8 Clustering

Marco Gaertler

Clustering is a synonym for the decomposition of a set of entities into 'natural groups'. There are two major aspects to this task: the first involves algorithmic issues on how to find such decompositions, i.e., tractability, while the second concerns quality assignment, i.e., how good is the computed decomposition. Owing to the informal notion of natural groups, many different disciplines have developed their view of clustering independently. Originally, clustering was introduced to the data mining research as the unsupervised classification of patterns into groups [324]. Since that time a comprehensive framework has started to evolve. In the following, the simple yet fundamental paradigm of intra-cluster density versus inter-cluster sparsity will be discussed exclusively. This restriction is necessary in order to provide some insight into clustering theory, and to keep the scope of this chapter. However, many other interpretations of natural decomposition extend this framework, or are relatively similar. Another specialty is that the input data is represented as networks that are not complete in general. In the classic clustering theory, entities were embedded in metric spaces, and the distance between them was related to their similarity. Thus, all pairwise similarities were known at the beginning. In standard network analysis, the input networks are usually sparse. Even if they are not, it is very unlikely that they are complete. This will be the motivation for studying clustering methods that deal with network input data.

Clustering, based either on the simple paradigm of intra-cluster density versus inter-cluster sparsity or on other more sophisticated formulations, focuses on disjoint cliques as the ideal situation. In some instances the desired cases are totally different. Models where clusters and their connection between each other can have more complex structural properties are targeted in the following chapters about roles (Chapter 9) and blockmodels (Chapter 10).

The popularity of density-based clustering is due to its similarity to the human perception. Most things in our daily life are naturally grouped into categories. For example books are classified with respect to their content, e.g., scientific, fiction, guidebooks, law, etc. Each topic can be refined, e.g., scientific publications can be grouped according to their scientific discipline. The relationship of elements within the same group is strong, whereas elements in different groups typically have a weak relation. Most approaches that deal with information processing are based upon this fact. For example finding a book with a certain content. First, related topics are selected and only those books that

U. Brandes and T. Erlebach (Eds.): Network Analysis, LNCS 3418, pp. 178–215, 2005.

belong to these groups are examined closer. The recursive structure of topics and subtopics suggests a repeated application of this technique. Using the clustering information on the data set, one can design methods that explore and navigate within the data with a minimum of human interaction. Therefore, it is a fundamental aspect of automatic information processing.

Preliminaries

Let $G = (V, E)$ be a directed graph. A *clustering* $\mathcal{C} = \{C_1, \ldots, C_k\}$ of G is a partition of the node set V into non-empty subsets C_i. The set $E(C_i, C_j)$ is the set of all edges that have their origin in C_i and their destination in C_j; $E(C_i)$ is a short-hand for $E(C_i, C_i)$. Then $E(\mathcal{C}) := \bigcup_{i=1}^{k} E(C_i)$ is the set of *intra-cluster edges* and $\overline{E(\mathcal{C})} := E \setminus E(\mathcal{C})$ the set of *inter-cluster edges*. The number of intra-cluster edges is denoted by $m(\mathcal{C})$ and the number of inter-cluster edges by $\overline{m}(\mathcal{C})$. In the following, we often identify a cluster C_i with the induced subgraph of G, i.e., the graph $G[C_i] := (C_i, E(C_i))$. A clustering is called *trivial* if either $k = 1$ (*1-clustering*) or $k = n$ (*singletons*). A clustering with $k = 2$ is also called a cut (see also Section 2.2.3).

The set of all possible clusterings is denoted by $\mathcal{A}(G)$. The set $\mathcal{A}(G)$ is partially ordered with respect to inclusion. Given two clusterings $\mathcal{C}_1 := \{C_1, \ldots, C_k\}$ and $\mathcal{C}_2 := \{C'_1, \ldots, C'_\ell\}$, Equation (8.1) shows the definition of the partial ordering.

$$\mathcal{C}_1 \leq \mathcal{C}_2 \colon \iff \forall\, 1 \leq i \leq k \colon \exists\, j \in \{1, \ldots, \ell\} \colon C_i \subseteq C'_j \tag{8.1}$$

Clustering $\mathcal{C}_1$ is called a *refinement* of $\mathcal{C}_2$, and $\mathcal{C}_2$ is called a *coarsening* of $\mathcal{C}_1$. A chain of clusterings, i.e., a subset of clusterings such that every pair is comparable, is also called a *hierarchy*. The hierarchy is called *total* if both trivial clusterings are contained. A hierarchy that contains exactly one clustering of k clusters for every $k \in \{1, \ldots, n\}$ is called *complete*. It is easy to see that such a hierarchy has n clusterings and that no two of these clusterings have the same number of clusters.

Besides viewing a clustering as a partition, it can also be seen as an equivalence relation $\sim_{\mathcal{C}}$ on $V \times V$, where $u \sim_{\mathcal{C}} v$ if u and v belong to the same cluster in $\mathcal{C}$. Note that the edge set E is also a relation over $V \times V$, and it is an equivalence relation if and only if the graph consists of the union of disjoint cliques.

The power set of a set X is the set of all possible subsets, and is denoted by $\mathcal{P}(X)$, see also Section 2.4. A *cut function* $S \colon \mathcal{P}(V) \to \mathcal{P}(V)$ maps a set of nodes to a subset of itself, i.e.,

$$\forall\, V' \subseteq V \colon S(V') \subseteq V' \ . \tag{8.2}$$

Cut functions formalize the idea of cutting a node-induced subgraph into two parts. For a given node subset V' of V the cut function S defines a cut by $(S(V'), V' \setminus S(V'))$. In order to exclude trivial functions, we require a cut function to assign a non-empty proper subset whenever possible. *Proper* cut functions in addition fulfill the condition (8.3).

$$\forall\, V' \subseteq V \colon |V'| > 1 \Longrightarrow \emptyset \neq S(V') \subset V' \ . \tag{8.3}$$

These functions are important for clustering techniques that are based on recursive cutting. These methods will be introduced in Section 8.2.1, and some examples are given in Section 8.2.2.

Graph model of this chapter. In this chapter, *graph* usually means simple and directed graphs with edge weights and without loops.

Content Organization

The following is organized into three parts. The first one introduces measurements for the quality of clusterings. They will provide a formal method to define 'good' clusterings. This will be important due to the informal foundation of clustering. More precisely, these structural indices rate partitions with respect to different interpretations of natural groups. Also, they provide the means to compare different clusterings with respect to their quality. In the second part, generic concepts and algorithms that calculate clusterings are presented. The focus is directed on the fundamental ideas, and not on concrete parameter choices. Finally, the last section discusses potential enhancements. These extensions are limited to alternative models for clusterings, some practical aspects, and Kleinberg's proposal of an axiomatic system for clustering.

8.1 Quality Measurements for Clusterings

As was pointed out in the introduction, clustering techniques are used to find groups that are internally dense and that are only sparsely connected with each other. Although this paradigm of intra-cluster density versus inter-cluster sparsity is more precise than the term 'natural groups', it is still based on our intuition and not on a formal quantification. One way to mathematically express it is by structural indices. These are mappings that assign a non-negative real number to each clustering. Often their range is normalized to the unit interval, where one means best possible structural behavior and zero means worst possible structural behavior. In the following a general framework for indices is presented. Most of the measurements can be expressed within it.

Let $G = (V, E, \omega)$ be a simple, weighted and directed graph, where $\omega \colon E \to \mathbb{R}_0^+$ represents the strength of the similarity relation modeled by the edges, and let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of G. It is indeed possible to have negative values as strength of similarity, which would express the dissimilarity of two nodes. However, it is rarely the case that similarity and dissimilarity are expressed within the same relation. A far more common case is to have two (weighted) relations, one for similarity and one for dissimilarity. In the following, we will focus on only one relation that expresses similarity. For the unweighted case, the weighting function ω is assumed to be constantly one. In many cases ω will be a mapping to $\mathbb{R}^+$, however, in some cases it is useful to distinguish between edges with weight zero and those node pairs that are not connected by an

edge. We will also use the following short-cut for summing up the weight of an edge subset:

$$\omega(E') := \sum_{e \in E'} \omega(e) \qquad \text{for } E' \subseteq E \ .$$

For simplicity, we assume that $\omega(E) \neq 0$.

Before defining the actual indices, their framework is presented. The indices will be composed of two independent functions $f, g \colon \mathcal{A}(G) \to \mathbb{R}_0^+$, where f measures the density inside the clusters and g the sparsity between clusters. The functions are combined in the following way:

$$\mathsf{index}(\mathcal{C}) := \frac{f(\mathcal{C}) + g(\mathcal{C})}{\max\{f(\mathcal{C}') + g(\mathcal{C}') : \mathcal{C}' \in \mathcal{A}(G)\}} \tag{8.4}$$

In order to guarantee the well-definition of Equation (8.4), we assume that there is at least one clustering $\mathcal{C}'$ such that $f(\mathcal{C}') + g(\mathcal{C}')$ is not zero. For some indices either f or g is constantly zero. These indices examine only the (internal) density or the (external) sparsity.

Indices serve two different purposes simultaneously: first and foremost, they rate a partition with respect to clustering paradigms, and, second, they compare clusterings regarding quality. Before we explain both aspects in detail, please note that, while our indices have these properties, there exist other measures that realize only one aspect.

The quality of a partition as a clustering is expressed in quantitative terms, i.e., an index (discretely) counts certain substructures like intra-cluster edges, triangles, or cliques. These structural elements are related to clustering properties. Many intra-cluster edges, triangles, or cliques inside the clusters indicate that the cluster is dense. In an analogous way the lack of these elements between clusters imply the inter-cluster sparsity. Thus, the quality of clustering is reduced to a quantitative aspect. Intuitively, there will always be a maximum number of these indicators. The number of intra-cluster edges, triangles, or cliques is limited by the size of clusters. In the ideal case the bounds are met. Thus, if an index counts only half of them, then the clustering is only half as good as possible. As we will see, these bounds are not tight for all indices and all input graphs. Thus, most of them are 'absolute' in the sense that they do not depend on the input graph, but rather on the clustering paradigms. In conclusion, the actual range of indices provides useful information.

Because of this absolute, quantitative structure of our indices, they can be used to compare clusterings regardless of whether the underlying graph is the same or not. Different graphs can have different bounds for the number of substructures, but the indices rate the quality relative to the individual bounds. For example, in a graph with 10 nodes and 15 edges a clustering could have 12 intra-cluster edges, and in another graph with 18 nodes and 30 edges another clustering could have 27 inter-cluster edges. Although we have three inter-cluster edges in both cases, the second clustering would be better since $27/30 = 9/10 > 4/5 = 12/15$. This property is required when algorithms

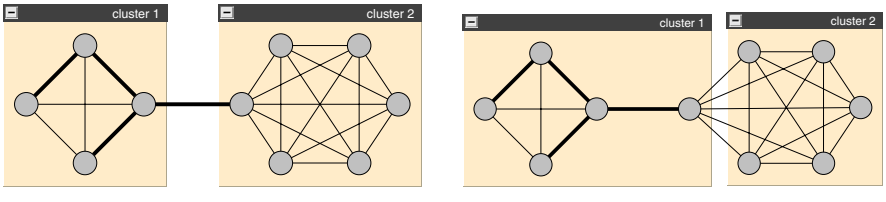
are evaluated with random instances, or the data of the input network is not reliable, i.e., different data collections result into different networks. The clustering methods could be applied to all networks, and the clustering with the best score would be chosen. However, if the index uses characteristics of the input graph, like the number of edges, maximum weight, etc., then this comparison can only be done when the underlying graph is the same for all clusterings. Therefore, indices that depend on the input graph are not appropriate for all applications, such as benchmarks. Although this dependency will seldom occur, one has to consider these facts when designing new indices.

8.1.1 Coverage

The *coverage* $\gamma(\mathcal{C})$ measures the weight of intra-cluster edges, compared to the weight of all edges. Thus $f(\mathcal{C}) = \omega(E(\mathcal{C}))$ and $g \equiv 0$. The maximum value is achieved for $\mathcal{C} = \{V\}$. Equation (8.5) shows the complete formula.

$$\gamma(\mathcal{C}) := \frac{\omega(E(\mathcal{C}))}{\omega(E)} = \frac{\sum_{e \in E(\mathcal{C})} \omega(e)}{\sum_{e \in E} \omega(e)} \tag{8.5}$$

Coverage measures only the accumulated density within the clusters. Therefore, an individual cluster can be sparse or the number of inter-cluster edges can be large. This is illustrated in Figure 8.1. Coverage is also the probability of randomly selecting an intra-cluster edge (where the probability of selection an edge is proportional to its weight, i.e., $\Pr[e] \sim \omega(e)$). The structure of clusterings with optimal coverage value is related to the connectivity structure of the graph. A clustering is *compatible* with the connectivity structure if clusters consist only of unions of connected components of the graph. Proposition 8.1.1 gives a characterization of clusterings with optimal coverage value.



(a) intuitive clustering

(b) non-trivial clustering with best coverage

Fig. 8.1. A situation where coverage splits an intuitive cluster. The thickness of an edge corresponds to its weight. If normal edges have weight one and bold edges weight 100, then the intuitive clustering has $\gamma = 159/209 \approx 0.76$ while the optimal value for coverage is $413/418 \approx 0.99$

Proposition 8.1.1. *A clustering has $\gamma = 1$ if and only if either the set of inter-cluster edges is empty or all inter-cluster edges have weight zero. Especially, clusterings that are compatible with the connectivity structure have coverage value 1.*

Sketch of Proof. The edge set is disjointly partitioned into intra-cluster edges and inter-cluster edges, therefore Equation (8.6) holds for any clustering $\mathcal{C}$.

$$\omega(E) = \omega(E(\mathcal{C})) + \omega(\overline{E(\mathcal{C})}) \tag{8.6}$$

Thus, coverage $\gamma(\mathcal{C}) = 1$ holds if and only if $\omega(\overline{E(\mathcal{C})}) = 0$.

Because clusterings that are compatible with the connectivity structure of the graph have no inter-cluster edge, one can use the equivalence to prove the proposition. □

A conclusion of Proposition 8.1.1 is that the 1-clustering always has a coverage value 1. The case of disconnected input graphs is rather special and most techniques even assume that it is connected. Proposition 8.1.1 can be extended to characterize non-trivial clusterings that are optimal with respect to coverage.

A further characterization of non-trivial clusterings that are optimal when trivial clusterings are excluded is possible and given in Proposition 8.1.2.

Proposition 8.1.2. *Let $G = (V, E)$ be a connected graph where every cut has positive weight. Then the clusterings that have more than one cluster and have optimal coverage value are those that are induced by a minimum cut.*

Proof. First, it is obvious that every clustering $\mathcal{C}$ with $k > 1$ can be transformed into a clustering $\mathcal{C}'$ with less than k clusters, with $\gamma(\mathcal{C}) \leq \gamma(\mathcal{C}')$. This is achieved by merging any two clusters. Therefore, a non-trivial clustering with optimal coverage has two clusters, and thus is a cut. Second, maximizing coverage is equivalent to minimizing the weight of the inter-cluster edges, and the edges that are contained in the cut have minimum weight. That completes the proof. □

Because of the properties described in Propositions 8.1.1 and 8.1.2, coverage is rarely used as the only quality measurement of a clustering. Minimum cuts often cannot catch the intuition, and separate only a very small portion of the graph. However, there are a few exceptions [286], where both, the input graph and good clusterings, have a very special structure. An extended version will be used for the next index (Section 8.1.2).

8.1.2 Conductance

In contrast to coverage, which measures only the accumulated edge weight within clusters, one can consider further structural properties like connectivity. Intuitively, a cluster should be well connected, i.e., many edges need to be removed to bisect it. Two clusters should also have a small degree of connectivity between each other. In the ideal case, they are already disconnected. Cuts are a useful

method to measure connectivity (see also Chapter 7). The standard minimum cut has certain disadvantages (Proposition 8.1.2), therefore an alternative cut measure will be considered: *conductance*. It compares the weight of the cut with the edge weight in either of the two induced subgraphs. Informally speaking, the conductance is a measure for bottlenecks. A cut is a bottleneck, if it separates two parts of roughly the same size with relatively few edges.

Definition 8.1.3. *Let $\mathcal{C}' = (C_1', C_2')$ be a cut, i.e., $(C_2' = V \setminus C_1')$ then the* conductance-weight $a(C_1')$ *of a cut side and the* conductance $\varphi(\mathcal{C}')$ *are defined in Equations* (8.7) *and* (8.8).

$$a(C_1') := \sum_{(u,v) \in E(C_1', V)} \omega((u,v)) \tag{8.7}$$

$$\varphi(\mathcal{C}') := \begin{cases} 1, & \text{if } C_1' \in \{\emptyset, V\} \\ 0, & \text{if } C_1' \notin \{\emptyset, V\}, \omega(\overline{E(\mathcal{C})}) = 0 \\ \dfrac{\omega(\overline{E(\mathcal{C})})}{\min(a(C_1'), a(C_2'))}, & \text{otherwise} \end{cases} \tag{8.8}$$

The conductance *of the graph G is defined by*

$$\varphi(G) = \min_{C_1 \subseteq V} \varphi((C_1, V \setminus C_1)) \ . \tag{8.9}$$

Note that the case differentiation in Equation (8.8) is only necessary in order to prevent divisions by zero. Before presenting further general information about conductance, graphs with maximum conductance are characterized.

Lemma 8.1.4. *Let $G = (V, E, \omega)$ be an undirected and positively weighted graph. Then G has maximum conductance, i.e., $\varphi(G) = 1$ if and only if G is connected and has at most three nodes, or is a star.*

Proof. Before the equivalence is shown, two short observations are stated:

1. All disconnected graphs have conductance 0 because there is a non-trivial cut that has zero weight and the second condition of the Formula (8.8) holds.
2. For a non-trivial cut $\mathcal{C}' = (C_1', V \setminus C_1')$ the conductance-weight $a(C_1')$ can be rewritten as

$$a(C_1') = \sum_{e \in E(C_1', V)} \omega(e) = \omega(E(C_1')) + \omega(\overline{E(\mathcal{C})})$$

in undirected graphs. Thus, the third condition in Formula (8.8) can be simplified to

$$\frac{\omega(\overline{E(\mathcal{C})})}{\min\left(a(C_1'), a(V \setminus C_1')\right)} = \frac{\omega(\overline{E(\mathcal{C})})}{\omega(\overline{E(\mathcal{C})}) + \min\left(\omega(E(C_1')), \omega(E(V \setminus C_1'))\right)} \ . \tag{8.10}$$

'$\Longleftarrow$': If G has one node, then the first condition of Formula (8.8) holds and thus $\varphi(G) = 1$.
If G has two or three nodes or is a star, then every non-trivial cut $\mathcal{C}' = (C'_1, V \setminus C'_1)$ isolates an independent set, i.e., $E(C'_1) = \emptyset$. This is achieved by setting C'_1 to the smaller cut set if G has at most three nodes and to the cut set that does not contain the center node if G is a star. Therefore $\omega(E(C'_1)) = 0$ and Equation (8.10) implies $\varphi(\mathcal{C}') = 1$. Because all non-trivial cuts have conductance 1, the graph G has conductance 1 as well.

'$\Longrightarrow$': If G has conductance one, then G is connected (observation 1) and for every non-trivial cut $\mathcal{C}' = (C'_1, V \setminus C'_1)$ at least one edge set $E(C'_1)$ or $E(V \setminus C'_1)$ has 0 weight (observation 2). Because ω has only positive weight, at least one of these sets has to be empty.
It is obvious that connected graphs with at most three nodes fulfill these requirements, therefore assume that G has at least four nodes. The graph has a diameter of at most two because otherwise there is a path of length three with four pairwise distinct nodes $v_1, \ldots, v_4$, where $e_i := \{v_i, v_{i+1}\} \in E$ for $1 \leq i \leq 3$. Then the non-trivial cut $\mathcal{C}' = (\{v_1, v_2\}, V \setminus \{v_1, v_2\})$ cannot have conductance 1, because first the inequality $\omega(E(\mathcal{C}')) \geq \omega(e_2) \geq 0$ implies the third condition of Formula (8.8) and second both cut sides are non-empty ($e_1 \in E(\{v_1, v_2\})$ and $e_3 \in E(V \setminus \{v_1, v_2\})$). By the same argument, G cannot contain a simple cycle of length four or greater. It also cannot have a simple cycle of length three. Assume G has such a cycle v_1, v_2, v_3. Then there is another node v_4 that is not contained in the cycle but in the neighborhood of at least one v_i. Without loss of generality $i = 1$. Thus, the non-trivial cut $(\{v_1, v_4\}, V \setminus \{v_1, v_4\})$ is a counterexample. Thus G cannot contain any cycle and is therefore a tree. The only trees with at least four nodes, and a diameter of at most two, are stars.

□

It is $\mathcal{NP}$-hard to calculate the conductance of a graph [39]. Fortunately, it can be approximated with a guarantee of $\mathcal{O}(\log n)$ [565] and $\mathcal{O}(\sqrt{\log n})$ [36]. For some special graph classes, these algorithms have constant approximation factors. Several of the involved ideas are found in the theory of Markov chains and random walks. There, conductance models the probability that a random walk gets 'stuck' inside a non-empty part. It is also used to estimate bounds on the rate of convergence. This notion of 'getting stuck' is an alternative description of bottlenecks. One of these approximation ideas is related to spectral properties. Lemma 8.1.5 indicates the use of eigenvalues as bounds.

Lemma 8.1.5 ([521, Lemma 2.6]). *For an ergodic*[1] *reversible Markov chain with underlying graph G, the second (largest) eigenvalue λ_2 of the transition matrix satisfies:*

[1] Aperiodic and every state can be reached an arbitrary number of times from all initial states.

$$\lambda_2 \geq 1 - 2 \cdot \varphi(G) \ . \tag{8.11}$$

A proof of that can be found in [521, p. 53]. Conductance is also related to isoperimetric problems as well as expanders, which are both related to similar spectral properties themselves. Section 14.4 states some of these characteristics.

For unweighted graphs the conductance of the complete graph is often a useful boundary. It is possible to calculate its exact conductance value. Proposition 8.1.6 states the result. Although the formula is different for even and odd number of nodes, it shows that the conductance of complete graphs is asymptotically 1/2.

Proposition 8.1.6. *Let n be an integer, then equation* (8.12) *holds.*

$$\varphi(K_n) = \begin{cases} \frac{1}{2} \cdot \frac{n}{n-1} & , \textit{ if } n \textit{ is even} \\ \frac{1}{2} + \frac{1}{n-1} & , \textit{ if } n \textit{ is odd} \end{cases} \tag{8.12}$$

Proof. Evaluating Equation (8.8) in Definition 8.1.3 with $G = K_n$ leads to

$$\varphi(K_n) = \min_{C \subset V, 1 \leq |C| < n} \frac{|C| \cdot (n - |C|)}{\min(|C|(|C| - 1), (n - |C|)(n - |C| - 1))} \ . \tag{8.13}$$

Node subsets of size k of a complete graph are pairwise isomorphic, therefore only the size of the subset C matters. Thus, Equation (8.13) can be simplified to

$$\varphi(K_n) = \min_{1 \leq k < n} \frac{k(n-k)}{\min(k^2 - k, n^2 - 2nk - n - k)} \ . \tag{8.14}$$

The fraction in Equation (8.14) is symmetric, thus it is sufficient if k varies in the range from 1 to $\lceil n/2 \rceil$. Using the fact that the fraction is also monotonic decreasing with increasing k, the minimum is assumed for $k = \lfloor n/2 \rfloor$. A simple case differentiation for even and odd ks leads to the final Equation (8.12). □

In the following, two clustering indices are derived with the help of conductance. These will be intra-cluster conductance and inter-cluster conductance, and both will focus on only one property. The first one measures internal density, while the second rates the connection between clusters.

The intra-cluster conductance α is defined as the minimum conductance occurring in the cluster-induced subgraphs $G[C_i]$, i.e.,

$$f(\mathcal{C}) = \min_{1 \leq i \leq k} \varphi(G[C_i]) \qquad \text{and} \qquad g \equiv 0 \ . \tag{8.15}$$

Note that $G[C_i]$ in $\varphi(G[C_i])$ denotes a subgraph, and therefore is independent of the rest of the original graph G. The conductance of a (sub)graph is small if it can naturally be bisected, and great otherwise. Thus, in a clustering with small intra-cluster conductance there is supposed to be at least one cluster containing a bottleneck, i.e., the clustering is possibly too coarse in this case. The minimum conductance cut itself can also be used as a guideline to split the cluster further (refer to Section 8.2.2 for further details).

The inter-cluster conductance δ considers the cuts induced by clusters, i.e.,

$$f \equiv 0 \qquad \text{and} \qquad g = \begin{cases} 1, & \text{if } \mathcal{C} = \{V\} \\ 1 - \max\limits_{1 \le i \le k} \varphi\left((C_i, V \setminus C_i)\right), & \text{otherwise} \end{cases}. \tag{8.16}$$

Note that $(C_i, V \setminus C_i)$ in $\varphi\left((C_i, V \setminus C_i)\right)$ denotes a cut within the graph G. A clustering with small inter-cluster conductance is supposed to contain at least one cluster that has relatively strong connections outside, i.e., the clustering is possibly too fine. In contrast to the intra-cluster conductance, one cannot directly use the induced cut information to merge two clusters.

For both indices the maximum of $f+g$ is one, which leads to the final formula:

$$\alpha\left(\mathcal{C}\right) := \min_{1 \le i \le k} \varphi\left(G[C_i]\right) \tag{8.17}$$

$$\delta\left(\mathcal{C}\right) := \begin{cases} 1, & \text{if } \mathcal{C} = \{V\} \\ 1 - \max\limits_{1 \le i \le k} \varphi\left((C_i, V \setminus C_i)\right), & \text{otherwise} \end{cases} \tag{8.18}$$

Again a characterization of optimal clusterings is possible.

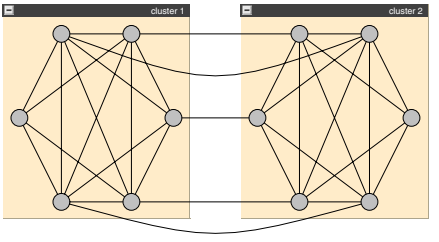
Proposition 8.1.7. *Only clusterings where the clusters consist of connected subgraphs that are stars or have size of at most three, have intra-cluster conductance* 1*.*

Proposition 8.1.8. *Only clusterings that have an inter-cluster edge weight of zero, including the* 1*-clustering, have inter-cluster conductance* 1*.*

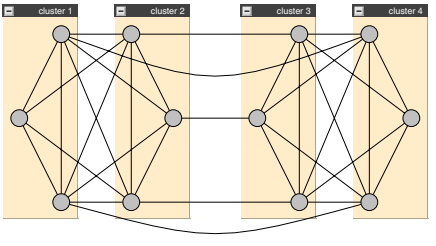
Both Proposition 8.1.7 and 8.1.8 are immediate consequences of the Definition 8.1.3 and Lemma 8.1.4. Both measures, intra-cluster conductance and inter-cluster conductance, have certain disadvantages. Two examples are shown in Figures 8.2 and 8.3. Both examples explore the 'artificial' handling of small graphs considering their conductance. In practical instances, intra-cluster conductance values are usually below 1/2. Small clusters with few connections have relatively small inter-cluster conductance values.

8.1.3 Performance

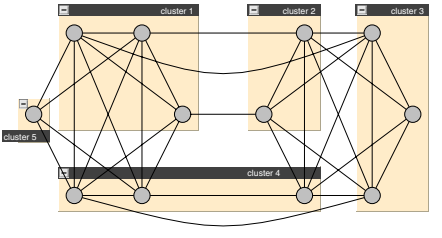
The next index combines two non-trivial functions for the density measure f and the sparsity measure g. It simply counts certain node pairs. According to the general intuition of intra-cluster density versus inter-cluster sparsity, we define for a given clustering a *'correct' classified* pair of nodes as two nodes either belonging to the same cluster and connected by an edge, or belonging to different clusters and not connected by an edge. The resulting index is called *performance*. Its density function f counts the number of edges within all clusters while its sparsity function g counts the number of nonexistent edges between clusters, i.e.,



(a) intuitive clustering



(b) non-trivial clustering with best intra-cluster conductance

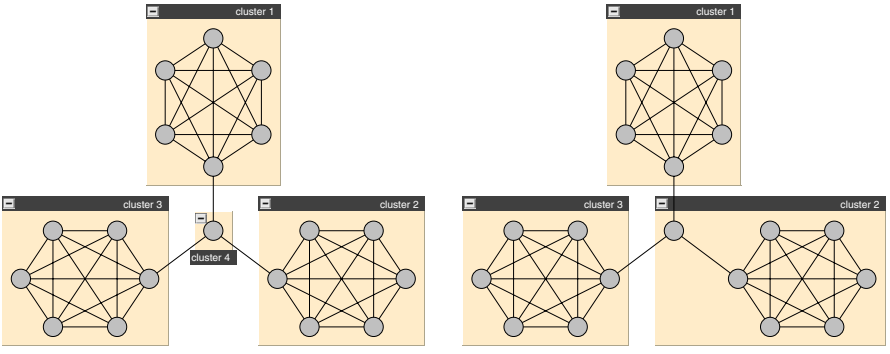


(c) non-trivial clustering with best intra-cluster conductance

Fig. 8.2. A situation where intra-cluster conductance splits intuitive clusters. The intuitive clustering has $\alpha = 3/4$, while the other two clusterings have $\alpha = 1$. The split in Figure 8.2(b) is only a refinement of the intuitive clustering, while Figure 8.2(c) shows a clusterings with same intra-cluster conductance value that is skew to the intuitive clustering

$$\begin{aligned} f(\mathcal{C}) &:= \sum_{i=1}^{k} |E(C_i)| \quad \text{and} \\ g(\mathcal{C}) &:= \sum_{u,v \in V} [(u,v) \notin E] \cdot [u \in C_i, v \in C_j, i \neq j] \ . \end{aligned} \tag{8.19}$$

The definition is given in Iverson Notation, first described in [322], and adapted by Knuth in [365]. The term inside the parentheses can be any logical statement. If the statement is true the term evaluates to 1, if it is false the term is 0. The maximum of $f+g$ has $n \cdot (n-1)$ as upper bound because there are $n(n-1)$ different node pairs. Please recall that loops are not present and each pair contributes with either zero or one. Calculating the maximum of $f + g$ is $\mathcal{NP}$-hard (see [516]), therefore this bound is used instead of the real maximum. By using some duality aspects, such as the number of intra-cluster edges and the number of inter-cluster edges sum up to the whole number of edges, the formula of performance can be simplified as shown in Equation (8.21).



(a) intuitive clustering

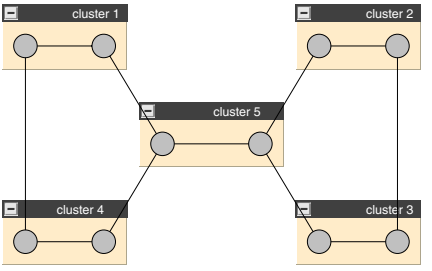
(b) non-trivial clustering with best inter-cluster conductance

Fig. 8.3. A situation where two very similar clusterings have very different inter-cluster conductance values. The intuitive clustering has $\delta = 0$, while the other has the optimum value of 8/9

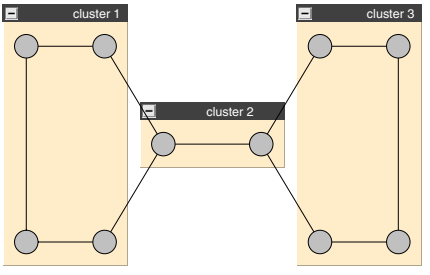
$$\begin{aligned}
\mathsf{perf}\,(\mathcal{C}) &= \frac{m\,(\mathcal{C}) + \left(n(n-1) - \sum_{i=1}^{k} |C_i|(|C_i|-1) - \overline{m}\,(\mathcal{C})\right)}{n(n-1)} \\
&= \frac{n(n-1) - m + 2m\,(\mathcal{C}) - \sum_{i=1}^{k} |C_i|(|C_i|-1)}{n(n-1)} \qquad (8.20) \\
&= 1 - \frac{m(1 - 2\frac{m(\mathcal{C})}{m}) + \sum_{i=1}^{k} |C_i|(|C_i|-1)}{n(n-1)}\ . \qquad (8.21)
\end{aligned}$$

Note that the derivation from Equation (8.20) to (8.21) applies the equality $m = m\,(\mathcal{C}) + \overline{m}\,(\mathcal{C})$, and that $m\,(\mathcal{C})\,/m$ is just the coverage $\gamma\,(\mathcal{C})$ in the unweighted case. Similarly to the other indices, performance has some disadvantages. Its main drawback is the handling of very sparse graphs. Graphs of this type do not contain subgraphs of arbitrary size and density. For such instances the gap between the number of feasible edges (with respect to the structure) and the maximum number of edges (regardless of the structure) is also huge. For example, a planar graph cannot contain any complete graph with five or more nodes, and the maximum number of edges such that the graph is planar is linear in the number of nodes, while in general it is quadratic. In conclusion, clusterings with good performance tend to have many small clusters. Such an example is given in Figure 8.4.

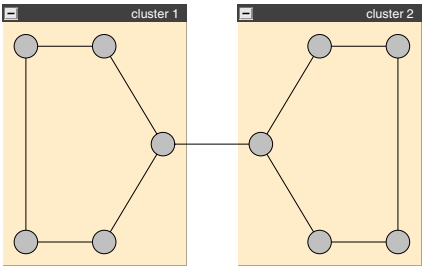
An alternative motivation for performance is given in the following. Therefore recall that the edge set induces a relation on $V \times V$ by $u \sim v$ if $(u, v) \in E$, and clusterings are just another notion for equivalence relations. The problem of finding a clustering can be formalized in this context as finding a transformation of the edge-induced relation into an equivalence relation with small cost. In



(a) clustering with best performance



(b) intuitive clustering



(c) another intuitive clustering

Fig. 8.4. A situation where the clustering with optimal performance is a refinement (Figure 8.4(b)) of an intuitive clustering and is skew (Figure 8.4(c)) to another intuitive clustering

other words, add or delete edges such that the relation induced by the new edge set is an equivalence relation. As cost function, one simply counts the number of additional and deleted edges. Instead of minimizing the number of changes, one can consider the dual version: find a clustering such that the clustering-induced relation and the edge-set relation have the greatest 'intersection'. This is just the maximizing $f + g$. Thus performance is related to the 'distance' of the edge-set relation to the closed clustering. Because finding this maximum is $\mathcal{NP}$-hard, solving this variant of clustering is also $\mathcal{NP}$-hard. Although the problem is hard, it possesses a very simple integer linear program (ILP). ILPs are also $\mathcal{NP}$-hard, however there exist many techniques that lead to usable heuristics or approximations for the problems. The ILP is given by n^2 decision variables $X_{uv} \in \{0, 1\}$ with $u, v \in V$, and the following three groups of constraints:

$$\begin{aligned}
&\text{reflexivity} && \forall\, u\colon X_{uu} = 1\\
&\text{symmetry} && \forall\, u, v\colon X_{uv} = X_{vu}\\
&\text{transitivity} && \forall\, u, v, w\colon \begin{cases} X_{uv} + X_{vw} - 2 \cdot X_{uw} \leq 1\\ X_{uw} + X_{uv} - 2 \cdot X_{vw} \leq 1\\ X_{vw} + X_{uw} - 2 \cdot X_{uv} \leq 1 \end{cases}
\end{aligned}$$

and minimizing objective function:

$$\sum_{(u,v)\in V^2} (1 - E_{uv})X_{uv} + E_{uv}(1 - X_{uv}) \ ,$$

$$\text{with} \ \ E_{uv} = \begin{cases} 1 & \text{, if } (u,v) \in E \text{ or } u = v\\ 0 & \text{, otherwise} \end{cases} \ .$$

The idea is that the X variables represent equivalence relations, i.e., two nodes $u, v \in V$ are equivalent if $X_{uv} = 1$ and the objective function counts the number of not 'correct' classified node pairs.

There exist miscellaneous variations of performance that use more complex models for classification. However, many modifications highly depend on their applicational background. Instead of presenting them, some variations to include edge weights are given. As pointed out in Section 8.1, indices serve two different tasks. In order to preserve the comparability aspect, we assume that all the considered edge weights have a meaningful maximum M. It is not sufficient to replace M with the maximum occurring edge weight because this value depends on the input graph. Also, choosing an extremely large value of M is not suitable because it disrupts the range aspects of the index. An example of such weightings with a meaningful maximum are probabilities where $M = 1$. The weighting represents the probability that an edge can be observed in a random draw. Using the same counting scheme of performance, one has to solve the problem of assigning a real value for node pairs that are not connected. This problem will be overcome with the help of the meaningful maximum M.

The first variation is straightforward and leads to the measure functions given in Equation (8.22):

$$\begin{aligned}
f(\mathcal{C}) &:= \sum_{i=1}^{k} \omega\left(E(C_i)\right) \quad \text{and}\\
g(\mathcal{C}) &:= \sum_{u,v\in V} M \cdot [(u,v) \notin E] \cdot [u \in C_i, v \in C_j, i \neq j] \ .
\end{aligned} \tag{8.22}$$

Please note the similarity to the unweighted definition in Formula (8.19). However, the weight of the inter-cluster edges is neglected. This can be integrated by modifying g:

$$g'(\mathcal{C}) := g(\mathcal{C}) \underbrace{+ M \cdot \left|\overline{E(\mathcal{C})}\right| - \omega\left(\overline{E(\mathcal{C})}\right)}_{=: g_w(\mathcal{C})} \ . \tag{8.23}$$

The additional term $g_w(\mathcal{C})$ corresponds to the difference of weight that would be counted if no inter-cluster edges were present and the weight that is assigned to the actual inter-cluster edges. In both cases the maximum is bounded by $M \cdot n(n-1)$, and the combined formula would be:

$$\mathsf{perf}_w(\mathcal{C}) = \frac{f(\mathcal{C}) + g(\mathcal{C}) + \vartheta \cdot g_w(\mathcal{C})}{n(n-1)M} \quad , \tag{8.24}$$

where $\vartheta \in [0,1]$ is a scaling parameter that rates the importance of the weight of the inter-cluster edges (with respect to the weight of the intra-cluster edges). In this way there is a whole family of weighted performance indices.

An alternative variation is based on the duality. Instead of counting 'correct' classified node pairs the number/weight of the errors is measured. Equation (8.21) will be the foundation:

$$\begin{aligned} \tilde{f}(\mathcal{C}) &= \sum_{i=1}^{k} \left(M|C_i|(|C_i|-1) - \theta \cdot \omega(E(C_i)) \right) \quad \text{and} \\ \tilde{g}(\mathcal{C}) &= \omega\left(\overline{E(\mathcal{C})}\right) \quad , \end{aligned} \tag{8.25}$$

where θ is a scaling parameter that rates the importance of the weight of the intra-cluster edges (with respect to the weight of the inter-cluster edges). The different symbols for density $\tilde{f}$ and sparsity $\tilde{g}$ functions are used to clarify that these functions perform inversely to the standard functions f and g with respect to their range: small values indicate better structural behavior instead of large values. Both can be combined to a standard index via

$$\mathsf{perf}_m(\mathcal{C}) = 1 - \frac{\tilde{f}(\mathcal{C}) + \tilde{g}(\mathcal{C})}{n(n-1)M} \quad . \tag{8.26}$$

Note that the versions are the same for $\vartheta = \theta = 1$. In general, this is not true for other choices of ϑ and θ. Both families have their advantages and disadvantages. The first version (Equation (8.24)) should be used, if the clusters are expected to be heavy, while the other version (Equation (8.26)) handles clusters with inhomogeneous weights better.

8.1.4 Other Indices

Clearly these indices are only a small fraction of the whole spectrum used to formalize and evaluate clusterings. However, they clarify different concepts very well. This part covers some historical indices that have been used for clustering, as well as some measures that are based on certain assumptions.

Clustering was originally applied to entities embedded in metric spaces or vector spaces with a norm function. Many indices have been developed that use essential structural properties of these spaces, e.g., the possibility to calculate a barycenter or geometric coverings. Over time, some indices have been transferred to graph clustering as well. Because these spaces usually provide all pair

information, the first problem is to estimate the similarity of nodes that are not connected with an edge. This is often solved using shortest path techniques that respect all information, like weight or direction, or only partial information, or none. The most common measures resulting are: diameter, edge weight variance, and average distance within the clusters. In contrast to the previous indices, these measures do not primarily focus on the intra-cluster density versus inter-cluster sparsity paradigm. Most of them even ignore the inter-cluster structure completely. Another difference is that these indices usually rate each cluster individually, regardless of its position within the graph. The resulting distribution is then rated with respect to the average or the worst case. Thus, a density measure[2] π can be transformed into an index by applying π on all cluster-induced subgraphs and rating the resulting distribution of values, e.g., via minimum, maximum, average, or mean:

$$\begin{aligned} \text{worst case: } & \min_i \{\pi(G[C_1]), \ldots, \pi(G[C_k])\} \\ \text{average case: } & \frac{1}{k} \sum_i \pi(G[C_i]) \\ \text{best case: } & \max_i \{\pi(G[C_1]), \ldots, \pi(G[C_k])\} \end{aligned}$$

Their popularity is partially based on the easy computation in metric or normed vector spaces, where the distance (inverse similarity) of (all) pairs is defined by their distance within the given space. The other reason is their use for greedy approaches that will be introduced in Section 8.2.1.

Another kind of measure compares the partition with an average case of the graph instead of an ideal case. This is especially preferred when the ideal situation is unknown, i.e., the network has very specific properties and thus general clustering paradigms could hardly be fulfilled. For example, Newman [442] postulated that a (uniform) random graph does not contain a clustering structure at all. Therefore, measuring the difference of numbers of edges within a cluster and the expected number of edges that would randomly fall within the cluster should be a good indicator whether the cluster is significant or not. The whole formula is shown in Equation (8.27).

$$\sum_{i=1}^{k} \left(|E(C_i)| - m \frac{|C_i| \cdot (|C_i| - 1)|}{n \cdot (n-1)} \right) \quad . \tag{8.27}$$

One advantage is that one can now distinguish between good, 'random', and bad clusterings depending on the sign and magnitude of the index. However, the expected number of edges that fall at random in a subgraph may be a too global view. The local density may be very inhomogeneous and thus an average global view can be very inaccurate. Figure 8.5 sketches such an example. The graph consists of two groups, i.e., a cycle (on the left side) and a clique (on the right side), which are connected by a path. In this example there is no subgraph with average density, in fact the two groups have a significantly different density. One

[2] greater values imply larger density

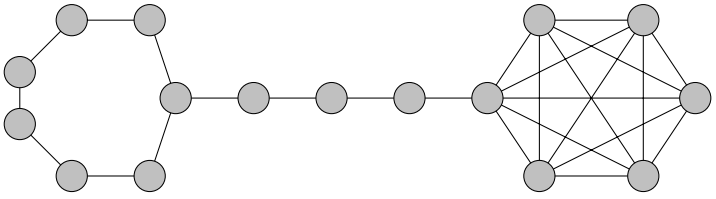


Fig. 8.5. Graph with inhomogeneous density

way to restrict the global view of average density is to fix the degree of each node and consider expected number of edges that fall at random in a subgraph. The modified formula is given in Equation (8.28).

$$\sum_{i=1}^{k} \left[\frac{\omega(E(C_i))}{\omega(E)} - \left(\frac{\sum_{e \in E(C_i,V)} \omega(e)}{\omega(E)} \right)^2 \right] . \tag{8.28}$$

In general, these measures that compare the partition with an average case of the graph seem to introduce a new perspective that has not yet been fully explored.

8.1.5 Summary

The presented indices serve as quality measurements for clusterings. They provide a formal way to capture the intuitive notation of natural decompositions. Several of these indices can be expressed in a simple framework that models the paradigm of intra-cluster density versus inter-cluster sparsity. It has been introduced in Section 8.1. Table 8.1 summarizes these indices, the general form is given in Equation (8.29).

$$\mathsf{index}\,(\mathcal{C}) := \frac{f\,(\mathcal{C}) + g\,(\mathcal{C})}{N} \tag{8.29}$$

A commonality of these measures is that the associated optimization problem, i.e., finding the clustering with best score, usually is $\mathcal{NP}$-hard. If the optimal structure is known in advance, as is the case for coverage or conductance, the restriction to interesting or practically relevant clusterings leads to $\mathcal{NP}$-hard problems.

Bibliographic Notes. Further information about indices can be found in [324, 325]. In [569, Chapter 7] a collection of density concepts is presented that have nearly the same intuition. Benchmark tests that are based on indices can be found in [95].

There is also a lot of work done within the theory community, especially disciplines like machine learning or neural networks, which targets approximating indices or extracting structural properties to qualitatively rate partitions. Owing to our limited scope, these aspects could not be covered. [170, 565] may serve as an entry point to these issues.

Table 8.1. Summary of clustering indices

Name	$f(\mathcal{C})$	$g(\mathcal{C})$	N
coverage $\gamma(\mathcal{C})$	$\omega(E(\mathcal{C}))$	0	$\omega(E)$
intra-cluster conductance $\alpha(\mathcal{C})$	$\min\limits_{1\leq i\leq k} \varphi(G[C_i])$	0	1
inter-cluster conductance $\delta(\mathcal{C})$	0	$\begin{cases} 1, & \text{if } \mathcal{C} = \{V\} \\ 1 - \max\limits_{1\leq i\leq k} \varphi((C_i, V \setminus C_i)), & \text{otherwise} \end{cases}$	1
performance $\mathsf{perf}(\mathcal{C})$	$\sum\limits_{i=1}^{k} \lvert E(C_i)\rvert$	$\sum\limits_{u,v\in V} [(u,v) \notin E] \cdot [u \in C_i, v \in C_j, i \neq j]$	$n(n-1)$
performance $\mathsf{perf}_w(\mathcal{C})$	$\sum\limits_{i=1}^{k} \omega(E(C_i))$	$M \sum\limits_{u,v\in V} [(u,v) \notin E] \cdot [u \in C_i, v \in C_j, i \neq j]$ $+\vartheta \cdot \left(M \cdot \left\lvert\overline{E(\mathcal{C})}\right\rvert - \omega\left(\overline{E(\mathcal{C})}\right)\right)$	$M \cdot n(n-1)$
performance $1 - \mathsf{perf}_m(\mathcal{C})$	$\sum\limits_{i=1}^{k} \left(M\lvert C_i\rvert(\lvert C_i\rvert - 1) - \theta \cdot \omega(E(C_i)\right)$	$\omega\left(\overline{E(\mathcal{C})}\right)$	$M \cdot n(n-1)$

8.2 Clustering Methods

This section covers the description of algorithmic paradigms and approaches used to calculate clusterings. It is split in three parts: First, some fundamental techniques are explained. Second, instances that embody these principles are described. Third, related problems are discussed.

8.2.1 Generic Paradigms

The description of generic methods is structured into three groups: Greedy and shifting approaches as well as general optimizing techniques utilized to find near-optimal clusterings. Mostly the concepts and algorithms apply a certain number of reductions to obtain an instance where a solution can easily be computed, and then extend it to the original input. This is very similar to the standard divide and conquer techniques, where instances are recursively divided until the problem is trivially solvable. In contrast to this, the reduction step is more general and can modify the instance completely. Thus, the reduced instance does not need to be a subinstance of the original problem. The reduction step is usually composed of two parts itself. First, significant substructures are identified. These can be bridges that are likely to separate clusters, or dense groups that indicate parts of clusters. Such an identification is often formulated as a hypothesis, and the recognized substructures are considered as evidence for its correctness. After the recognition phase a proper modification is applied to the current input graph. Such transformations can be arbitrary graph operations, however, the usual modifications are: addition and deletion of edges, as well as collapsing subgraphs, i.e., representing a subgraph by a new meta-node. A sketch of this idea is given in Figure 8.6. The 'shapes' of the (sub)instances indicate the knowledge of clustering, i.e., smooth shapes point to fuzzy information while the rectangles indicate exact knowledge. In the initial instance (left figure, upper row), no clustering information is present. During the reduction phases parts of the graph became more and more separated, which is indicated by the disjoint objects. The right instance in the upper row can be easily solved, and the fuzzy information is transformed into exact information with respect to the given instance. Based upon it, the expansion phase transfers this knowledge to the original instance (left figure, lower row). An additional difference to divide and conquer methods is that the size of the considered graphs can increase during the reduction phases.

Greedy Concepts. Most greedy methods fit into the following framework: start with a trivial and feasible solution and use update operations to lower its costs recursively until no further optimization is possible. This scheme for a greedy approach is shown in Algorithm 19, where $c(L)$ denotes the cost of solution L and $N_g(L)$ is the set of all solutions that can be obtained via an update operation starting with solution L. This iterative scheme can also be expressed for clusterings via hierarchies. A hierarchy represents an iterative refinement (or

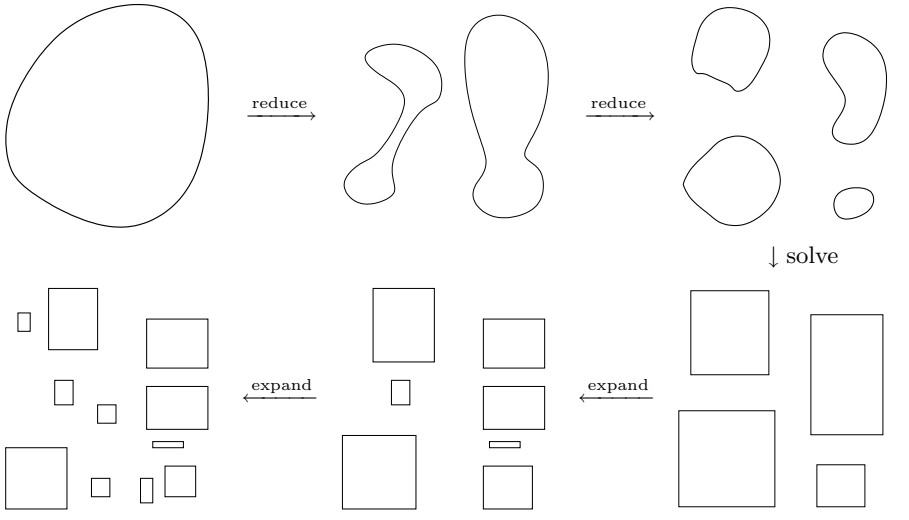


Fig. 8.6. Example of successive reductions. The 'shapes' of the (sub)instances indicate the knowledge of clustering, i.e., smooth shapes point to fuzzy information while the rectangles indicate exact knowledge. The first row shows the application of modifications, while the second indicates the expansion phases

Algorithm 19: Scheme for greedy methods

let L_0 be a feasible solution
$i \leftarrow 0$
while $\{L : L \in N_g(L_i), c(L) < c(L_i)\} \neq \emptyset$ **do**
  $L_{i+1} \leftarrow \operatorname{argmin}_{L \in N_g(L_i)} c(L)$
  $i \leftarrow i + 1$
return L_i

coarsening) process. Greedy methods that use either merge or split operations as updates define a hierarchy in a natural way. The restriction to one of these operations guarantees the comparability of clusterings, and thus leads to a hierarchy. These two concepts will be formalized shortly, before that some facts of hierarchies are briefly mentioned.

Hierarchies provide an additional degree of freedom over clusterings: the number of clusters is not fixed. Thus, they represent the group structure independently of its granularity. However, this feature usually increases the space requirement, although a hierarchy can be implicitly represented by a tree, also called a dendrogram, that represents the merge operations. Algorithms tend to construct it explicitly. Therefore, their space consumption is often quadratic or larger. For a few special cases these costs can be reduced with the help of data structures [178].

The *Linkage process* (Agglomeration) iteratively coarses a given clustering by merging two clusters until the 1-clustering is reached. The formal description is shown in Definition 8.2.1.

Definition 8.2.1 (Linkage). *Given a graph $G = (V, E, \omega)$, an initial clustering $\mathcal{C}_1$ and either a global cost function $c_{global} \colon \mathcal{A}(G) \to \mathbb{R}_0^+$ or a cost function $c_{local} \colon \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$ for merging operations, the* Linkage *process merges two clusters in the current clustering $\mathcal{C}_i := \{C_1, \ldots, C_k\}$ while possible in the following recursive way:*

global: let P be the set of all possible clusterings resulting from $\mathcal{C}_i$ by merging two clusters, i.e.,

$$\mathsf{P} := \left\{ \{C_1, \ldots, C_k\} \setminus \{C_\mu, C_\nu\} \cup \{C_\mu \cup C_\nu\} \mid \mu \neq \nu \right\} ,$$

then the new clustering $\mathcal{C}_{i+1}$ is defined as an element in P with minimum cost with respect to c_{global}.

local: let μ, ν be those two distinct indices such that c_{local} has one global minimum in the pair (C_μ, C_ν), then the new clustering $\mathcal{C}_{i+1}$ is defined by merging C_μ and C_ν, i.e.,

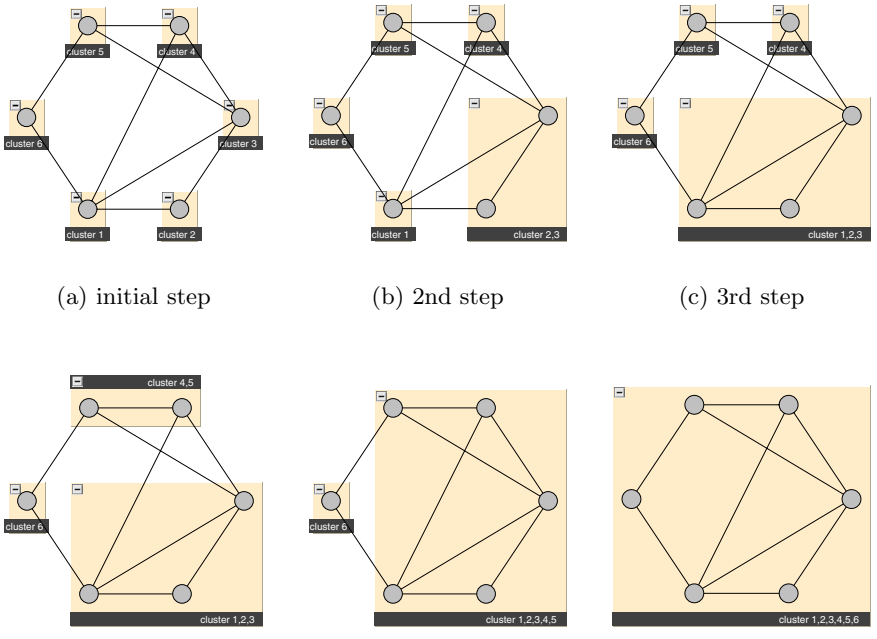
$$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \setminus \{C_\mu, C_\nu\} \cup \{C_\mu \cup C_\nu\} .$$

Although the definition is very formal, the basic idea is to perform a cheapest merge operation. The cost of such an operation can be evaluated using two different view points. A local version charges only merge itself, which depends only on the two involved clusters. The opposite view is a global version that considers the impact of the merge operation. These two concepts imply also the used cost functions, i.e., a global cost function has the set of clusterings as domain while the local cost function uses a pair of node subsets as arguments. An example of linkage is given in Figure 8.7. The process of linkage can be reversed, and, instead of merging two clusters, one cluster is split into two parts. This dual process is called *Splitting* (Diversion). The formal description is given in Definition 8.2.2.

Definition 8.2.2 (Splitting). *Let a graph $G = (V, E, \omega)$, an initial clustering $\mathcal{C}_1$, and one of the following function sets be given:*

global: a cost function $c_{global} \colon \mathcal{A}(G) \to \mathbb{R}_0^+$
semi-global: a cost function $c_{global} \colon \mathcal{A}(G) \to \mathbb{R}_0^+$ and a proper cut function $S_{local} \colon \mathcal{P}(V) \to \mathcal{P}(V)$
semi-local: a cost function $c_{local} \colon \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$ and a proper cut function $S_{local} \colon \mathcal{P}(V) \to \mathcal{P}(V)$
local: a cost function $c_{local} \colon \mathcal{P}(V) \times \mathcal{P}(V) \to \mathbb{R}_0^+$

The Splitting *process splits one cluster in the current clustering $\mathcal{C}_i := \{C_1, \ldots C_k\}$ into two parts. The process ends when no further splitting is possible. The cluster that is going to be split is chosen in the following way:*



(a) initial step (b) 2nd step (c) 3rd step

(d) 4th step (e) 5th step (f) final step

Fig. 8.7. Example of linkage

global: *let* P *be the set of all possible clusterings resulting from* $\mathcal{C}_i$ *by splitting one cluster into two non-empty parts, i.e.,*

$$P := \left\{ \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{C'_\mu, C_\mu \setminus C'_\mu\} \mid \emptyset \neq C'_\mu \subsetneq C_\mu \right\} ,$$

then the new clustering $\mathcal{C}_{i+1}$ *is defined as an element in* P *with minimum cost with respect to* c_{global}.

semi-global: *let* P *be the set of all possible clusterings resulting from* $\mathcal{C}_i$ *by splitting one cluster into two non-empty parts according to* S_{local}*, i.e.,*

$$P := \left\{ \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{S_{local}(C_\mu), C_\mu \setminus S_{local}(C_\mu)\} \right\} ,$$

then the new clustering $\mathcal{C}_{i+1}$ *is defined as an element in* P *with minimum cost with respect to* c_{global}.

semi-local: *let* μ *be an index such that* c_{local} *has one global minimum in the pair* $(S_{local}(C_\mu), C_\mu \setminus S_{local}(C_\mu))$ *then the new clustering* $\mathcal{C}_{i+1}$ *is defined by splitting cluster* C_μ *according to* S_{local}*, i.e.,*

$$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{S_{local}(C_\mu), C_\mu \setminus S_{local}(C_\mu)\} .$$

local: *let μ be an index and C_ν be a proper subset of cluster C_μ such that c_{local} has one global minimum in the pair $(C_\nu, C_\mu \setminus C_\nu)$, then the new clustering $\mathcal{C}_{i+1}$ is defined by splitting cluster C_μ according to S_{local}, i.e.,*

$$\mathcal{C}_{i+1} := \{C_1, \ldots, C_k\} \setminus \{C_\mu\} \cup \{C_\mu, C_\mu \setminus C_\nu\} \ .$$

Similar to the Linkage process, the definition is rather technical but the basic idea is to perform the cheapest split operation. In contrast to the Linkage method, the cost model has an additional degree of freedom because clusters can be cut in several ways. Again, there is a global and a local version that charge the impact of the split and the split itself, respectively. Both correspond to the views in the Linkage process. However, rating every possible non-trivial cut of the clusters is very time consuming and usually requires sophisticated knowledge of the involved cost functions. One way to reduce the set of possible splittings is to introduce an additional cut function S_{local}. It serves as an 'oracle' to produce useful candidates for splitting operations. The semi-global and semi-local versions have the same principles a the global and the local version, however, their candidate set is dramatically reduced. Therefore, they are often quite efficiently computable, and no sophisticated knowledge about the cost function is required. However, the choice of the cut function has usually a large impact on the quality.

Both, the Linkage and the Splitting process, are considered to be greedy for several reasons. One is the construction of the successive clusterings, i.e., an update operation chooses always the cheapest clustering. These can produce total or complete hierarchies quite easily. Total hierarchies can be achieved by simply adding the trivial clusterings to the resulting hierarchy. They are comparable to all other clusterings, therefore preserving the hierarchy property. Recall that complete hierarchies are hierarchies such that a clustering with k clusters is included for every integer $k \in [1, n]$. Both processes lead to a complete hierarchy when initialized with the trivial clusterings, i.e., singletons for Linkage and the 1-clustering for Splitting. Note that in the case of the Splitting process, it is essential that the cut functions are proper. Therefore, it is guaranteed that every cluster will be split until each cluster contains only one node. Although the cost can be measured with respect to the result or the operation itself, clairvoyance or projection of information into the future, i.e., accepting momentarily higher cost for a later benefit, is never possible.

Because of their simple structure, especially in the local versions, both concepts are frequently used and are also the foundation of clustering algorithms in general. The general local versions can be very efficiently implemented. For the Linkage process, a matrix containing all cluster pairs and their merging cost is stored. When an update operation takes place, only the cost of merging the new resulting cluster with another is recalculated. For certain cost functions this scheme can even be implemented with less than quadratic space and runtime consumption [178]. In the case of the Splitting process, only the cut information needs to be stored for each cluster. Whenever a cluster gets split, one has to recompute the cut information only for the two new parts. This is not true for any global version in general. However, a few very restricted cost and cut functions can be handled efficiently also in the global versions.

Shifting Concept. In contrast to the global action of the previous greedy strategies, the shifting approaches work more locally. They choose an initial clustering and iteratively modify it until a local optimum is found. Usually three operations are permitted: first, a node moves from one cluster to another existing cluster, second, a node moves from one cluster to create a new cluster, and, third, two nodes exchange their cluster assignments. Sometimes more complex operations, like instant removal of one cluster and reassigning nodes to already existing clusters, are allowed. However, these complex operations are usually only used for speed-up, or to avoid artifical situations, therefore they will not be discussed here. Note that one can typically simulate them by a sequence of simple operations. Regarding algorithmic aspects, shifting concepts are relatively close to the greedy ones. Algorithm 20 shows the general procedure where $N_\mathrm{s}(L)$ denotes the set of all clusterings that can be obtained by applying the modifications to the clustering L. Step 1 can be based on cost or potential functions,

Algorithm 20: Scheme for shifting methods

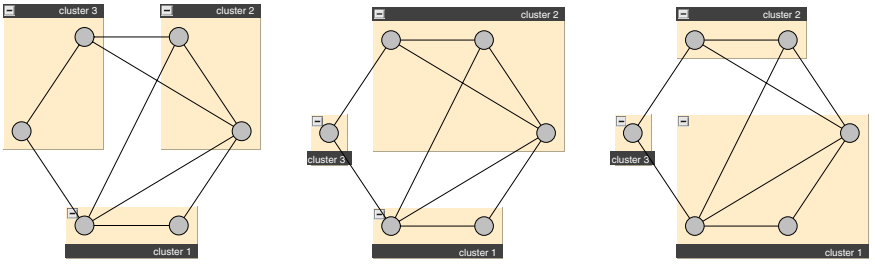
```
   let L_0 be an initial solution
   i ← 0
   while N_s(L_i) ≠ ∅ do
1  |  choose L_{i+1} ∈ N_s(L_i)
   |  i ← i + 1
   return L_i
```

random selecting, or on the applicational background. The technical definition of the shifting concept is given in Definition 8.2.3, and uses a potential function as selection criteria.

Definition 8.2.3. *Given a graph $G = (V, E, \omega)$, an initial clustering $\mathcal{C}_1$ and a potential function $\Phi\colon \mathcal{A}(G) \times \mathcal{A}(G) \to \mathbb{R}$, the* Shifting *process is defined as performing any operation on the current clustering $\mathcal{C}_i$ that results in a new clustering $\mathcal{C}_{i+1}$ such that $\Phi(\mathcal{C}_i, \mathcal{C}_{i+1}) > 0$.*

Shifting concepts have many degrees of freedom. The choice of potential functions is critical. There are two common subtypes of potentials: type-based and compressed. *Type-based potentials* are functions that heavily depend on the type of operations. They are often used to preserve certain properties. In the case that creating a new cluster via a node movement is very expensive in contrast to the other node movements, it is very likely that the number of clusters in the final clustering is roughly the same as in the initial clustering. *Compressed* or *sequenced shifts* collapse a series of operations into one meta-operation, and rate only the output of this operation with its argument. Thus, a certain number of operations are free of charge. These functions are often used in combination with a standard potential function that has many local optima. By ignoring a number of intermediate steps, it may be easier to reach a global optimum.

Owing to their contingent iterative nature, shifting approaches are rarely used on their own. There can be sequences of shifting operations where the initial and final clustering are the same, so-called loops. Also, bounds on the runtime are more difficult to establish than for greedy approaches. Nonetheless, they are a common postprocessing step for local improvements. An example of shifting is given in Figure 8.8.



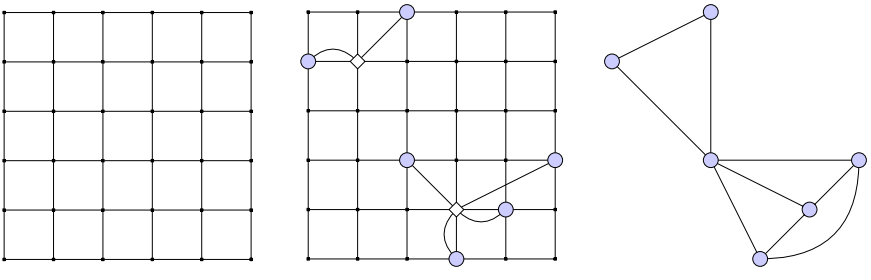
(a) initial step (b) 2. step (c) 3. step

Fig. 8.8. Example of shifting

General Optimization Concepts for Clustering. The two previous concepts, the greedy and the shifting framework, were fairly adapted. Both defined precisely the permitted operations and constraints, and the conditions of their application. The following concepts can be used for arbitrary optimization approaches. They are based on the idea that clusterings can be formulated as the result of a generic optimization process. The input data may be generated in a certain way with an implicit clustering structure. The optimization problem is to extract a clustering that is relatively close to the hidden one. Alternatively, the contained clustering is the result of an unknown optimization process. It is only known that this process respects certain paradigms, like intra-cluster density, inter-cluster sparsity, or both. The related problem is again to extract a clustering that is relatively close to the hidden one. The variety of techniques to solve optimization problems is gigantic, therefore only the following are considered here: parameter estimation, evolutionary and search-based approaches.

Parameter estimation is based on the assumption that the input data was created by a (random) sampling process: There is a hidden graph with a certain clustering, then a sample of the (whole) graph is drawn that will be the input graph. The approach then tries to estimate the parameters of this sampling process. These parameters will be used to reconstruct the clustering of the hidden graph. Originally, these methods were introduced to find clusterings for data embedded in metric spaces [325, Section 5.3]. There are clusterings that may be

represented by a union of distributions, and the goal is to estimate the number of distributions and their parameters (mean, deviation, etc.). The *Expectation Maximization* (EM) is the most commonly used method. In general, it is only applied to data that is embedded in a metric space. Although graphs are typically not embedded, one can think of many processes that involve an implicit (metric) topology. An example is the following: let a finite space with a topology, a set of points, referred to as cluster centers, and a probability distribution for each cluster center be given; then n nodes/points are introduced by choosing one cluster center and a free spot around it that respects both the topology and its distribution. Two nodes will be connected by edges if their distance (with respect to the topology) is smaller than or equal to a given parameter. An example of this process is shown in Figure 8.9. Thus, the graph (Figure 8.9(c)) would be



(a) initial topology

(b) 2 cluster centers (diamonds) and 6 nodes (circles)

(c) resulting graph with distance threshold 3

Fig. 8.9. Example of generating a graph and its clustering using distributions.

the input graph, and the estimation approach would try to estimate the number of cluster centers as well as the assignment of each node to a cluster center. In the EM case, the resulting clustering should have the largest expectation to be the original hidden clustering, i.e., the same number of cluster points and the correct node cluster-point assignment.

Evolutionary approaches such as genetic algorithms (GA), evolution strategies (ES) and evolutionary programming (EP) iteratively modify a population of solution candidates by applying certain operations. 'Crossover' and 'mutation' are the most common ones. The first creates a new candidate by recombining two existing ones, whilst the second modifies one candidate. To each candidate a fitness value is associated, usually the optimization function evaluated on the candidate. After a number of basic operations, a new population is generated based on the existing one, where candidates are selected according to their fitness value. A common problem is to guarantee the feasibility of modified solutions. Usually this is accomplished by the model specification. In the context of cluster-

ing, the model can either use partitions or equivalence relations. As presented in Section 8.1.3, clusterings can be modeled as 0-1 vectors with certain constraints.

Search-based approaches use a given (implicit) topology of the candidate space and perform a random walk starting at an arbitrary candidate. Similar to evolutionary approaches, the neighborhood of a candidate can be defined by the result of simple operations like the mutations. The neighborhood of a clustering usually is the set of clusterings that result from node shifting, cluster merging, or cluster splitting. The selection of a neighborhood is also based on some fitness value, usually the optimization function evaluated on the candidate. The search usually stops after a certain number of iterations, after finding a local optimum, or a combination of both.

8.2.2 Algorithms

Clustering methods have been developed in many different fields. They were usually very adapted, either for specific tasks or under certain conditions. The reduction of algorithms to their fundamental ideas, and constructing a framework on top, started not that long ago. Thus, this part can only give a short synopsis about commonly used methods.

Instances of Linkage. The different instances of the Linkage framework were originally designed for distance edge weights. Distances are the 'dual' version of similarities. Historically, the input data for clusterings algorithms was metrically embedded and complete (the similarity/dissimilarity of every pair is known). In these scenarios, it is possible to find clusterings using distance functions instead of similarities, i.e., one has to search for spatially dense groups that are well-separated from each other. If the distance function is only partially known, it is no longer possible to derive information about the similarity of two objects from their distance to other objects.

However, the use of distance functions had certain advantages that can be carried over to similarity weights. One reason was that distances can be easily combined to estimate the distance of a path. The most common way is the summation of the edge weights along the path. The standard local cost functions are defined as:

$$c_{local}(C_i, C_j) := \bigodot \{d(u,v) \colon u \in C_i, v \in C_j\} \ , \tag{8.30}$$

where $d(u,v)$ is the length of any shortest path connecting u and v, and $\bigodot$ is any set evaluation function, like minimum, average, or maximum. Indeed these three versions are called *Single Linkage*, *Average Linkage*, and *Complete Linkage*. A possible explanation of the name Single Linkage is that the cheapest, shortest path will be just an edge with minimum weight inside of $E(C_i, C_j)$. Note that the cost function can be asymmetric and have infinity as its value. Also, note that it is necessary to use the length of the shortest paths because not every node pair $(u,v) \in C_i \times C_j$ will be connected by an edge. In fact, the set $E(C_i, C_j)$ will be empty in the ideal case.

When dealing with similarities instead of distances one has to define a meaningful path 'length'. A simple way is to ignore it totally and define the cost function as

$$c_{local}(C_i, C_j) := \bigodot\{M - \omega(e) \colon e \in E(C_i, C_j)\} \ , \tag{8.31}$$

where M is the maximum edge weight in the graph. Alternatively, one can define the similarity of a path $P \colon v_1, \ldots, v_\ell$ by:

$$\omega(P) := \left(\sum_{i=1}^{\ell-1} \frac{1}{\omega(v_i, v_{i+1})} \right)^{-1} \ . \tag{8.32}$$

Although this definition is compatible with the triangle inequality, the meaning of the original range can be lost along with other properties. Similar to cost definition in Equation (8.31), the distance value (in Equation(8.30)) would be replaced by $(n-1)M - \omega(P)$. These 'inversions' are necessary to be compatible with the range meaning of cost functions. Another definition that is often used in the context of probabilities is

$$\omega(P) := \prod_{i=1}^{\ell-1} \omega(v_i, v_{i+1}) \ . \tag{8.33}$$

If $\omega(v_i, v_{i+1})$ is the probability that the edge (v_i, v_{i+1}) is present and these probabilities are independent of each other, then $\omega(P)$ is the probability that the whole path exists.

Lemma 8.2.4. *The dendrogram of Single Linkage is defined by a Minimum Spanning Tree.*

Only a sketch of the proof will be given. A complete proof can be found in [324]. The idea is the following: consider the algorithm of Kruskal where edges are inserted in non-decreasing order, and only those that do not create a cycle. From the clustering perspective of Single Linkage, an edge that would create a cycle connects two nodes belonging to the same cluster, thus that edge cannot be an inter-cluster edge, and thus would have never been selected.

The Linkage framework is often applied in the context of sparse networks and networks where the expected number of inter-cluster edges is rather low. This is based on the observation that many Linkage versions tend to produce chains of clusters. In the case where either few total edges or few inter-cluster edges are present, these effects occur less often.

Instances of Splitting. Although arbitrary cut functions are permitted for Splitting, the idea of sparse cuts that separate different clusters from each other has been the most common one. Among these are: standard cuts (Equation (8.34)), Ratio Cuts (Equation (8.35)), balanced cuts (Equation (8.36)),

Table 8.2. Definition of various cut functions

$$S(V) := \min_{\emptyset \neq V' \subset V} \omega(E(V', V \setminus V')) \tag{8.34}$$

$$S_{\text{ratio}}(V) := \min_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{|V'| \cdot (|V| - |V'|)} \tag{8.35}$$

$$S_{\text{balanced}}(V) := \min_{\emptyset \neq V' \subset V} \frac{\omega(E(V', V \setminus V'))}{\min(|V'|, (|V| - |V'|))} \tag{8.36}$$

$$S_{\text{conductance}}(V) := \min_{\emptyset \neq V' \subset V} \delta(V') \tag{8.37}$$

$$S_{2\text{–sector}}(V) := \min_{\substack{V' \subset V, \\ \lfloor |V|/2 \rfloor \leq |V'| \leq \lceil |V|/2 \rceil}} \omega(E(V', V \setminus V')) \tag{8.38}$$

Conductance Cuts (Equation (8.37)), and Bisectors (Equation (8.38)). Table 8.2 contains all the requisite formulae.

Ratio Cuts, balanced cuts, and Bisectors (and their generalization, the k–Sectors) are usually applied when the uniformity of cluster size is an important constraint. Most of these measures are $\mathcal{NP}$-hard to compute. Therefore, approximation algorithms or heuristics are used as replacement. Note that balanced cuts and Conductance Cuts are based on the same fundamental ideas: rating the size/weight of the cut in relation to the size/weight of the smaller induced cut side. Both are related to node and edge expanders as well as isoperimetric problems. These problems focus on the intuitive notion of bottlenecks and their formalizations (see Section 8.1.2 for more information about bottlenecks). Some spectral aspects are also covered in Section 14.4, and [125] provides further insight. Beside these problems, the two cut measures have more in common. There are algorithms ([565]) that can be used to simultaneously approximate both cuts. However, the resulting approximation factor differs.

Splitting is often applied to dense networks or networks where the expected number of intra-cluster edges is extremely high. An example for dense graphs are networks that model gene expressions [286]. A common observation is that Splitting methods tend to produce small and very dense clusters.

Non-standard Instances of Linkage and Splitting. There are several algorithms that perform similar operations to 'linkage' or 'splitting', but do not fit into the above framework. In order to avoid application-specific details, only some general ideas will be given without the claim of completeness.

The *Identification of Bridge Elements* is a common Splitting variant, where cuts are replaced by edge or node subsets that should help to uncover individual clusters. One removal step can lead to further connected components, however it is not required. Figure 8.10 shows such an example. Most of the techniques that are used to identify bridge elements are based on structural indices, or properties

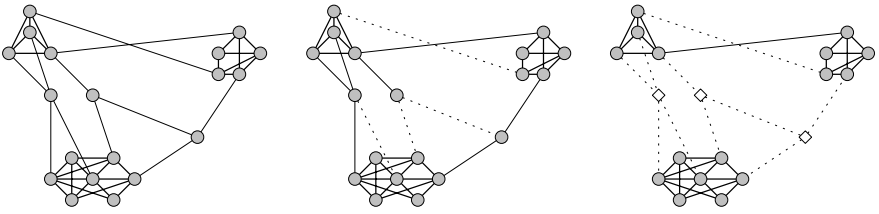
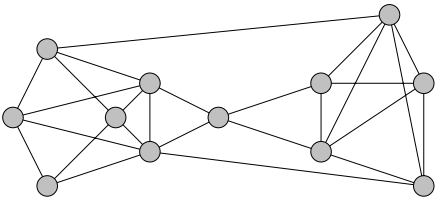


Fig. 8.10. Example for the removal of bridge elements. Removed elements are drawn differently: edges are dotted and nodes are reduced to their outline

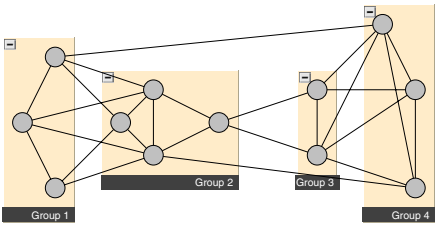
derived from shortest path or flow computations. Also centralities can be utilized for the identification [445].

Multi-Level Approaches are generalizations of the Linkage framework, where groups of nodes are collapsed into a single element until the instance becomes solvable. Afterwards, the solution has to be transformed into a solution for the original input graph. During these steps the previously formed groups need not be preserved, i.e., a group can be split and each part can be assigned to individual clusters. In contrast to the original Linkage framework, here it is possible to tear an already formed cluster apart. Multi-level approaches are more often used in the context of equi-partitioning, where k groups of roughly the same size should be found that have very few edges connecting them. In this scenario, they have been successfully applied in combination with shiftings. Figure 8.11 shows an example.

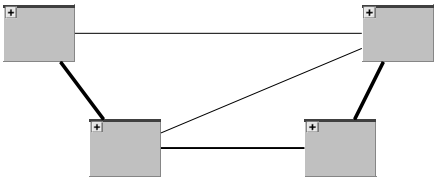
Modularity – as presented at the beginning of Section 8.2.1, clustering algorithms have a very general structure: they mainly consist of 'invertible' transformations. Therefore, a very simple way to generate new clustering algorithms is the re-combination of these transformations, with modifications of the sequence where appropriate. A reduction does not need to reduce the size of an instance, on the contrary it also can increase it by adding new data. There are two different types of data that can be added. The first is information that is already present in the graph structure, but only implicitly. For example, an embedding such that the distance in the embedding is correlated to the edge weights. Spectral embeddings are quite common, i.e. nodes are positioned according to the entries of an eigenvector (to an associated matrix of the graph). More details about spectral properties of a graph can be found in Chapter 14. Such a step is usually placed at the beginning, or near the end, of the transformation sequence. The second kind is information that supports the current view of the data. Similar to the identification of bridge elements, one can identify cohesive groups. Bridges and these cohesive parts are dual to each other. Thus, while bridges would be removed, cohesive groups would be extended to cliques. These steps can occur during the whole transformation sequence.



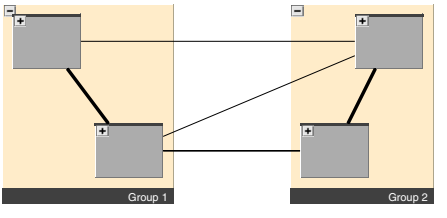
(a) input graph



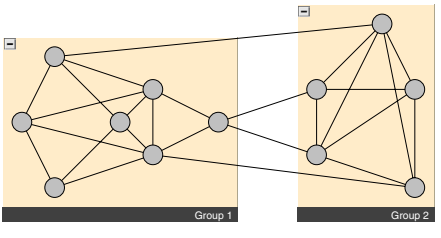
(b) identifying groups



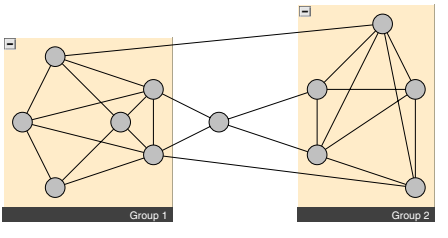
(c) collapsing groups



(d) solving the instance



(e) expanding internal groups



(f) locally optimizing groups

Fig. 8.11. Example of a multi-level approach

8.2.3 Bibliographic Notes

A good introduction to the variety of clustering techniques is given in [324, 325]. Although data mining is their primary motivation, other applicational aspects are covered as well. Equi-partitioning, i.e., finding a clustering where clusters have roughly the same size, is a slightly different problem from the general clustering problem. It is rooted in many divide and conquer methods. Many solving techniques involve splitting via cut functions and shiftings. Chip design (VLSI) is one major research field that has developed such algorithms. A survey that covers general methods as well as their relevance to VLSI is presented in [27]. More generally, finding good partitions, approximating sparse-cuts, and calculating (multi-commodity) flows has to be solved via (integer) linear programming. An introduction is given in [439, 507]. There are several other disciplines that deal with similar clustering and pattern matching problems, like Artifical Intelligence, neural networks, image processing [564], genetics [286], and facility location [439, 507]. Especially, facility location and its derivatives can be seen as extended clustering problems. There, a set of elements has to be covered by a subset of candidates, fulfilling certain constraints and optimizing a global target function. This is also closely related to the parameter estimation framework. An introduction to the general topic is provided in [169].

There is also a lot of work done within the theory community that develops sophisticated approximations and sampling techniques. These methods have the advantage that a provable guarantee of quality of resulting clustering can be given (see for example [36, 565, 170]). On the other hand, these techniques rarely operate on the clustering structure itself, but utilize other paradigms for finding a suitable decomposition. One of these tools is spectral decomposition. Unfortunately, these aspects go beyond the scope of this chapter. Some examples can be found in [27, 565, 170]. Spectral properties of networks are discussed in more detail in Chapter 14.

8.3 Other Approaches

The previous sections about quality measuring (Section 8.1) and clustering techniques (Section 8.2) provide an introduction both for application as well as for the topic as a whole. However, certain aspects have been neglected. Among them are extensions of clustering, alternative descriptions, axiomatics, dynamic updates, evaluation, pre- and post-processings. Discussing everything would definitely go beyond our scope, therefore only extensions of clusterings and axiomatics will be presented.

8.3.1 Extensions of Clustering

Clusterings were introduced as partitions of the node set. The extensions that are presented also group the node set.

Fuzzy clustering relaxes the disjoint constraint, thus clusters can overlap each other. The basic idea is that bridging elements belong to adjacent clusters

rather than build their own one. In order to avoid redundancy, one usually requires that a cluster is not contained in the union of the remaining clusters. Figure 8.12 shows such an example where the two groups have the middle node in common. It is seldom used, due to its difficult interpretation. For example,

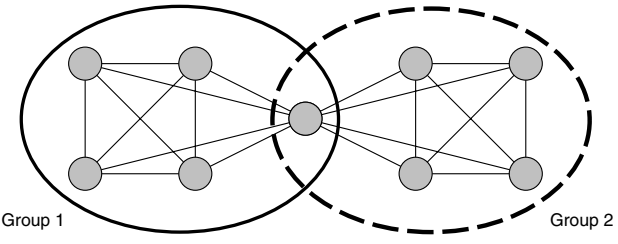


Fig. 8.12. Example of a fuzzy clustering

it is very difficult to judge single nodes or small node subsets that belong to a relatively large number of (fuzzy) clusters. When the number of clusters is also restricted, then artefacts occur more frequently. For example, if the number is restricted to a constant k, then a difficult situation is where more than k cliques of size at least k have a large number of nodes in common (see Figure 8.13(a)). If the number can scale according to a node's degree, then sparse clusters can be torn apart. For example, a star with k leaves may be decomposed into k fuzzy clusters, each containing one leaf and the central node (see Figure 8.13(b)).

Another extension is to enhance clusterings with *representatives*. Each cluster has a representative. It is very similar to the facility location problems (Section 8.2.3), where a candidate covers a subset of elements. This can be seen as 'representing' the group by one element. It is usually a node that is located 'in the center' of the cluster. This form of enhancement can be very effective when the graph is embedded in a metric or a vector space. In these cases the representative can also be an element of the space and not of the input. The concept is also used to perform speed-ups or approximate calculations. For example, if all the similarity/distance values between the nodes in two clusters are needed, then it can be sufficient to calculate the similarity/distance values between the representatives of the clusters.

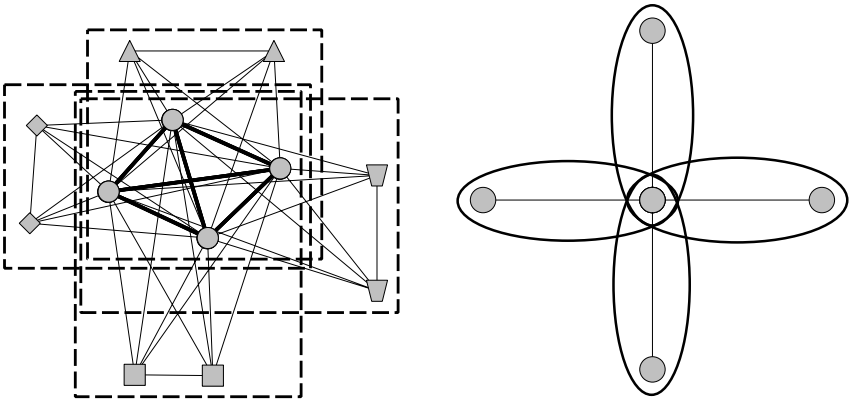
Nested clustering represents a nested sequence of node subsets, i.e., a mapping $\eta\colon \mathbb{N} \to \mathcal{P}(V)$ such that:

1. the subsets are nested, i.e.,

$$\forall\, i \in \mathbb{N}\colon \eta(i+1) \subseteq \eta(i)$$

2. and the sequence is finite, i.e.,

$$\exists\, k \in \mathbb{N}\colon \forall \ell > k\colon \eta(\ell) = \emptyset \ .$$



(a) Four cliques of size six that have a common K_4. Each maximum clique is a fuzzy cluster.

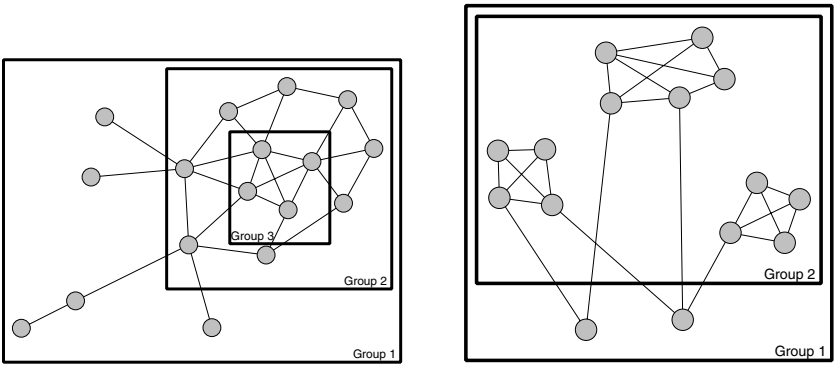
(b) A star with four leaves and each fuzzy cluster contains the center node and one leaf.

Fig. 8.13. Two examples of fuzzy clusterings where many clusters intersect each other

The smallest possible k is also called the *size* of the sequence, and $\eta(k)$ the top element. The intuition behind this structure is that the top element $\eta(k)$ consists of locally maximal dense groups. The density of the subsets $\eta(i)$ also decreases with decreasing argument i. Therefore the argument i can be seen as degree of density. One can distinguish two extreme types: The first one is called *hierarchies*, where each $\eta(i)$ induces a connected graph. The corresponding graphs can be seen as onions, i.e., having a unique core and multiple layers around it with different density. The second type is called *peaks*, and is complementary to hierarchies, i.e., at least one subset $\eta(i)$ induces a disconnected graph. An appropriate example may be boiling water, where several hotspots exist that are separated by cooler parts. If the graph has a skew density distribution then a hierarchy type can be expected. In this scenario, it can reveal structural information, unlike standard clustering. If the graph has a significant clustering, then peaks are more likely to occur. In that case, the top element consists of the core-parts of the original clusters. Figure 8.14 shows such examples. *Cores* are an often used realization of nested clusterings. They were introduced in [513, 47]. They can be efficiently computed and capture the intuition quite well.

8.3.2 Axiomatics

Axioms are the usual way to formulate generic properties and reduce concepts to their cores. They also provide a simple way to prove characteristics, once a property is proven for a system of axioms. Every structure fulfilling the axioms possesses this property. In the following, only Kleinberg's proposal introduced



(a) hierarchy (b) peaks

Fig. 8.14. Examples of a nested clustering

in [361] is presented. A graph clustering version of his result will be presented here. This version will be equivalent to the original.

Let $K_n = (V, E)$ be the complete graph on n nodes, and $\omega\colon E \to \mathbb{R}_0^+$ a distance function on the edges. The set of all possible distance functions is denoted by D.

Definition 8.3.1. *A* clustering function f *is a mapping* $f\colon D \to \mathcal{A}(K_n)$ *fulfilling the following axioms:*

Scale-Invariance:

$$\forall\, \alpha \in \mathbb{R}^+, \omega \in D\colon f(\omega) = f(\alpha \cdot \omega) \ ,$$

where $(\alpha \cdot \omega)(u, v) := \alpha \cdot (\omega(u, v))$

Richness: $f(D) = \mathcal{A}(K_n)$

Consistency: for all $\omega, \omega' \in D$ *hold*

$$\left(\forall\, u, v \in V\colon \omega'(u, v) \begin{cases} \leq \omega(u, v) & , \text{ if } u \sim_{f(\omega)} v \\ \geq \omega(u, v) & , \text{ otherwise} \end{cases}\right) \Longrightarrow f(\omega) = f(\omega') \ .$$

A brief intuition of the axioms in Definition 8.3.1 is given before the consequences are presented. Scale-invariance ensures that a clustering does not depend on hard-coded values, but rather on ratios. The clustering should not change if the distances are homogeneously increased. Richness ensures that every possible clustering has at least one edge weighting as a preimage. It is self-evident that every clustering should be constructable by assigning suitable weights to the edges. Finally, consistency handles the relation between different clusterings. Assume that ω is fixed, so $f(\omega)$ represents a clustering. If the weights on the edges are changed respecting $f(\omega)$, then the clustering should be respected as well. The modifications on ω consist of non-increasing the distance inside of clusters and

non-decreasing the distance between clusters. In this way clusters may become more compact (distances can decrease) and different clusters may become more separated (distances can increase).

Theorem 8.3.2 ([361, Theorem 2.1]). *For all $n \geq 2$ there exists no clustering function.*

Only a sketch of the proof is given here, but full details can be found in [361].

Sketch of Proof. Assume there exists a clustering function f for a fixed n. The basic idea is to show that $f(D)$, i.e., the image of f, cannot contain both a clustering and a refinement of it simultaneously. This is achieved by using the axiom consistency (and scale-invariance). There exists a clustering $\mathcal{C}$ in $\mathcal{A}(K_n)$ such that at least one cluster has more than one element, and an edge weighting ω with $f(\omega) = \mathcal{C}$. Informally speaking, it is possible to modify ω to isolate a proper subset of a cluster within $\mathcal{C}$. This can be done in such a way that consistency is fulfilled for the two weighting functions. However, one can show that the modified weighting also leads to a refinement of $\mathcal{C}$. Therefore, it is not possible that $f(D)$ contains a clustering and its refinement. Thus $f(D)$ is an antichain. This contradicts the fact that $\mathcal{A}(K_n)$ is not an antichain for $n \geq 2$. □

Theorem 8.3.2 is a negative result for clustering functions and, as a consequence, for clustering algorithms. However, the set of axioms is very restrictive. In fact, there are many functions that are close to being clustering functions.

Lemma 8.3.3 ([361, Theorem 2.2]). *There are many functions that fulfill two of the three axioms of Definition 8.3.1.*

For every combination of two axioms, there is a selection criterion for Single Linkage such that it is a clustering function fulfilling these two axioms.

Further investigations reveal that relaxing the conditions of scale-invariance and consistency to refinements leads to clustering functions. This is not the only possibility. Another way is to explore the 'lack' of information (edges). Usually graphs are not complete, and cannot be complete for several reasons. The standard technique, completing the edge set and assigning only extremely large values to the additional edges, is not possible in this scenario. First of all, all values need to be finite, and, second, the two axioms, scale-invariance and consistency, allow general manipulations so there is no guarantee that an artificially introduced extreme value can keep its role. In order to include the lack of information, a clustering function could be defined as a mapping from the set of weighted relations to the set of partitions over the set. For a set of elements X, the set of all weighted (binary) relations over X is denoted by $\Omega(X)$ – or in short Ω, if the set X is clear. For every relation $\omega \in \Omega(X)$ its domain is given by $E(\omega)$.

Definition 8.3.4. *Given a set of elements X, a* graph clustering function *f is a mapping $f\colon \Omega(X) \to \mathcal{A}(X)$ fulfilling the following axioms:*

Scale-Invariance:

$$\forall\, \alpha \in \mathbb{R}^+, \omega \in \Omega(X)\colon f(\omega) = f(\alpha \cdot \omega)$$

Richness: $f(\Omega(X)) = \mathcal{A}(X)$
Consistency: *for all* $\omega, \omega' \in \Omega(X)$ *with* $E(\omega) \subseteq E(\omega')$ *hold*

$$\left(\forall\, u, v \in V \colon \omega'(u,v) \begin{cases} \leq \omega(u,v) & , \text{ if } u \sim_{f(\omega)} v \\ \geq \omega(u,v) & , \text{ otherwise} \end{cases}\right) \Longrightarrow f(\omega) = f(\omega') \ ,$$

where $\omega(u,v) = \infty$ *for* $(u,v) \in E(\omega') \setminus E(\omega)$.

Note that the axioms in Definition 8.3.1 and Definition 8.3.4 are essentially the same. The complex rule for consistency is due to the fact that the relations in Ω can have different domains. This implies the additional constraints. Also the set D of all edge weightings (of a complete graph) is a subset of Ω. The name *graph clustering function* is inspired by the fact that X and $\omega \in \Omega$ can be seen as a graph: $G_\omega = (X, E(\omega), \omega)$. Thus, the graph clustering function f maps the set of (weighted and simple) graphs with n elements to the set of all partitions over n elements.

Lemma 8.3.5. *The function* f_{comp} *that maps* $\omega \in \Omega$ *to the clustering where the clusters are the connected components of* G_ω *is a graph clustering function.*

Proof. It is sufficient to check the axioms since f_{comp} has the correct domain and codomain. Scale-invariance and consistency are fulfilled. This is due to the fact that for every relation $\omega \in \Omega$, the clustering $f_{\text{comp}}(\omega)$ has no inter-cluster edges. Although additional edges can be introduced via the axiom consistency, these edges cannot be inter-cluster edges with respect to $f_{\text{comp}}(\omega)$. The axiom richness is also satisfied because, for every clustering $\mathcal{C}$, there exists a spanning forest F with the same connected components as the clusters. Its edge relation E_F induces a weighted relation ω via $\omega(u,v) = 1$ if $(u,v) \in E_F$. The relation ω will be mapped to $\mathcal{C}$ by f_{comp}. □

Definition 8.3.4 and Lemma 8.3.5 clearly indicate that the lack of information indeed can be more explanatory than complete information. The presented graph clustering function does not provide further insight, but shows the potential of missing information. Alternatively to graph clustering functions, the axiom consistency may be redefined in various ways: restricted to a cluster or the connection between clusters, homogenous scalings, or predetermined breaking/merging points for controlled splits/linkages. These ideas are mostly unexplored, but have a certain potential.

A closing remark: it would also be interesting to know if there exist axiom systems that describe, or even characterize, already used frameworks. As mentioned previously, Single Linkage (with certain selection criteria) fulfills any two axioms, therefore it may be possible to extend current frameworks with the help of axioms. This might also be very valuable in identifying fundamental clustering concepts.

8.4 Chapter Notes

In this chapter, graph clustering has been introduced that was based on cohesion. The main objective of clustering is to find 'natural' decompositions of graphs. In our case, partitions of the node set modeled the clustering, and the paradigm of intra-cluster density versus inter-cluster sparsity had to be respected.

Both the term natural decomposition and the considered paradigm define clusterings only in an intuitive way, without mathematical parameters. To capture the informal notion more formally, structural indices were used that rate partitions with respect to clustering paradigms (see Section 8.1). Another benefit of these indices is that clusterings of different graphs can be compared, which is required for benchmark applications and to handle graphs with uncertainties. Most optimization problems related to indices, i.e., finding a non-trivial clustering with best index score, are $\mathcal{NP}$-hard.

In the first part of Section 8.2.1, generic frameworks were presented. They include simple methods like iteratively merging or splitting clusters, shifting concepts, and general optimization techniques. The frameworks are very flexible and can integrate a large number of problem-specific requirements. The next part dealt with the frameworks' parameters. It is still a high level view, and can be applied to most of the clustering problems. Some extensions with relatively similar methods concluded the section.

Alternative approaches for the representation of clusterings, as well as the methods to find suitable clusterings, were covered in the final section. First, the data structure partition was either enhanced with additional information or replaced by other decomposition types. Also, an axiom system was introduced to characterize clustering methods. In contrast to Section 8.2.2, where techniques were distinguished according to their different underlying ideas, the axiomatic framework describes the common foundation. Kleinberg's axioms were discussed in detail.

Owing to the large variety of applications for graph clustering, a complete and homogenous framework has not yet been established. However, several basic clustering techniques are fairly well understood, both from the informal and the mathematical points of view. The most difficult aspect is still the formalization of the 'natural decomposition', and, therefore, a mathematical framework for qualitatively good clusterings. Indices present a suitable method.

Alternative approaches to graph decompositions that are not based on density, but rather on structural properties, are presented in Chapters 9 and 10.