Complete Hierarchical Cut-Clustering: A Case Study on Modularity and Expansion[⋆,⋆⋆]

Michael Hamann, Tanja Hartmann and Dorothea Wagner

Department of Informatics, Karlsruhe Institute of Technology (KIT)
mhamann@ira.uka.de, {t.hartmann, dorothea.wagner}@kit.edu

Abstract. We present a simple and efficient method for constructing a cut-clustering hierarchy as introduced by Flake et al. Cut-clusterings excel by a clearly indicated membership of the vertices to the clusters due to strong connections inside the clusters compared to only weak connections outside. Their coarseness depends on a parameter that provides a quality guarantee in terms of expansion, which is NP-hard to compute. In this work we introduce a parametric search approach that guarantees the completeness of the resulting hierarchy and supersedes the necessity of choosing feasible parameter values in advance or applying a binary search to find those values. Our method is easy to implement and in a brief running time experiment it turns out to be significantly faster than a binary search. We further investigate the resulting clusterings with respect to modularity, a quality measure widely used in practice. In this context we propose a parameter-free approach that helps to estimate how well a graph can be generally clustered by cut-clusterings. With due regard to this estimation the algorithm of Flake et al. competes surprisingly well with respect to reliable reference clusterings, although it is not designed to optimize modularity. Further experiments focusing on the given guarantee on expansion exhibit that the actual expansion is even better than guaranteed and compared to trivial bounds the guarantee constitutes a true gain of knowledge.

1 Introduction

Clustering a graph means finding internally dense subgraphs, called clusters, that are only loosely connected to the remainder of the graph. The growing interest in graph clustering during the last decades has been driven by applications in physics and biology as well as sociology and many other fields. Attempts of formalizing the properties that characterize a set of good clusters resulted in a variety of different quality measures, which still affect the design of algorithms. Flake et al. [3], however, postulate a different approach. They search for clusterings where the membership of the vertices to a cluster is clearly indicated by strong connections inside the cluster while the connections to other clusters are weaker, a condition not expressible by any of the previous measures. This is what we call a *tight* clustering in the following. The strict behavior of tight clusterings—not clearly assigned vertices remain unclustered—is desirable whenever it is essential that ambiguous cases are interpreted by human experts as for example

⋆ This work was partially supported by the DFG under grant WA 654/15 within the Priority Programme "Algorithm Engineering".

⋆⋆ Also submitted to SEA 2012

in sociology applications. Flake et al. introduce an algorithm that exploits properties of minimum s-t-cuts in order to find a tight clustering depending on a parameter. This parameter controls the coarseness of the resulting clustering and constitutes a guarantee on intra-cluster expansion, a common quality index, which is hard to compute. Different parameter values result in at most $n-1$ different clusterings, which form a hierarchy. Having a hierarchy of tight clusterings at hand, it is then possible to choose the best clustering with respect to any quality measure that suits a particular application.

Our Contribution. We characterize four different types of tight clusterings and introduce the problem Tight Clustering, which asks for an optimal tight clustering of a designated type with respect to a given quality measure. In this light we investigate the behavior of the cut-clustering algorithm of Flake et al. We develop a simple parametric search approach for efficiently constructing a complete cut-clustering hierarchy and provide a brief running time experiment in which our method outperforms a naive binary search, which contrariwise gives no guarantee on finding all different clusterings in the hierarchy. We further conduct a comparative analysis of the modularity values reachable by cut-clusterings. As reference we use a greedy modularity-based approach. This, however, is not restricted to tight clusterings and thus benefits from a larger search space. In order to assess the results as fair as possible we, hence, introduce a parameter-free method that on the other hand helps to exclude the existence of tight clusterings in instances of special shape. Our experiments demonstrate that the algorithm of Flake et al. competes surprisingly well with respect to the objective of Tight Clustering and modularity, and we reveal that the restriction to tight clusterings not necessarily causes a substantial loss of modularity. Finally, we compare the guaranteed intra-cluster expansion of the cut-clusterings to the expansion of the modularity-based references. Since expansion is hard to compute, we consider lower bounds. Our study gives evidence that trivial bounds do not match up to the given guarantee, and an analysis of special non-trivial bounds further indicates that also the true expansion of the cut-clusterings surpasses the modularity-based references.

Related Work. The cut-clustering algorithm of Flake et al. [3], CutC as a shorthand, is the protagonist in our work. The notion of tight clusterings is introduced in [1], however, there the clusters are called web-communities. The implementation of the modularity-based reference algorithm, which is a greedy approach based on vertex moves [2], we took from Lisowski [15]. The notion of modularity was introduces in [10], Montgolfier et al. [7] study the asymptotic behavior of modularity in selected graph classes, which helps us to explain some outlier results in our experiments. Apart from these, there is a huge number of publications on clustering algorithms and quality measures, for an overview see [8]. In our proofs we finally exploit some insights and lemmas on minimum s-t-cuts that date back to Gomory and Hu [4] and Gusfield [5].

2 Preliminaries

Throughout this work we consider a simple, undirected, weighted graph $G = (V, E, c)$ with vertex set V, edge set E and a non-negative edge cost function c. In unweighted graphs we count each edge by one. We denote the number of vertices (edges) by $n := |V|$ ($m := |E|$) and the costs of a set $E' \subseteq E$ by $c(E') := \sum_{e \in E'} c(e)$. Whenever we consider the degree $\deg(v)$ of $v \in V$, we implicitly mean the sum of all edge costs incident to v.

Table 1. Overview of different types of tight subgraphs.

A subgraph $S \subseteq V$ is a			WC	SC	sSC	ES
WC	$\forall u \in S$	$c(\{u\}, S \setminus \{u\}) > c(\{u\}, V \setminus S)$	x			
SC	$\exists s \in S : \forall U \subset S, s \notin U$	$c(U, S \setminus U) \geq c(U, V \setminus S)$		x		
strict SC	$\exists s \in S : \forall U \subset S, s \notin U$	$c(U, S \setminus U) > c(U, V \setminus S)$		x	x	
ES	$\forall U \subset S$	$c(U, S \setminus U) > c(U, V \setminus S)$	x	x	x	x

With $S, T \subset V$ we write $c(S,T)$ for the costs of all edges having one endpoint in S and one in T. If S, T induce a cut in G, $c(S,T)$ describes the costs of this cut. Let (S,T) denote a minimum s-t-cut, $s \in S$, $t \in T$. The cut (S,T) is called the *community cut* of s with respect to t if $|S|$ is minimum for all minimum s-t-cuts in G. Set S is the unique *community* of s while s is a *representative* of S, denoted by $r(S)$, not necessarily unique.

We reserve the term *node* for compound vertices of abstracted graphs, which may contain several basic vertices. A *contraction* by $C \subseteq V$ means replacing the set C in G by a single node and leaving this node adjacent to all former adjacencies u of vertices in C, with costs equal to the sum of all former edges between C and u.

Our understanding of a *clustering* $\mathcal{C}(G)$ is a partition of V into subsets $C^1, \ldots, C^k$, which define vertex-induced subgraphs, called *clusters*. A cluster is called *trivial* if it corresponds to a connected component. A vertex that forms a non-trivial singleton cluster we consider as *unclustered*. A clustering is *trivial* if it consists of trivial clusters or if $k = n$, i.e., all vertices are unclustered. A *hierarchy of clusterings* is a sequence $\mathcal{C}_1(G) \leq \cdots \leq \mathcal{C}_r(G)$ 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*. Furthermore, we distinguish four types of *tight* clusterings: Those consisting of *web-communities* (WC), *source-communities* (SC), *strict source-communities* and *extreme sets* (ES). Table 1 gives exact definitions of the notions of the *tight* subgraphs in the clusterings and an overview in which other types a particular type is nested. SCs are characterized by the following (for a proof see the full version [16]).

Lemma 1. *A set $S \subset V$ is a source-community of a vertex $s \in S$ if and only if there exists a set $T \subset V$ such that $(S, V \setminus S)$ is a minimum s-T-cut in $G = (V, E, c)$. We call s the representative $r(S)$ of S.*

Remark 1. For an SC S it holds $c(S, V \setminus S) \leq deg(s)$. Otherwise it would be $c(S \setminus \{s\}, S) > c(S \setminus \{s\}, V \setminus S)$. Furthermore, let $S \subset V$ denote a community of s with respect to t. Then, due to the uniqueness and minimality of S, $V \setminus S$ is the unique (inclusion)maximal SC of t with $s \notin V \setminus S$, and S is a strict SC of s.

Tight Clustering. Tight Clustering is the problem of finding a tight clustering of maximum quality. Regarding the different types of tight clusters in Table 1 and the variety of existing quality indices, Tight Clustering encompasses a whole family of problems. In this work, we focus on modularity as objective quality measure, and we require the clusters to be at least source-communities, as SCs are closely related to minimum s-t-cuts, which can be efficiently calculated. In contrast, decomposing a graph into k web-communities is NP-hard [3], whereas all extreme sets can be computed in

Table 2. Overview of intra-cluster expansion bounds.

cut expansion $\Psi(S, C \setminus S)$:	$\frac{c(S,C\setminus S)}{\min\{\lvert S\rvert, \lvert C\setminus S\rvert\}}$	guarantee $\Psi_g(\mathcal{C})$:	by parameter
trivial lower bound $\Psi_\ell(C)$:	$\frac{c(A,C\setminus A)}{\lfloor \lvert C\rvert /2 \rfloor}$	non-trivial bound $\Psi_a(\mathcal{C})$:	$\min_{C\in\mathcal{C}} \Psi_a(C)$

$O(nm + n^2 \log n)$ time [6]. The latter are either nested or disjoint. Thus, for ESs Tight Clustering can be efficiently solved if the quality index is easy to calculate. We are interested in how well CutC approximates a solution of Tight Clustering with respect to modularity and SCs.

Quality Measures. A *quality measure* for clusterings is a mapping to real numbers. Depending on the measure, either high or low values correspond to high quality. The measures considered in this work, modularity and intra-cluster expansion, indicate high quality by high values.

Modularity bases on the total edge costs covered by clusters. The values range between -0.5 and 1 and express the significance of a given clustering compared to a random clustering. Formally, the modularity $\mathcal{M}(\mathcal{C})$ of a clustering $\mathcal{C}$ is defined as $\mathcal{M}(\mathcal{C}) := \sum_{C\in\mathcal{C}} c(E_C)/c(E) - \sum_{C\in\mathcal{C}} (\sum_{v\in C} deg(v))^2/4c(E)^2$, where E_C denotes the set of edges with both endpoints in C.

The intra-cluster expansion of a clustering derives from the expansion defined for cuts. The expansion $\Psi(S, C \setminus S)$ of a cut $(S, C \setminus S)$ in a cluster C evaluates the ratio of the costs and the size of the cut and is given in Table 2. The expansion $\Psi(C)$ of a cluster equals the minimum expansion of all cuts in C. The intra-cluster expansion $\Psi(\mathcal{C})$ of a clustering finally is the minimum expansion of all clusters in $\mathcal{C}$. Note that expansion is not defined for singleton clusters. Thus, only non-singleton clusters count for $\Psi(\mathcal{C})$. Trivial clusterings consisting of singleton clusters are omitted in the respective experiments. Computing $\Psi(C)$ is known to be NP-hard, however, a trivial lower bound $\Psi_\ell(C)$ can be easily determined from any global minimum cut $(A, C \setminus A)$ (see Table 2). The expansion of such a minimum cut further constitutes an upper bound on $\Psi(C)$. We denote this by $\Psi_u(C)$. In our experiments we compare the analog trivial bounds $\Psi_\ell(\mathcal{C})$ and $\Psi_u(\mathcal{C})$ to the guarantee given by the parameter of CutC, which we denote by $\Psi_g(\mathcal{C})$. We further consider an alternative non-trivial lower bound $\Psi_a(\mathcal{C})$ resulting from individually applying CutC to the subgraphs induced by the clusters. For those subgraphs we want CutC to return the trivial clustering that consists of the whole subgraph. This can be reached by accordingly choosing the parameter of CutC, which controls the coarseness. The chosen parameter value then constitutes a non-trivial bound $\Psi_a(C)$ on the expansion of the considered cluster/subgraph. Since this method considers the clusters as independent instances ignoring the edges between the clusters, the resulting bound $\Psi_a(\mathcal{C})$ often lies above $\Psi_g(\mathcal{C})$.

3 The Algorithms

In this section we review the parametric cut-clustering approach of Flake et al. and introduce a simple method for efficiently computing all different clusterings in the parameter range. We further give a short idea of the parameter-free approach we use to exclude the existence of tight clusterings in some particular cases.

3.1 Basic and Hierarchical Cut-Clustering

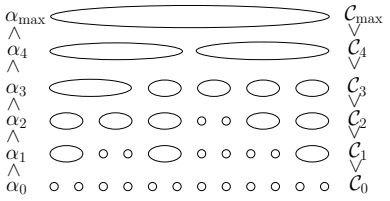


Fig. 1. Clustering hierarchy by CutC. Note that $\alpha_{\max} < \alpha_0$, $\mathcal{C}_{\max} > \mathcal{C}_0$.

The basic CutC algorithm of Flake et al. works as follows: Given a graph G and parameter $\alpha > 0$, as a preprocessing step augment G by inserting an artificial vertex t and connecting t to each vertex in G by an edge of costs α. Denote the resulting graph by $G_\alpha = (V_\alpha, E_\alpha, c_\alpha)$. Then apply CutC by iterating V and computing a community with respect to t for each vertex not yet contained in a community. Since communities are either disjoint or nested we finally get a set of (inclusion)maximal communities decomposing V. We call such a decomposition a cut-clustering. It inherits the uniqueness of the communities and, according to Remark 1, embodies a tight clustering $\mathcal{C}$ of strict SCs. Furthermore it holds $\Psi(\mathcal{C}) \geq \alpha$. Applying CutC iteratively with decreasing parameter values yields a hierarchy of at most n different clusterings (cp. Fig. 1). Note that for α_0 equal to the maximum edge costs in G CutC returns the trivial clustering consisting of singletons, while $\alpha_{\max} = 0$ yields the connected components.

Algorithm 1 describes a naive hierarchical approach, which exploits the hierarchical nesting property in order to shrink the next instance by contracting the previous clusters (line 5). The crucial point with this approach, however, is the choice of α. If we choose the next value too high we get the previous clustering again, which implies unnecessary effort. If we choose the next value too low we possibly miss a meaningful clustering. Flake et al. propose a binary search approach for the choice of α, however, this necessitates a discretization of the parameter range and still does not prevent missing clusterings. That is why we introduce a simple parametric search approach for constructing a complete hierarchy.

Algorithm 1: HIERARCHICAL CUTC

Input: $\alpha_0 = \max\{c(e) | e \in E\}$, G_{α_0}
1 $\mathcal{C}_0(G) \leftarrow \{\{v\} \mid v \in V\}$; $i \leftarrow 0$
2 **forall the** $v \in V$ **do** $r(\{v\}) \leftarrow v$
3 **while** $\mathcal{C}_i(G)$ *differs from conn. comp.* **do**
4 choose $\alpha_{i+1} < \alpha_i$
5 $G'_{\alpha_{i+1}} \leftarrow$ contract $\mathcal{C}_i(G)$ in $G_{\alpha_{i+1}}$
6 $\mathcal{C}_{i+1}(G) \leftarrow$ CutC($G'_{\alpha_{i+1}}$)
7 $i \leftarrow i+1$
8 **return** $\mathcal{C}_0, \ldots, \mathcal{C}_i$

3.2 Simple Parametric Search Approach

Our simple approach for constructing a complete hierarchy of cut-clusterings exploits the properties of cut-cost functions. The *cut-cost function* ω_C of a set $C \in V$ is a linear function depending on α that represents the costs of cut $(C, V_\alpha \setminus C)$ in G_α.

$$\begin{aligned} \omega_C : \mathcal{R}_0^+ &\longrightarrow [c(C, V \setminus C), \infty) \subset \mathcal{R}_0^+ \\ \omega_C(\alpha) &:= c(C, V \setminus C) + |C|\, \alpha \end{aligned}$$

For two consecutive hierarchy levels $\mathcal{C}_i < \mathcal{C}_{i+1}$ we call $\hat{\alpha}$ the breakpoint if CutC returns $\mathcal{C}_i$ for $\hat{\alpha}$ and $\mathcal{C}_{i+1}$ for $\hat{\alpha} - \varepsilon$. By construction, each breakpoint is an intersection point of the cut-cost functions of two clusters $C_i \subset C_j$. Thus, the idea is to compute relevant intersection points and check if they yield new clusterings.

Theorem 1. *Let $\mathcal{C}_i < \mathcal{C}_j$ denote two different clusterings with parameter values $\alpha_i > \alpha_j$. In time $O(|\mathcal{C}_i|)$ a parameter value α_m with $\alpha_j < \alpha_m \leq \alpha_i$ can be computed such that 1) $\mathcal{C}_i \leq \mathcal{C}_m < \mathcal{C}_j$, and 2) $\mathcal{C}_m = \mathcal{C}_i$ implies that α_m is the breakpoint between $\mathcal{C}_i$ and $\mathcal{C}_j$.*

Proof. We observe that 1) two functions ω_{C_i} and ω_{C_j} with $C_i \subset C_j$ intersect, as by construction it is $c(C_j, V \setminus C_j) \leq c(C_i, V \setminus C_i)$ and $|C_j| > |C_i|$. For the intersection point α' it holds $\omega_{C_i}(\alpha) > \omega_{C_j}(\alpha)$ if $\alpha < \alpha'$ and $\omega_{C_i}(\alpha) < \omega_{C_j}(\alpha)$ if $\alpha > \alpha'$ (cp. Fig. 2). Consider 2) a cluster $C_j \in \mathcal{C}_j$, a child $C_i \in \mathcal{C}_i$ of C_j (i.e., $C_i \subset C_j$) and the intersection point α'_j of ω_{C_i} and ω_{C_j}. It is $\alpha'_j \leq \alpha_i$ as otherwise (by 1) $\omega_{C_j}(\alpha_i) < \omega_{C_i}(\alpha_i)$, and thus, $C_j \ni r(C_i)$ would be a cheaper community in G_{α_i}. If 3) $r(C_j) \in C_i$ it is $\alpha'_j > \alpha_j$. Otherwise (by 1) $\omega_{C_i}(\alpha_j) \leq \omega_{C_j}(\alpha_j)$, $|C_i| < |C_j|$, and C_i would be a smaller community in G_{α_j}.

To determine α_m let $\hat{\alpha}_j := \max_{C_i \subset C_j}\{\alpha'_j\}$, suppose $\hat{\alpha}_j := \infty$ if C_j is also a cluster on level $\mathcal{C}_i$. Finally define $\alpha_m := \min_{C_j \in \mathcal{C}_j}\{\hat{\alpha}_j\}$. Due to 1)–3), $\alpha_j < \alpha_m \leq \alpha_i$ and α_m can be computed in time $O(|\mathcal{C}_i|)$. Let C_m denote the result of CutC when applied to G_{α_m}.

Claim 1: $\mathcal{C}_m \neq \mathcal{C}_j$. Let $C_j \in \mathcal{C}_j$ denote a cluster with $\hat{\alpha}_j = \alpha_m$. It holds $\alpha_m \geq \alpha'_j$ for all children C_i of C_j and (by 1) $\omega_{C_i}(\alpha_m) \leq \omega_{C_j}(\alpha_m)$, $|C_i| < |C_j|$. This is, $C_j \notin \mathcal{C}_m$.

Claim 2: If $\mathcal{C}_m = \mathcal{C}_i$ then α_m is the breakpoint between $\mathcal{C}_i$ and $\mathcal{C}_j$. Let $C_i \in \mathcal{C}_m$ denote a child with $\alpha'_j = \hat{\alpha}_j$ of any cluster C_j. Due to the construction of α_m it is $\alpha'_j \geq \alpha_m$ and (by 1) $\omega_{C_j}(\alpha_m - \varepsilon) < \omega_{C_i}(\alpha_m - \varepsilon)$. For any set C' with $C_i \subset C' \subset C_j$ $\omega_{C'}$ also intersects ω_{C_j} in α'_j but with lower slope. Thus, C_j is the community of $r(C_i)$ in $G_{\alpha_m - \varepsilon}$.

Theorem 1 allows for a simple parametric search starting with the trivial clusterings $\mathcal{C}_0 < \mathcal{C}_{\max}$ ($\alpha_0 > \alpha_{\max}$). In contrast to a binary search on the discretized parameter range this approach definitely returns a complete hierarchy.

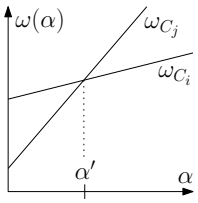


Fig. 2. Intersecting cut-cost functions.

Running time. The parametric search also outperforms the binary search on running times, since it calls CutC at most twice per level in the hierarchy. This yields a running time of $O((h-2)T(n))$ with h the number of levels and $T(n)$ the worst case running time for CutC, compared to $O(h\log(d)T(n))$ of the binary search, where $d \gg n$ is the number of discretization steps. Furthermore, the hierarchical nesting property still allows to contract the clusters on the lower level before applying CutC, and by scheduling the recursive calls carefully such that steps descending into the lower part are executed first we can even reuse the previously contracted structure.

As a proof of concept we conduct a brief experiment on running times of the parametric search and the binary search. The implementation was realized within the LEMON framework [14], version 1.2.1. The max-flow implementation provided by LEMON runs in $O(n^2\sqrt{m})$. For details on instances see Section 4. In order to discretize the continuous parameter range for the binary search, we use between 2^{10} and 2^{30} steps depending on the size of the instances. Comparing the resulting hierarchies to the complete ones confirms this discretization being detailed enough to find all levels for most of the instances. Nevertheless, we do not know how far from optimal our discretization is, although we tried to keep the number of steps low. The difficulty to determine a good discretization is one of the main drawbacks of the binary search approach. Both algorithms run on an AMD Opteron Processor 252 with 2.6 GHz and 16 GB RAM. Table 3 lists ascending CPU times of the parametric search without contraction (PasS).

Table 3. Running times of the parametric search without contraction (ParS) and the binary search with (BinS cont.) and without contraction (BinS). Instances sorted by CPU times of ParS. Times longer than six days are marked by *. See also the full version. [16].

graph	n	m	h	ParS [m:s]	BinS cont. [fac]	BinS [fac]
celegans_metabolic	453	2025	8	0.300	7.620	8.380
celegansneural	297	2148	17	0.406	8.653	9.919
netscience	1589	2742	38	4.310	4.030	11.952
power	4941	6594	66	1:25.736	8.773	15.742
as-22july06	22963	48436	33	39:54.495	12.419	20.583
cond-mat	16726	47594	80	44:15.317	14.917	27.425
rgg_n_2_15	32768	160240	46	245:25.644	32.748	22.573
G_n_pin_pout	100000	501198	4	369:29.033	*	*
cond-mat-2005	40421	175691	82	652:32.163	*	21.446

The factors listed for the binary search with contraction (BinS cont.) and without (BinS) describe how much longer the applications run compared to ParS.

ParS outperforms both binary search approaches by a factor of four up to 32. However, the running time does not only depend on the input size but also on the number of different levels in the hierarchy. This effect can be nicely observed at the two last instances. Although G_n_pin_pout is the biggest graph in this list it takes less time to find four levels therein than constructing 82 levels in the smaller instance cond-mat-2005. The random geometric graph rgg_n_2_15 further demonstrates the impact of constant factors hidden in the asymptotic running time of BinS cont. Asymptotically both binary search approaches are comparable since contraction takes only linear time in terms of m, and is thus dominated by the max-flow computation. In practice BinS cont. performs at least 45 contractions for this instance, each merging only few vertices. Thus, the decreasing size of the graph does not offset the additional costs for contraction.

3.3 Parameter-Free Exclusion Approach

Our parameter-free exclusion approach (ParFree as a shorthand) aims at finding meaningful maximal SCs for most of the vertices in a given instance. If these maximal SCs are still small we conclude that there exists no coarser tight clustering of nice clusters.

ParFree considers all components separately. A maximal SCs is meaningful if it contains at most half of the vertices of the current component. Components that are smaller than half of the largest component in G become trivial clusters with an arbitrary vertex as representative, which is marked *green* in order to illustrate the special type of this cluster. Any other component H is decomposed into SCs by iterating the vertices in H in a non-increasing order by their weighted degrees. Thereby the SCs are marked with the help of further colors indicating the individual properties of the SCs.

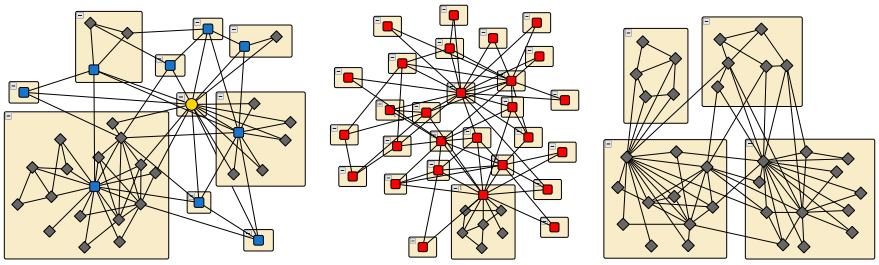
The first vertex designates the source s. The algorithm consecutively computes the community S of s with respect to the next vertex t that is not yet covered by an SC. If $|H \setminus S| \leq |H|/2$ and does not intersect any previously found SC, $H \setminus S$ is the maximal SC of t not containing s (cp. Remark 1). Thus t is marked *blue*. If $|H \setminus S| \leq |H|/2$ but intersects with another SC, $H \setminus S$ is replaced by a non-intersecting SC Q of t according to Gusfield [5] and Gomory and Hu [4] (see Lemma 3 in [16]). Since Q is no longer maximal, t is marked *red*. The representative of any other SC nested in Q becomes

uncolored again. If $|S| < |H|/2$ vertex t becomes the current source and S is an SC of s according to Remark 1; then s is also marked *red*. In the end all vertices are assigned to SCs apart from the source considered last. This source s is marked *orange*.

In a post-processing step the algorithm then searches for a meaningful maximal SC of s consisting of unclustered vertices. If such an SC is found s changes from orange to *yellow*. For a detailed description of the post processing and a proof of the following lemma see the full version [16].

Lemma 2. *Let v denote a blue or uncolored vertex in an SC Q. Then any coarser possibly existing SC Q' of v with $Q \subset Q'$ is also an SC of a red, orange or yellow vertex.*

This is, in any coarser clustering the number of further possibly existing SCs is bounded by the number of red, orange and yellow vertices. On the other hand, these vertices might still induce coarser SCs, which we do not know. Thus we call the number of red, orange and yellow vertices the *uncertainty* of a parfree-clustering. Figure 3(a) shows an example of uncertainty one.



(a) Parfree-clustering. (b) Cut-clustering. (c) Modularity-based clus.

Fig. 3. Different clusterings for karate. The parfree-clustering (a) consists of one yellow vertex (round) besides blue (rectangular) and uncolored ones (diamond shaped): 4 maximal non-singleton SCs of different size, 5 unclustered vertices, uncertainty is one. (b) Cut-clus: one small non-singleton cluster, 27 unclustered vertices. (c) mod-clus: 4 clusters of balanced size.

4 Experiments

For the experimental analysis we used several instances of the clustering testbed of the 10th DIMACS Implementation Challenge [12] as well as the protein interaction network bo_cluster published by Jeong et al. [13] and a snapshot of the linked wiki pages at www.dokuwiki.org (cp. Fig. 4 or Tab. 5 in [16]). Furthermore, we consider a large number of snapshots of the email-communication network of the Department of Informatics at KIT [11] (cp. Tab. 6 to 8 in [16]).

Modularity analysis. Our first experiment addresses the question how close cut-clusterings can get to a modularity-optimal tight clustering with respect to SCs, and which modularity values can be reached in general. Thus, we focus on the best cut-clusterings in the hierarchies with respect to this objective and compare them to reference clusterings of good modularity (mod-clusterings for short) generated with the help of a modularity-based greedy agglomerative approach [2], as computing a modularity-optimal clustering is NP-hard [9]. Figure 4 shows the results.

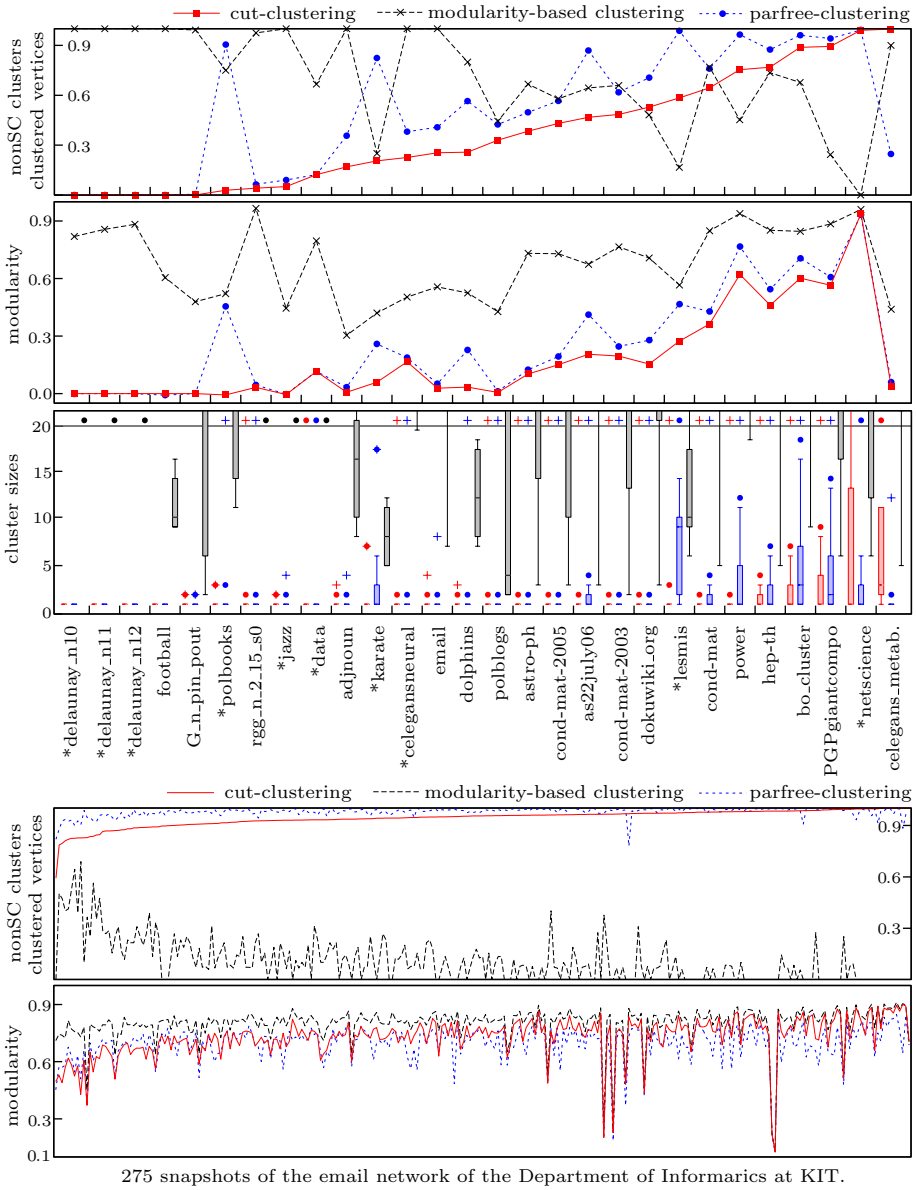


Fig. 4. Results of the modularity analysis of cut-clusterings, modularity-based clusterings and parfree-clusterings. The results for the email snapshots are displayed in the lower part, the upper part addresses the remaining instances. Instances where the uncertainty of the parfree-clustering is at most two are marked by *. In both parts the upper charts show the ratio of clustered vertices in the cut- and parfree-clusterings and the ratio of nontrivial clusters missing the SC-property in the modularity-based clusterings. For the upper instances the cluster sizes are shown by whisker-bars regarding cut-, parfree- and mod-clustering with maximum (+) and minimum (•) of the outliers. Note that values greater than 20 are printed at the edge of the displayed range. Due to the high number of email snapshots whisker-bars are omitted for those instances.

As expected the mod-clusterings are of higher modularity than the cut-clusterings and prefer clusters of decent size. In contrast the cut-clusterings are finer with several unclustered vertices (see also Fig. 3), which is due to the restriction to tight clusterings. Nevertheless, the modularity of the latter increases with the amount of clustered vertices and the size of the clusters.

The fact that for some instances CutC returns clusterings with a modularity much lower than the references, however, does not necessarily mean that the cut-clustering is far from the objective. The instance might just lack a tight clustering of better modularity. In order to find out if this is indeed the case we focus on the properties of the parfree-clustering. As all clusters in this clustering, apart from those counted by the uncertainty, constitute maximal SCs, small clusters and a low uncertainty indicate the absence of a coarser tight clustering, which might provide a higher modularity. In this case, it seems to be more appropriate to compare the modularity of the cut-clustering to the value reached by ParFree. The Delaunay triangulations are nice examples where the modularity gap between the cut-clustering and the mod-clustering is large, but the parfree-clustering consists only of singleton clusters and has a low uncertainty. Thus we can exclude the existence of a better tight clustering with high probability and suppose the cut-clustering, which is basically the same as the parfree-clustering, is close to a modularity-optimal tight clustering, although the reference clustering has a much higher modularity. Note that all clusters in the references of the Delaunay triangulations miss the SC-property. We derive the existence of such degenerated cases from the fact that the asymptotic modularity of some graph classes is provably high [7] whereas the same classes often lack any meaningful SC such that CutC has no chance to return a nontrivial clustering of good modularity. In this light CutC competes surprisingly well.

For the main part of the email snapshots and for the netscience graph CutC reaches modularity values very close to the references, which is rather unexpected since CutC is not designed to optimize modularity. We further observe that the absolute modularity values for these instances are quite high and the amount of clusters in the mod-clusterings that miss the SC-property decreases. The common trend of both modularity curves finally reveals that most instances are either difficult for both clustering approaches or for none of them. We conjecture that if there exists a tight clustering of good modularity, CutC most often finds it.

Expansion analysis. In order to answer the question if there is an advantage from knowing the quality guarantee given by the parameter of CutC, our second experiment compares this guarantee to the trivial lower bound Ψ_ℓ induced by global minimum cuts. We further study the non-trivial lower bound Ψ_a to get an idea on the exact expansion values. Recall Table 2 for an overview of the different intra-cluster expansion bounds. We consider the same cut-clusterings and mod-clusterings as before, however, we skip the Delaunay triangulations and the football graph, as for those instances CutC returns only singleton clusters. The results are given in Figure 5.

We observe that the trivial lower bound Ψ_ℓ in both clustering categories follows a similar trend and stays below the guarantee Ψ_g for most of the instances, which confirms the guarantee as a truly meaningful bound. A value that exceeds Ψ_g appears for example if all the non-singleton clusters are close to cliques of maximum edge costs. This yields a trivial bound close to two times the maximum edge costs, while Ψ_g is

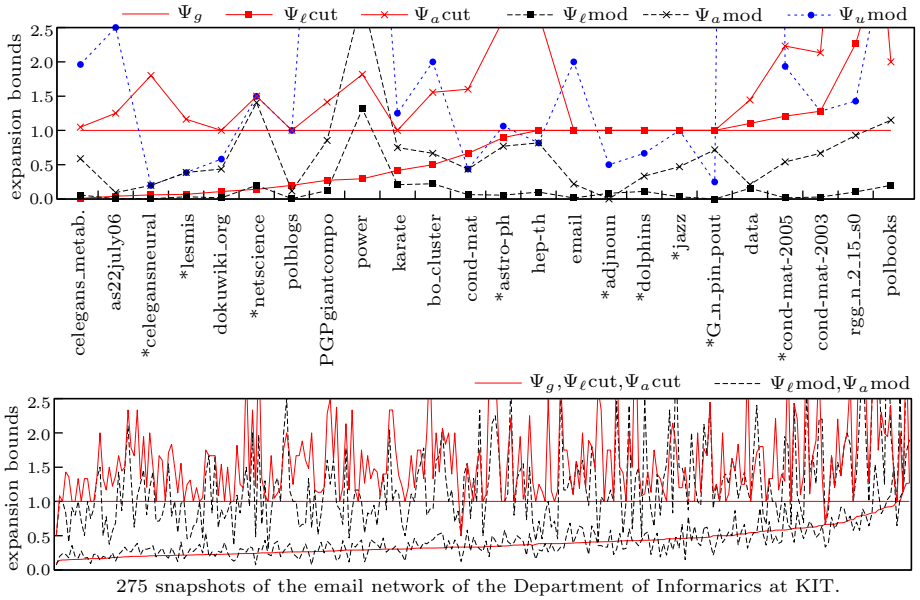


Fig. 5. Trivial and non-trivial bounds on the intra-cluster expansion of cut-clusterings and modularity-based clusterings. For definitions of these bounds recall Table 2. In both charts the guarantee Ψ_g is normalized to one, further values are displayed proportional. Instances where $\Psi_u$cut (not shown) meets the maximum lower bound in the cut-clustering are marked by *. For those instances the maximum lower bound equals the true intra-cluster expansion in the cut-clustering. In the lower chart $\Psi_\ell$cut is shown by the monotone curve different from one, $\Psi_\ell$mod is given by the dashed line close to $\Psi_\ell$cut; $\Psi_a$cut and $\Psi_a$mod are represented by the remaining solid and dashed line. For the sake of readability $\Psi_u$mod is omitted in the lower chart.

bounded by the maximum edge costs. On the other hand, the non-trivial lower bound Ψ_a for the cut-clusterings clearly outperforms the guarantee, and thus, reveals that the actual intra-cluster expansion in the cut-clusterings is even higher than guaranteed. At the same time it also exceeds the analog bound for the mod-clusterings, and hence, suggests that the cut-clusterings also outperform the expansion of the modularity-based reference clusterings. This becomes even a fact whenever the upper bound Ψ_u for the mod-clusterings drops below the maximum lower bound in the cut-clusterings, which indeed happens for some instances. This proves a truly better intra-cluster expansion on the latter. Finally, for some cut-clusterings we yet know the actual intra-cluster expansion, as the analog upper bound Ψ_u meets the lower bounds. The corresponding instances are marked by * in the upper chart of Figure 5, for the email snapshots the amount of those graphs is about 20%.

Conclusion. In this work we studied the behavior of the cut-clustering algorithm of Flake et al. [3] in the light of tight clusterings and the quality measures modularity and expansion. We introduced a characterization of different types of tight clusterings and gave a simple but efficient approach for constructing all different levels of tight clusterings in the cut-clustering hierarchy formed by different parameter values. Our new approach directly computes all breakpoints in the parameter range where new clus-

terings come up, and thus, outperforms binary search approaches in running time and accuracy. Our experiments further exhibited that, although it is not designed to optimize modularity, the cut-clustering algorithm fairly well combines the significance of tight clusterings with a good modularity and a guaranteed intra-cluster expansion that is often better than trivial bounds, provided that there exists a reasonable tight clustering in the given instance.

References

1. G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-Organization and Identification of Web Communities. *IEEE Computer*, 35(3):66–71, 2002.
2. R. Rotta and A. Noack. Multilevel local search algorithms for modularity clustering. *ACM Journal of Experimental Algorithmics*, 16: 2.3:2.1–2.3:2.27, 2011.
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. H. Nagamochi. Graph Algorithms for Network Connectivity Problems. *Journal of the Operations Research Society of Japan*,47(4):199–223, 2004.
7. F. de Montgolfier, M. Soto and L. Viennot, Laurent. Asymptotic Modularity of some Graph Classes. *22nd International Symposium on Algorithms and Computation (ISAAC)*, pages 435-444, December 2011.
8. R. Görke. An Algorithmic Walk from Static to Dynamic Graph Clustering. Doctoral thesis, Department of Informatics, KIT, 2010.
http://digbib.ubka.uni-karlsruhe.de/volltexte/1000018288
9. U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Höfer, Z. Nikoloski and D. Wagner. On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, February 2008.
10. M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(026113):1–16, 2004.
11. Dynamic network of email communication at the Department of Informatics at Karlsruhe Institute of Technology (KIT). Data collected, compiled and provided by Robert Görke and Martin Holzer of ITI Wagner and by Olaf Hopp, Johannes Theuerkorn and Klaus Scheibenberger of ATIS, all at KIT. 2011. i11www.iti.kit.edu/projects/spp1307/emaildata
12. *10th DIMACS Implementation Challenge - Graph Partitioning and Graph Clustering*. http://www.cc.gatech.edu/dimacs10/
13. H. Jeong, S. Mason, A. L. Barabási and Z. N. Oltvai. *Centrality and lethality of protein networks*. Nature, 411(11):41, 2001
14. *LEMON graph library*. Library for Efficient Modeling and Optimization in Networks. http://lemon.cs.elte.hu/trac/lemon
15. D. Lisowski. Modularity-basiertes Clustern von dynamischen Graphen im Offline-Fall. Mastersthesis, Department of Informatics, KIT, 2011.
http://i11www.iti.uni-karlsruhe.de/_media/teaching/theses/da-lisowski.pdf
16. M. Hamann, T. Hartmann and D. Wagner. Complete Hierarchical Cut-Clustering: A Case Study on Modularity and Expansion. Karlsruhe Reports in Informatics 2012-4, KIT, 2012.