered as cut sides that are already known, and thus, omitted when choosing sinks for cut computations in ISCC. Particularly, Alg. 4 avoids cut computations for clusters in reusable subtrees. Consequently, we deduct the costs T for explicitly constructing reusable subtrees (see Fig. 3(a)).

In case of deletion a bottom-up approach would not allow the reuse of subtrees. Thus, we iterate the old hierarchy top-down updating each level in the same way as level $\ell - 1$ in the first update step, but using a smaller instance due to already known subtrees. As we have seen before, this method detects each reusable subtree, possi-

	arbitrary hierarchy general costs	hierarchy remains valid lowest possible costs
insertion	$\mathcal{T}([1,k-1],C_k^{b,d}) - T$	$2(k-1)$ cpc
deletion	$\sum_{i=k}^{r}\mathcal{T}([i,i],C_i^{b,d}) + \mathcal{T}([l-1,k-1],G^{\ominus})$ $+\mathcal{T}([1,l-2],C_k^{b,d}) - T$	$(r-k+1) + \sum_{i=1}^{k-1} \lvert\mathcal{C}_i^*(G)\rvert$ cpc

Table 1. Sketch of costs, cpc = costs per cut. $\mathcal{C}_i^*(G) := \{C \in \mathcal{C}_i(G) \mid C \subseteq C_{i+1}^*\}$ with $C_{i+1}^* := C_{i+1}^{b,d}$ or $C_{i+1}^* := C_{i+1}^b \cup C_{i+1}^d$.



(a) Costs regarding an arbitrary hierarchy. (b) Lower bounds if hierarchy still valid.

Fig. 3. Sketch of costs for updating a hierarchy applying our new update approach. Shaded areas represent saved costs compared to a hierarchical approach from scratch.

bly apart from highest roots. Thus, we copy those subtrees to the new hierarchy and merge the found roots with the node in $G_{\alpha_i}^{\ominus}$ that represents previously found subtrees in order to save costs. Hence, completing the hierarchy in case of deletion costs about $\mathcal{T}([1,\ell-2],C_k^{b,d}) - T$ (see Fig. 3(a)). Since for inter-cluster deletions LU bases on ISCC, it further respects the copy-property (cf. Fact d)).

Observation 15. Each level $i \leq \ell - 1$ (inter-del) or $i \leq k-1$ (inter-ins) that fulfills the copy-property and each reusable subtree that is rooted on level $i \leq k-1$ is part of the new hierarchy (with valid repres.), possibly apart from highest roots.

Performance. In the following we just sum up the costs and the observations regarding temporal smoothness already given with the description of our new update approach. The latter—which we left unformalized— in parts synergizes with cost saving, an observation foremost reflected in the first update step.

Theorem 16. *Each level fulfilling the copy-property and each reusable subtree (possibly apart from highest roots) is part of the new hierarchy (with valid representatives) build by our update algorithm. In particular, our algorithm returns the previous hierarchy if this is still a cut-clustering hierarchy after the change.*

We sketch the general costs for updating an arbitrary hierarchy in Table 1 and visualize them in Fig. 3(a). Furthermore, we consider the—possibly rather common—case that the old hierarchy is still valid after some graph modification. For this case we list the lowest possible costs in Table 1, which occur if on each inter-ins level Alg. 4 in line 6 chooses valid representatives for C^b and C^d as first nodes, or if on each intra-del level Alg. 3 in line 4 hits a representative for $C^{b,d}$ at the beginning (see Fig. 3(b)).

5 Conclusion

The hierarchical clustering algorithm by Flake et al. [3] returns a set of clusterings at different levels of granularity. The striking feature of graph clusterings computed by this method is that they are guaranteed to yield a certain *expansion*—a bottleneck measure—within and between clusters, tunable by an input parameter α. However, their method, which is based on minimum *s*-*t*-cuts, was restricted to the use of communities, and hence, was not complete. We have proven that the hierarchical approach by Flake et al. [3] remains correct if we introduce a new degree of freedom by permitting the use of arbitrary minimum *s*-*t*-cuts instead of communities. This makes the method more powerful since construction may actually use the most appropriate cut, depending on the application. We have further given a simple characterization of the set of all clustering hierarchies based on minimum *s*-*t*-cuts and have shown that the unrestricted approach is complete, i.e., any clustering hierarchy in this set can be found by choosing the right cuts. This allows for a more detailed analysis of a graph's structure.

Furthermore, we have presented an algorithm which efficiently and fully-dynamically maintains an entire hierarchy of clusterings, as computed by the unrestricted method. Clusterings in the updated hierarchy fulfill the same quality guarantee regarding expansion and, as a secondary criterion, we encourage temporal smoothness, i.e., changes to the clustering hierarchies are kept at a minimum, whenever possible. Thereby, our update algorithm employs the new degree of freedom which allows to reuse clusters independently of their special shape, and thus, saves computational costs and increases temporal smoothness. We also conjecture our new update algorithm, by implementing some small modifications, to be a correct dynamic version of the restricted hierarchical clustering algorithm by Flake et al., i.e., when restricted to maintaining clusters that are communities (see [8] for a first study).

Future work includes the proof of the conjecture, a systematic comparison of our algorithm to other dynamic clustering techniques and the analysis of batch updates.

References

1. U. Brandes and T. Erlebach, editors. *Network Analysis: Methodological Foundations*, volume 3418 of *LNCS*. Springer, February 2005.
2. U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Höfer, Z. Nikoloski, and D. Wagner. On Modularity Clustering. *IEEE TKDE*, 20(2):172–188, February 2008.
3. G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph Clustering and Minimum Cut Trees. *Internet Mathematics*, 1(4):385–408, 2004.
4. R. E. Gomory and T. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, December 1961.
5. D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing*, 19(1):143–155, 1990.
6. R. Kannan, S. Vempala, and A. Vetta. On Clusterings - Good, Bad and Spectral. In *Proc. of FOCS'00*, pages 367–378, 2000.
7. R. Görke and T. Hartmann and D. Wagner. Dynamic Graph Clustering Using Minimum-Cut Trees. In *Proc. of WADS'09*, pages 339–350, 2009.
8. C. Doll. Hierarchical Cut Clustering in Dynamic Scenarios. Student Research Project, Karlsruhe Institute of Technology (KIT), Department of Informatics, February 2011.

Appendix

A Discussion on Incompleteness of Restricted Approach

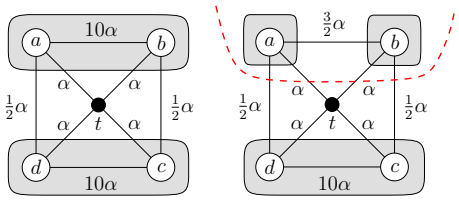


Fig. 4. Cut-clustering hierarchy of one level as example of incomleteness and missing smoothness in case of restriction on cuts.

The left side of Fig. 4 shows graph G_α together with the (unique) cut-clustering $\mathcal{C}(G) = \{\{a,b\},\{c,d\}\}$ (shaded). Now assume the weight on edge $\{a,b\}$ to decrease, as shown on the right side. For the modified graph $G_\alpha^{\ominus}$ exist now two valid cut-clusterings. The previous clustering $\mathcal{C}(G) = \mathcal{C}(G^{\ominus}) = \{\{a,b\},\{c,d\}\}$ is still valid, since the red dashed cut is still a minimum t-a-cut in $G_\alpha^{\ominus}$; but this cut is no community. Thus, this clustering can not be returned by the restricted clustering approach, i.e., this approach is not complete. The shaded clustering $\mathcal{C}'(G^{\ominus}) = \{\{a\},\{b\},\{c,d\}\}$ is a valid cut-clustering whose clusters are communities. Thus, the restricted approach returns $\mathcal{C}'(G^{\ominus})$ although $\mathcal{C}(G^{\ominus}) = \mathcal{C}(G)$ would be the better choice regarding high temporal smoothness.

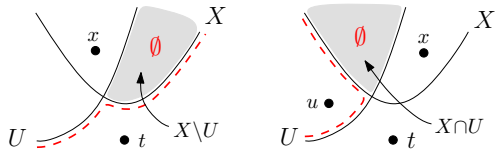
B Omitted Proofs in Section 2

Proof. **(of Lemma 1)** Consider a cut tree $T(G)$ that represents $e_u := (U, V \setminus U)$. Note, that for any minimum u-v-cut in G there exists such a cut tree. Let further denote e_x an edge representing a minimum x-v-cut in $T(G)$. With $x \in U$ it is $c(e_x) \leq c(e_u)$. If $c(e_x) = c(e_u)$ then $(U, V \setminus U)$ is also a minimum x-v-cut. Otherwise, consider the path from v to x, which is segmented by e_u. If e_x was in the segment between v and e_u it would separate v and u and e_u would not represent a minimum u-v-cut. Thus, e_x is in the segment between e_u and x and induces a cut $(X, V \setminus X)$ with $X \subset U$. □

C Omitted Proofs in Section 3

Proof. **(Lemma 5)** Consider $\theta_i := (X, V_{\alpha_i} \setminus X)$ and $\theta_j := (U, V_{\alpha_j} \setminus U)$. We distinguish two cases depending on the shape of θ_j. Case (a) is characterized by $x \in U$, case (b) by $x \in X \setminus U$ and $u \in U \setminus X$ (see also Fig. 5, which is the same as Fig. 1).

(a): We assume $X \setminus U \neq \emptyset$ and show that then $(U \cup X, V_{\alpha_j} \setminus (U \cup X))$ in G_{α_j} is cheaper than θ_j. This contradicts θ_j being a minimum t-u-cut in G_{α_j} and we conclude



(a) $X \subseteq U$ if $x \in U$. (b) $X \cap U = \emptyset$ if $x \notin U$ and $u \notin X$.

Fig. 5. Sketch to Lem. 5

$X \setminus U = \emptyset$, i.e., θ_j does not cut through X (red dashed). In the following we compare different costs expressing them in terms of costs in G and an addend depending on α. For θ_i and $(U \cap X, V_{\alpha_i} \setminus (U \cap X))$ we get:

$$\begin{aligned} c_{\alpha_i}(\theta_i) = c_{\alpha_i}(X, V_{\alpha_i} \setminus X) &= c(X \setminus U, V \setminus (U \cup X)) + c(U \cap X, V \setminus (U \cup X)) \\ &\quad + c(X \setminus U, U \setminus X) \quad + c(U \cap X, U \setminus X) \\ &\quad + \alpha_i |X| \\ c_{\alpha_i}(U \cap X, V_{\alpha_i} \setminus (U \cap X)) &= c(X \setminus U, U \cap X) \quad + c(U \cap X, V \setminus (U \cup X)) \\ &\quad + \alpha_i |U \cap X| \quad + c(U \cap X, U \setminus X) \end{aligned}$$

Since θ_i is a minimum t-x-cut in G_{α_i} it holds $c_{\alpha_i}(U \cap X, V_{\alpha_i} \setminus (U \cap X)) \geq c_{\alpha_i}(\theta_i)$ and we get $c(X \setminus U, U \cap X) \geq c(X \setminus U, V \setminus (U \cup X)) + c(X \setminus U, U \setminus X) + \alpha_i |X \setminus U|$. With $c(X \setminus U, U \setminus X) \geq 0$ it holds in particular

(i) $c(X \setminus U, U \cap X) + c(X \setminus U, U \setminus X) \geq c(X \setminus U, V \setminus (U \cup X)) + \alpha_i |X \setminus U|$

For θ_j and $(U \cup X, V_{\alpha_j} \setminus (U \cup X))$ we get

$$\begin{aligned} c_{\alpha_j}(\theta_j) = c_{\alpha_j}(U, V_{\alpha_j} \setminus U) &= c(U \setminus X, V \setminus (U \cup X)) + c(U \cap X, V \setminus (U \cup X)) \\ &\quad + c(X \setminus U, U \cap X) \quad + c(X \setminus U, U \setminus X) \\ &\quad + \alpha_j |U| \\ c_{\alpha_j}(U \cup X, V \setminus (U \cup X)) &= c(U \setminus X, V \setminus (U \cup X)) + c(U \cap X, V \setminus (U \cup X)) \\ &\quad + c(X \setminus U, V \setminus (U \cup X)) + \alpha_j |U \cup X| \end{aligned}$$

and finally

$$\begin{aligned} c_{\alpha_j}(U \cup X, V \setminus (U \cup X)) - c_{\alpha_j}(\theta_j) &= [c(X \setminus U, V \setminus (U \cup X)) + \alpha_j |X \setminus U|] \\ &\quad - [c(X \setminus U, U \cap X) + c(X \setminus U, U \setminus X)] \\ &< 0 \end{aligned}$$

since $c(X \setminus U, V \setminus (U \cup X)) + \alpha_j |X \setminus U| < c(X \setminus U, U \cap X) + c(X \setminus U, U \setminus X)$ with $\alpha_j < \alpha_i$ applied to (i) and the assumption that $X \setminus U \neq \emptyset$.

(b): We assume $X \cap U \neq \emptyset$ and show that then $(U \setminus X, V_{\alpha_j} \setminus (U \setminus X))$ in G_{α_j} is cheaper than θ_j. This contradicts θ_j being a minimum t-u-cut in G_{α_j} and we conclude $X \cap U = \emptyset$, i.e., θ_j does not cut through X (red dashed). In the following we compare different costs expressing them in terms of costs in G and an addend depending on α. For θ_i and $(X \setminus U, V_{\alpha_i} \setminus (X \setminus U))$ we get:

$$\begin{aligned} c_{\alpha_i}(\theta_i) = c_{\alpha_i}(X, V_{\alpha_i} \setminus X) &= c(X \cap U, U \setminus X) \quad + c(X \setminus U, U \setminus X) \\ &\quad + c(X \cap U, V \setminus (U \cup X)) + c(X \setminus U, V \setminus (U \cup X)) \\ &\quad + \alpha_i |X| \\ c_{\alpha_i}(X \setminus U, V_{\alpha_i} \setminus (X \setminus U)) &= c(X \cap U, X \setminus U) \quad + c(X \setminus U, U \setminus X) \\ &\quad + \alpha_i |X \setminus U| \quad + c(X \setminus U, V \setminus (U \cup X)) \end{aligned}$$

Since θ_i is a minimum t-x-cut in G_{α_i} it holds $c_{\alpha_i}(X \setminus U, V_{\alpha_i} \setminus (X \setminus U)) \geq c_{\alpha_i}(\theta_i)$ and we get $c(X \cap U, X \setminus U) \geq c(X \cap U, U \setminus X) + c(X \cap U, V \setminus (U \cup X)) + \alpha_i |X \cap U|$. With $\alpha_i > \alpha_j$ and the assumption that $X \cap U \neq \emptyset$ it further holds $c(X \cap U, X \setminus U) > c(X \cap U, U \setminus X) + c(X \cap U, V \setminus (U \cup X)) + \alpha_j |X \cap U|$; and with $c(X \cap U, V \setminus (U \cup X)) + \alpha_j |X \cap U| > 0$ it holds in particular

(ii) $c(X \cap U, X \setminus U) + c(X \cap U, V \setminus (U \cup X)) + \alpha_j |X \cap U| > c(X \cap U, U \setminus X)$

For θ_j and $(U \setminus X, V_{\alpha_j} \setminus (U \setminus X))$ we get

$$\begin{aligned} c_{\alpha_j}(\theta_j) = c_{\alpha_j}(U, V_{\alpha_j} \setminus U) &= c(U \setminus X, V \setminus (U \cup X)) + c(X \cap U, V \setminus (U \cup X)) \\ &\quad + c(U \setminus X, X \setminus U) \qquad + c(X \cap U, X \setminus U) \\ &\quad + \alpha_j |U| \\ c_{\alpha_j}(U \setminus X, V_{\alpha_j} \setminus (U \setminus X)) &= c(U \setminus X, V \setminus (U \cup X)) + c(X \cap U, U \setminus X) \\ &\quad + c(U \setminus X, X \setminus U) \qquad + \alpha_j |U \setminus X| \end{aligned}$$

and finally, due to (ii)

$$\begin{aligned} c_{\alpha_j}(U \setminus X, V_{\alpha_j} \setminus (U \setminus X)) - c_{\alpha_j}(\theta_j) &= c(X \cap U, U \setminus X) \\ &\quad - c(X \cap U, V \setminus (U \cup X)) \\ &\quad - c(X \cap U, X \setminus U) - \alpha_j |X \cap U| \\ &< 0 \end{aligned}$$

□

Proof. **(of Lemma 6)** If $r(C') \in C$ then $C' \subseteq C$ holds by Lemma 5(a). Now assume $r(C') \in C' \setminus C$ and let $\hat{C}$ denote the cluster in $\mathcal{C}_j(G)$ containing $r(C')$. By Lemma 1 there exists a minimum t-$r(C')$-cut $(X, V_{\alpha_j} \setminus X)$ in G_{α_j} with $r(C') \in X$ and $X \subseteq \hat{C}$. This contradicts Lemma 5(a) (applied to $r(C')$ and X and C') since $r(C) \in C' \setminus X$, and thus, $C' \not\subseteq X$. Hence, $r(C') \in C$ and $C' \subseteq C$ must hold.

By Lemma 5(a) any minimum t-$r(C')$-cut in G_{α_j} contains $C' \ni r(C)$, and thus, separates t and $r(C)$. On the other hand, $C \ni r(C')$ separates t and $r(C')$. Consequently, C is also a minimum t-$r(C')$-cut in G_{α_j}. □

Proof. **(of Theorem 7)** Consider two cut-clusterings $\mathcal{C}_i(G)$ and $\mathcal{C}_j(G)$ with respect to $\alpha_i > \alpha_j$. Let further denote $C \in \mathcal{C}_j(G)$ a cluster with representative $r(C)$. By Lemma 5 C does not cut through any cluster $C' \in \mathcal{C}_i(G)$ with $r(C) \notin C'$. By Lemma 6 C covers $\hat{C} \in \mathcal{C}_i(G)$ with $r(C) \in \hat{C}$. Thus, $\mathcal{C}_i(G) \leq \mathcal{C}_j(G)$. □

D Omitted Proofs in Section 4

Proof. **(of Lemma 9)** Let $\mathcal{C}_j(G)$ and $\mathcal{C}_i(G^{\oplus\ominus})$ denote two cut-clusterings with $\alpha_i > \alpha_j$. Let $D_{\alpha_j}^{\oplus\ominus}$ denote the graph resulting from $G_{\alpha_j}^{\oplus\ominus}$ by contracting each cluster $C' \in \mathcal{C}_i(G^{\oplus\ominus})$ into a node $[C']$ and associating $[C']$ with $r(C')$. In this situation the improved SCC is applied in order to compute a new cut-clustering $\mathcal{C}_j(G^{\oplus\ominus})$. Hence, u is a node in $D_{\alpha_j}^{\oplus\ominus}$, θ is a minimum t-u-cut in $D_{\alpha_j}^{\oplus\ominus}$ and $C \in \mathcal{C}_j(G)$ contains $r(u)$. Now assume that $c^{\oplus\ominus}(\theta) = c^{\oplus\ominus}(C)$ in $G_{\alpha_j}^{\oplus\ominus}$.

By Lemma 5 (and the deduced 1-1-correspondence of minimum t-u-cuts in $D^{\oplus\ominus}_{\alpha_j}$ and t-$r(u)$-cuts in $G^{\oplus\ominus}_{\alpha_j}$) θ is a minimum t-$r(u)$-cut in $G^{\oplus\ominus}_{\alpha_j}$, and C is also a minimum t-$r(u)$-cut in $G^{\oplus\ominus}_{\alpha_j}$ due to $r(u) \in C$ and the assumption above. Thus, again by Lemma 5, C is a minimum t-u-cut in $D^{\oplus\ominus}_{\alpha_j}$ with $u \subseteq C$. □

Proof. **(of Lemma 10)** Consider $M := \mathcal{C}(G) \setminus \{C^{b,d}\}$. We decompose M into a set A of clusters C that are still minimum t-$r(C)$-cuts in $G^{\ominus}_{\alpha}$ and into a set B of clusters C for which a new minimum t-$r(C)$-cut θ exists that is cheaper than C. Note that θ separates b and d. Conserning the clusters in B, by a result in [7] all corresponding cuts θ can be adjusted such that they do neither cross each other nor the clusters in A and such that each θ contains its cluster C (for the latter cf. Fact e)). Thus, there exists a run of SCC that reaches a set $X \subseteq (V \setminus \bigcup_{C \in M} C) = C^{b,d}$ in line 2 while in the intermediate clustering $\mathcal{C}(G^{\ominus})$ b and d are still in different clusters C^1 and C^2 which cover all vertices in $C^{b,d} \setminus X$. Note that the vertices in X are not clustered yet. Since in all cut-clusterings $\mathcal{C}(G^{\ominus})$ b,d share a cluster by precondition, there must exist a vertex $x \in X$ with a minimum t-x-cut θ' in $G^{\ominus}_{\alpha}$ containing b and d, i.e., containing $C^1 \cup C^2$. This is, $r(C^{b,d})$ is either in θ' or in the remaining set of free vertices not clustered yet. Since θ' is also a minimum t-x-cut in G_{α}, in the first case it follows that θ' is at most as cheap as $C^{b,d}$ (as $r(C^{b,d}) \in \theta'$) and with $x \in C^{b,d}$ we get $c^{\ominus}(C^{b,d}) = c(C^{b,d}) = c(\theta') = c^{\ominus}(\theta')$, and thus, $C^{b,d}$ is a minimum t-x-cut in $G^{\ominus}_{\alpha}$. Otherwise, $r(C^{b,d})$ is still free and there exists a minimum t-$r(C^{b,d})$-cut in $G^{\ominus}_{\alpha}$ that does not separate b and d since θ' covers these vertices. Thus, $C^{b,d}$ is still a minimum t-$r(C^{b,d})$-cut in $G^{\ominus}_{\alpha}$.

The second assertion follows directly since there exits no cut-clustering $\mathcal{C}_j(G^{\ominus})$ with b and d in diffrerent clusters if b,d share a cluster in $\mathcal{C}_i(G^{\ominus})$, by Theorem 7. □

Proof. **(of Lemma 12)** Since C induces a minimum t-$r(C)$-cut in $G^{\oplus\ominus}_{\alpha_j}$ with $r(C') \in C$, by Lemma 1 there exists a minimum t-$r(C')$-cut θ in $G^{\oplus\ominus}_{\alpha_j}$ nested in C. Thus, θ does not separate b and d and all minimum t-$r(C')$-cuts θ' in $G^{\oplus\ominus}_{\alpha_i}$ do also not separate b and d since they are nested in θ by Lemma 5(a) (consider $u = x$). This is, each θ' has the same weight as C' in G_{α_i} and $G^{\oplus\ominus}_{\alpha_i}$, and thus, C' induces a minimum t-$r(C')$-cut in $G^{\oplus\ominus}_{\alpha_i}$ by Observation 2.

We can even show that C remains a cut side in $G^{\oplus\ominus}_{\alpha_j}$ with respect to $r(C)$ iff C remains a cut side in $G^{\oplus\ominus}_{\alpha_j}$ with respect to any representative x in $\mathcal{C}_j(G)$.

($\Rightarrow$): Since C does not separate b and d it has the same weight in both graphs $G^{\oplus\ominus}_{\alpha_j}$ and G_{α_j}. According to Lemma 1 there exists a minimum t-x-cut in $G^{\oplus\ominus}_{\alpha_j}$ that is nested in C and thus, does not separate b and d. As a result C is still a cut side in $G^{\oplus\ominus}_{\alpha_j}$ with respect to x.

($\Leftarrow$): Assume x being the representative designated in $\mathcal{C}_j(G)$. Apply ($\Rightarrow$). Finally, the representatives of C in G_{α_j} and $G^{\oplus\ominus}_{\alpha_j}$ are the same. □

Lemma 17. *Let $\mathcal{C}(G)$ be an intra-del clustering fulfilling the copy-property and let $D^{\ominus}_{\alpha}$ result from $G^{\ominus}_{\alpha}$ by contracting clusters of a cut-clustering for $G^{\ominus}$ with respect to a parameter value bigger than α. Then Algorithm 3 returns $\mathcal{C}(G)$ with an updated representative for $C^{b,d}$ (the other representatives are still valid, cf. Fact b)).*

Proof. Clustering $\mathcal{C}(G)$ is valid for $G^{\ominus}$ on level α and $D_{\alpha}^{\ominus}$ represents a cut-clustering for $G^{\ominus}$ on a lower level. Hence, each cluster in $\mathcal{C}(G)$ is a proper union of nodes in $D_{\alpha}^{\ominus}$ according to Theorem 7. Thus, Algorithm 3 reaches the while-loop in line 4.

For each u in $D_{\alpha}^{\ominus}$ with $u \subseteq C^{b,d}$ there exists a minimum t-$r(u)$-cut in $G_{\alpha}^{\ominus}$ that is nested in $C^{b,d}$, by Lemma 5 (1-1-correspondence of t-u-cuts in $D_{\alpha}^{\ominus}$ and t-$r(u)$-cuts in $G_{\alpha}^{\ominus}$) and Lemma 1. In particular, the community U of u in $D_{\alpha}^{\ominus}$ equals the community of $r(u)$ in $G_{\alpha}^{\ominus}$ and is nested in $C^{b,d}$. Consequently, Algorithm 3 skips line 6. Furthermore, it is $c^{\ominus}(U) \leq c^{\ominus}(C^{b,d}) = c(C^{b,d})$.

Let y denote a node in $D_{\alpha}^{\ominus}$ containing a representative of $C^{b,d}$ regarding $G^{\ominus}$. If $y \in U$ (regarding the current u in line 4) it is $c^{\ominus}(U) = c^{\ominus}(C^{b,d}) = c(C^{b,d})$. Thus, $C^{b,d}$ is a minimum t-u-cut in $D_{\alpha}^{\ominus}$ and a minimum t-$r(u)$-cut in $G_{\alpha}^{\ominus}$. In this case, Algorithm 3 returns $\mathcal{C}(G)$ with u as representative of $C^{b,d}$.

Otherwise, if $y \notin U$, y is still a free node in $D_{\alpha}^{\ominus}$ not clustered yet in the next iteration of the while-loop. By induction we finally conclude that there exists an iteration step where y is covered by the current community U and the former case applies.

Due to the above arguments we further see that, given a cut-clustering $\mathcal{C}(G)$ fulfilling the copy-property, the while-loop by itself copies $\mathcal{C}(G)$ and updates the representative of $C^{b,d}$. □

Lemma 18. *Let $\mathcal{C}_i(G)$ denote an intra-del clustering on level i, and let further denote $\hat{C} \in \mathcal{C}_j(G)$ a cluster on a higher level j inducing a reusable subtree according to Theorem 13. Consider cluster $C \in C_i(G) \setminus C^{b,d}$ being part of this subtree. Then Algorithm 3 returns a new cut-clustering $\mathcal{C}_i(G^{\ominus})$ that contains C as cluster.*

Proof. If $\mathcal{C}_i(G)$ fulfills the copy-property the assertion holds by Lemma 17. Otherwise, Algorithm 3 reaches the second while-loop in line 9 or it applies line 2 or line 6. According to Theorem 7 $\hat{C}$ and C are proper unions of nodes in $D_{\alpha_i}^{\ominus}$, and the nodes in $\hat{C}$ are clustered independently by Corollary 8. Thus, in all tree cases mentioned above the nodes in $\hat{C}$ are still free nodes not clustered yet when Algorithm 3 turns into ISCC. For each node in $\hat{C}$ considered as a sink in intilligent SCC the new cut is a subset of $\hat{C}$, and thus, does not separate b and d and has the same weight as the previous minimum cut. Since ISCC is supposed to take the nodes containing the representatives of clusters in $\mathcal{C}_i(G)$ first the intelligent version finds and reconstructs all clusters in $\mathcal{C}_i(G)$ covered by $\hat{C}$. □

Lemma 19. *Let $\mathcal{C}_i(G)$ denote an inter-del clustering on level i, and let further denote $\hat{C} \in \mathcal{C}_j(G)$ a cluster on a higher level j inducing a reusable subtree according to Theorem 13. Consider cluster $C \in C_i(G) \setminus C^{b,d}$ being part of this subtree. Then LU applied to $D_{\alpha_i}^{\ominus}$ resulting from $G_{\alpha_i}^{\ominus}$ by contraction strategy 1 returns a new cut-clustering $\mathcal{C}_i(G^{\ominus})$ that contains C as cluster.*

Proof. We expect the reader to have [7] at hand since we omit a renewed description of LU in the case of inter-cluster deletion.

According to strategy 1 applied in an inter-del case $\hat{C}$ and C are proper unions of nodes in $D_{\alpha_i}^{\ominus}$, and the nodes in $\hat{C}$ are clustered independently by Corollary 8. For each node in $\hat{C}$ considered as a sink by LU the new cut is a subset of $\hat{C}$, and thus, does not separate b and d and has the same weight as the previous minimum cut. Since LU

to take the nodes containing the representatives of clusters in $\mathcal{C}_i(G)$ first LU finds and reconstructs all clusters in $\mathcal{C}_i(G)$ covered by $\hat{C}$. □

E Discussion on Different Contraction Strategies

When constructing a cut-clustering on level i HSCC contracts clusters on the level below in order to speed up the computation: contractions shrink the instance for cut computations in SCC and reduce the risk of computing unnecessary cuts. Our new update approach also contracts clusters in $G_{\alpha_i}^{\oplus\ominus}$, however, depending on properties of the different modification cases and the parts of a new hierarchy known so far. The two strategies we describe in the following provide about the same speed up as traditional contractions and do not change costs compared to an intelligent hierarchical approach. However, we prefer those in order to afford temporal smoothness. Lemma 20 confirms that LU applied with strategy 1 returns a valid cut-clustering. The correctness of (intelligent) SCC applied with strategy 2 is given by Lemma 21.

If no clustering is known on the level below, we contract each cluster in $C \in \mathcal{C}_i(G)$ that remains a cut side according to Fact a) and associate $[C]$ with $r(C)$ (strategy 1). We will use this strategy only for inter-del levels, which we update by applying LU. According to Fact d) this needs at most $|\mathcal{C}_i(G)| - 2$ cut computations.

The instances constructed by this strategy, without any knowledge about the level below, differ from the corresponding instances in a hierarchical approach in size and structure of nodes; in particular the representatives in both instances are not comparable. However, the number of cuts used for an inter-del update also develops hierarchically regarding several levels, and the considered representatives already induced clusters before the change. In all likelyhood, they are again a good choice such that we expect only a low risk of computing cuts without need. Thus, we estimate the costs for constructing inter-del level i using this strategy together with LU by $\mathcal{T}([i,i], G^{\ominus})$, ignoring the missing speed-up for cut computations due to a bigger instance.

If we already know a new clustering $\mathcal{C}_{i-1}(G^{\oplus\ominus})$, we contract the clusters $X \in \mathcal{C}_{i-1}(G^{\oplus\ominus})$ as the hierarchical approach does. Additionally, we contract each cluster in $C \in \mathcal{C}_i(G)$ that corresponds to a proper union of nodes in the graph contracted so far and withal remains a cut side according to Fact a). We associate $[C]$ with $r(X)$, where $X \in \mathcal{C}_{i-1}(G^{\oplus\ominus})$ contains $r(C)$ (strategy 2). The final graph is at most as big as a traditionally contracted instance and uses a subset of its representatives. Thus, we estimate the costs of (intelligent) SCC using the new instance by $\mathcal{T}([i,i], G^{\oplus\ominus})$, ignoring the additional cost for contracting the cut sides.

Lemma 20. *Let $\mathcal{C}_i(G)$ denote an inter-del clustering and let $D_{\alpha_i}^{\ominus}$ denote the graph resulting from $G_{\alpha_i}^{\ominus}$ by contracting vertices according to strategy 1. Then LU applyed to $D_{\alpha_i}^{\ominus}$ still returns a cut-clustering for $G^{\ominus}$.*

Proof. We expect the reader to have [7] at hand since we omit a renewed description of LU in the case of inter-cluster deletion. We show that LU applied to $D_{\alpha_i}^{\ominus}$ does exactly the same as LU applied to $G_{\alpha_i}^{\ominus}$.

The only difference between $D_{\alpha_i}^{\ominus}$ and $G_{\alpha_i}^{\ominus}$ is that C^b and C^d form nodes in $D_{\alpha_i}^{\ominus}$. For inter-cluster deletion, LU applied to $G_{\alpha_i}^{\ominus}$ considers C^b and C^d as cut sides of $r(C^b)$ and

$r(C^d)$ that are included in a tentative clustering from beginning until they are covered by another cut. Thus, for the run of LU it is irrelevant whether C^b and C^d are contracted or not. Vice versa, in $D^{\ominus}_{\alpha_i}$ the nodes $[C^b]$ and $[C^d]$ are associated with $r(C^b)$ and $r(C^d)$. □

Lemma 21. *Let $D^{\oplus\ominus}_{\alpha_i}$ denote the graph resulting from $G^{\oplus\ominus}_{\alpha_i}$ by contracting vertices according to strategy 2. Then (intelligent) SCC applyed to $D^{\oplus\ominus}_{\alpha_i}$ still returns a cut-clustering for $G^{\oplus\ominus}$.*

Proof. Let u denote a node in $D^{\oplus\ominus}_{\alpha_i}$. We show that there exists a minimum t-$r(u)$-cut in $G^{\oplus\ominus}_{\alpha_i}$ that contains u and respects all nodes in $D^{\oplus\ominus}_{\alpha_i}$, i.e., does not cut through any node in $D^{\oplus\ominus}_{\alpha_i}$. Then each minimum t-u-cut in $D^{\oplus\ominus}_{\alpha_i}$ is also a minimum t-$r(u)$-cut in $G^{\oplus\ominus}_{\alpha_i}$ and it holds that applying SCC is feasible. Finally we confirm that also ISCC works correctly on $D^{\oplus\ominus}_{\alpha_i}$. Let A denote the set of nodes in $D^{\oplus\ominus}_{\alpha_i}$ that correspond to a cut sides in $\mathcal{C}_i(G)$ and B the set of nodes that correspond to clusters in $\mathcal{C}_{i-1}(G^{\oplus\ominus})$ and are not in A.

- If u is a cut side $C \in A$ define $\theta := u$ which is a minimum t-$r(C)$-cut in $G^{\oplus\ominus}_{\alpha_i}$. Since $r(u) = r(X)$ with $r(C) \in X \in \mathcal{C}_{i-1}(G^{\oplus\ominus})$ θ is also a minimum t-$r(u)$-cut. Obviously θ contains u and respects the nodes in $D^{\oplus\ominus}_{\alpha_i}$.
- If u is a cluster $X \in B$ there exists a minimum t-$r(X)$-cut θ in $G^{\oplus\ominus}_{\alpha_i}$ that contains X and respects the clusters in $\mathcal{C}_{i-1}(G^{\oplus\ominus})$, according to Lemma 5. If θ cuts through a cut side $C \in A$, we can reshape θ according to Lemma 3 since $r(X) \notin C$. This yields a minimum t-$r(X)$-cut that further respects all nodes in A since the reshaping does not cause any new conflicts. With $X = u$ and $r(X) = r(u)$ θ is a minimum t-$r(u)$-cut that contains u.

Finally, consider a minimum t-u-cut in $D^{\oplus\ominus}_{\alpha_i}$ and let $C' \in \mathcal{C}_i(G)$ denote the cluster containing $r(u)$. If $u \in B$ the proof is analog to Lemma 9. If $u \in A$ we know from above that $u = C'$ is a minimum t-u-cut in $D^{\oplus\ominus}_{\alpha_i}$. Thus, also ISCC works correctly on $D^{\oplus\ominus}_{\alpha_i}$. □