Data Min Knowl Disc (2013) 27:344–371
DOI 10.1007/s10618-013-0311-4

A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies

R. J. G. B. Campello · D. Moulavi · A. Zimek · J. Sander

Received: 3 October 2012 / Accepted: 21 March 2013 / Published online: 4 April 2013

Abstract We introduce a framework for the optimal extraction of flat clusterings from local cuts through cluster hierarchies. The extraction of a flat clustering from a cluster tree is formulated as an optimization problem and a linear complexity algorithm is presented that provides the globally optimal solution to this problem in semi-supervised as well as in unsupervised scenarios. A collection of experiments is presented involving clustering hierarchies of different natures, a variety of real data sets, and comparisons with specialized methods from the literature.

Keywords Hierarchical clustering · Optimal selection of clusters · Should-link and should-not-link constraints · Cluster quality

1 Introduction

One of the primary data mining tasks is cluster analysis, in which one aims at determining a finite set of categories to describe a data set according to similarities or

Responsible editor: Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, Filip Zelezny.

R. J. G. B. Campello (✉) · D. Moulavi · A. Zimek · J. Sander
Department of Computing Sciences, University of Alberta, Edmonton, AB, Canada
e-mail: rcampell@ualberta.ca

D. Moulavi
e-mail: moulavi@ualberta.ca

A. Zimek
e-mail: zimek@ualberta.ca

J. Sander
e-mail: jsander@ualberta.ca

R. J. G. B. Campello
Department of Computer Sciences, University of São Paulo, São Carlos, Brazil
e-mail: campello@icmc.usp.br

relationships among its objects (Hartigan 1975; Everitt et al. 2001). This task is often a first and important step in analyzing data, understanding their properties, and preparing them for further analysis. An important paradigm in cluster analysis is hierarchical clustering, in which a clustering solution is represented as a tree describing hierarchical relationships between nested clusters (Jain and Dubes 1988; Everitt et al. 2001). This paradigm has been of particular interest in a large variety of application areas. The reasons are manifold. First, it is a property of real data sets that clusters may be nested. Typically, nested clusters are characterized by having different densities, which is essentially the rationale behind Hartigan's classic definition of *density–contour trees* (Hartigan 1975) as a conceptual hierarchical model for data clustering. In addition, in some areas, such as biology, domain experts may prefer tree representations, as they are more used to them. In fact, sometimes the underlying application domain is hierarchically structured in its nature, as it is usually the case in areas such as biological taxonomy and document categorization, just to mention a few (Zhao and Karypis 2005). In conceptual clustering, for example, the hierarchical structure can also provide an attribute-value description that allows interpretation of the resulting clusters (Blockeel et al. 1998; Struyf and Džeroski 2007). Furthermore, hierarchical models are useful tools for visualization of high-dimensional data, e.g., in the form of a traditional dendrogram (Jain and Dubes 1988; Everitt et al. 2001), a compacted cluster tree (Sander et al. 2003; Stuetzle and Nugent 2010), a reachability-like plot (Ankerst et al. 1999; Brecheisen et al. 2004), or a silhouette-like plot (Gupta et al. 2010). For all these reasons, hierarchical models often represent a natural choice to describe clustering structures.

Hierarchical models are able to provide richer descriptions of clustering structures than those provided by "flat" models, in which a given label (possibly null, representing "noise") is assigned to every object of the data set. In spite of this, applications in which the user also (or even only) needs a flat solution are common, either for further manual analysis by a domain expert or in automated KDD processes in which the output of a clustering algorithm is the input of a subsequent data mining procedure—e.g., pattern recognition based on image segmentation. In this context, the extraction of a flat clustering from a hierarchy may be advantageous when compared to the extraction directly from data by a partitioning-like (i.e. non-hierarchical) algorithm. One reason is that hierarchical models describe data from multiple levels of specificity/generality, providing a means for exploration of multiple possible solutions from different perspectives while having a global picture of the cluster structure available. For example, as we will discuss later, for a multitude of types of hierarchies we can effectively evaluate the quality of clusters according to their behavior (e.g. stability) along different hierarchical levels.

The usual approach to extract a flat solution from a hierarchical clustering is by (manually or automatically) choosing one of the levels of the hierarchy. Choosing the most appropriate level is the well-known problem of performing a horizontal cut through a dendrogram, which has been widely studied in classic cluster analysis (Milligan and Cooper 1985; Jain and Dubes 1988). In spite of its widespread use in practice, this approach has a major limitation as it cannot provide

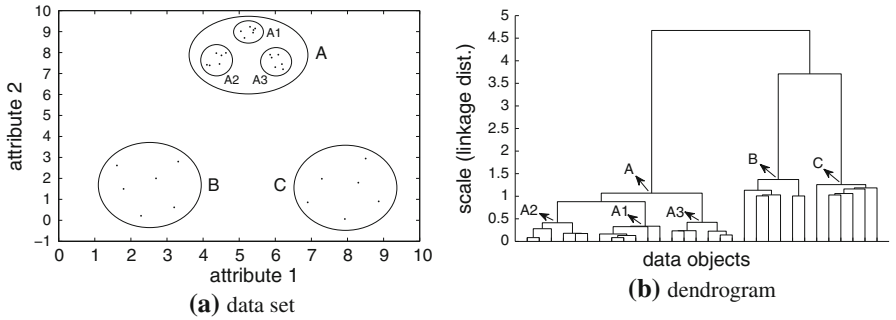


Fig. 1 **a** Illustrative data set and **b** its single-linkage dendrogram (Euclidean distance)

solutions composed of clusters described at different levels of abstraction or granularity (Boudaillier and Hébrail 1997, 1998). The toy example in Fig. 1 illustrates one possible consequence of such a limitation. Specifically, it is clear that there is no horizontal cut through the single-linkage dendrogram in Fig. 1b that would be able to simultaneously provide clusters $A1$, $A2$, $A3$, B, and C as a flat clustering solution for the data set in Fig. 1a, even though this is a discernibly valid alternative.

Situations like the one described above and illustrated in Fig. 1 can also occur when considering hierarchical clustering algorithms other than single-linkage (Kettenring 2006). In particular, in the case of hierarchical algorithms based on density estimates, such as OPTICS (Ankerst et al. 1999; Sander et al. 2003), AUTO-HDS (Gupta et al. 2006, 2010), and gSkeletonClu (Sun et al. 2010), a horizontal cut corresponds to a clustering solution induced by a single, global density threshold. As has been discussed in the literature, it is often not possible to simultaneously detect clusters of varied densities by using this kind of threshold (Ankerst et al. 1999; Stuetzle 2003; Tan et al. 2006; Kriegel et al. 2011), which is also a major shortcoming of many density-based non-hierarchical algorithms, such as DBSCAN (Ester et al. 1996) and DENCLUE (Hinneburg and Keim 1998), among others. Devising appropriate means to perform local cuts at different hierarchical levels for different branches (subtrees) of a cluster tree or dendrogram is therefore an important problem.

Performing local cuts through a clustering hierarchy is equivalent to adopting different granularity or density thresholds for different subsets of the data and allows one to get flat solutions that cannot be obtained by a traditional horizontal, global cut. Although there is a plethora of different methods for hierarchical clustering, only very few are (directly or indirectly) able to automatically perform some sort of local cut in the resulting hierarchy (Sander et al. 2003; Stuetzle 2003; Ferraretti et al. 2009; Stuetzle and Nugent 2010; Gupta et al. 2010), and they essentially do this in a fully unsupervised way.

Apart from unsupervised approaches, there has been a growing interest in semi-supervised hierarchical clustering methods, i.e., methods that produce cluster trees using partial supervision in the form of labels or constraints which represent previous knowledge about the data (e.g. Klein et al. 2002; Struyf and Džeroski 2007; Bade and Nürnberger 2008; Davidson and Ravi 2009; Zheng and Li 2011; Gilpin

and Davidson 2011).[1] The area of semi-supervised clustering has earned particular attention in recent years (Basu et al. 2008) and the branch of hierarchical approaches has followed a similar trend. Formulations of the problem have been discussed from a theoretical perspective (Davidson and Ravi 2005, 2009) and semi-supervised hierarchical clustering algorithms have been developed to deal with constraints in different ways. These algorithms are mostly based on some form of distance learning (Kim and Lee 2002; Klein et al. 2002; Bade and Nürnberger 2006, 2008; Bade et al. 2007; Zheng and Li 2011) or on the adaptation of agglomerative (Davidson and Ravi 2005, 2009; Gilpin and Davidson 2011; Hamasuna et al. 2012) and divisive (Kestler et al. 2006; Kraus et al. 2007; Xiong et al. 2011) hierarchical methods towards enforcing constraint satisfaction during the construction of the cluster tree. In spite of these advances, the focus has been on constructing hierarchies only, by satisfying constraints completely or as much as possible. To the best of our knowledge, the problem of selecting clusters to compose a flat solution from a cluster tree, when such type of solution is needed or desired, has been virtually untouched in the semi-supervised clustering literature. The only option is the traditional horizontal cut approach, which imposes an arbitrary and unnecessary additional constraint to the problem. In fact, by requiring that all extracted clusters must lie on the same level of the hierarchy, one would never be able to fully satisfy user-specified constraints that suggest the existence of $A1$, $A2$, $A3$, B, and C as clusters in the example of Fig. 1.

This paper is mainly intended to contribute towards filling the existing gaps in the literature in regard to the extraction of clusters from cluster trees. More specifically, the main contributions are:

- We formulate the problem of extracting a flat solution from local (non-horizontal) cuts through a generic cluster tree as an optimization problem.
- To solve this problem, we present an algorithm which is optimal in that, if an external collection of instance-level constraints of the type *should-link* and *should-not-link* is provided (semi-supervised case), the resulting solution globally maximizes the fraction of constraint satisfactions or, equivalently, it minimizes the fraction of constraint violations. This algorithm can in principle be applied to any cluster tree in which the clusters satisfy/violate the constraints to different degrees, because either (i) the tree has been constructed taking into account the constraints as soft (rather than hard) constraints, or (ii) it has been constructed not taking into account the constraints at all (i.e., in an unsupervised way).
- We present an unsupervised version of the algorithm that uses as an objective function an overall aggregation of the individual qualities of the composing clusters, measured by some local measure of cluster quality. This version can be applied to cluster trees in which clusters do not violate constraints, either because (i) there are no constraints at all (unsupervised case), or because (ii) the tree has been constructed by forcing all the existing (hard) constraints not to be violated by any cluster, which is sometimes possible under certain assumptions (e.g., see Gilpin and Davidson 2011).

[1] These methods should be distinguished from those that use hierarchical clustering and partially labeled data to categorize unlabeled data into *predefined categories* (semi-supervised categorization; e.g., see Skarmeta et al. 2000; Benkhalifa et al. 2001).

– We also discuss two different ways of integrating the unsupervised and semi-supervised versions into a single algorithm; namely: (i) unsupervised cluster quality can be used as a secondary objective to solve ties involving candidate clusters satisfying/violating the same fraction of constraints (the primary objective function); or (ii) the relative importance of these two objectives can be arbitrarily balanced by means of a convex combination of the corresponding objective functions. Unlike most non-hierarchical algorithms—which attempt at directly producing a flat clustering from the data—our method provides not only a globally optimal solution w.r.t. the criteria it optimizes, but also the number of clusters as a by-product, rather than as an explicit or implicit input parameter.
– A theoretical asymptotic analysis is provided which shows that the proposed method is scalable, having linear computational complexity w.r.t. both the number of constraints and the number of clusters in the cluster tree to be processed.
– A collection of experiments is presented that involves clustering hierarchies produced by different methods, from a variety of real data sets.

The remainder of this paper is organized as follows. In Sect. 2 we define and formulate the optimization problem of interest. In Sect. 3 we present a globally optimal solution to this problem along with an algorithm that can efficiently compute it in the semi-supervised, unsupervised, or mixed scenarios. In Sect. 4 we review the related work. In Sect. 5 we report experiments involving a variety of real data sets and clustering methods. At last, some final remarks and perspectives for future work are addressed in Sect. 6.

2 Problem statement

2.1 Preliminaries

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a data set containing n data objects and let $\{\mathbf{C}_1, \ldots, \mathbf{C}_\kappa\}$ be the collection of clusters in a clustering hierarchy of $\mathbf{X}$ from which a flat solution has to be extracted. We represent the clustering hierarchy as a cluster tree in which each node is associated with a particular cluster. Let $\mathbf{C}_1$ be the root of the tree, which represents the all-inclusive "cluster" composed of the entire data set (i.e. $\mathbf{C}_1 = \mathbf{X}$). For simplicity and without any loss of generality, we consider that the cluster tree is binary, for two main reasons: (i) it is simpler to formalize the problem for binary trees, which is actually the type of tree produced by most hierarchical clustering algorithms; and (ii) the solution of the problem can be straightforwardly generalized to deal with non-binary trees. Formally, the tree is such that each internal node, associated with a given cluster $\mathbf{C}_i$, has two children nodes associated with nested sub-clusters of $\mathbf{C}_i$. These nodes are denoted as $\mathbf{C}_{i_l}$ and $\mathbf{C}_{i_r}$, in reference to the left and right child of $\mathbf{C}_i$, respectively. The leaf nodes of the tree are associated with clusters that do not have sub-clusters. As we will see later, leaf nodes are not necessarily associated with singletons.

2.1.1 Problem overview

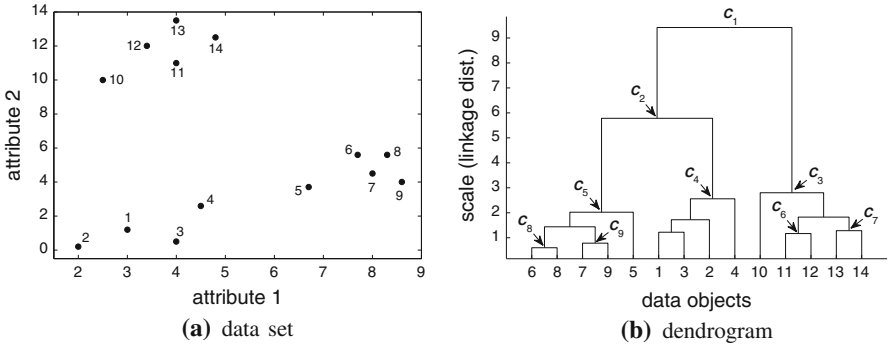
Before running into peculiarities and possible complications associated with the different aspects of clustering that will be considered in this paper, we first provide the reader with a preliminary, more abstract overview of the problem and outline the basic solution strategy that will be further elaborated in Sect. 3. Let us assume that we are given a cluster tree as described above as well as a numerical value associated with each cluster, which quantifies a certain property of interest related to the clusters. The fundamental problem we want to solve is to select a collection of clusters from the tree that represents a flat clustering solution and for which a predefined form of aggregation of the corresponding numerical values is maximized. For instance, if the meaningful aggregation for the problem at hand is the sum, then we want to maximize the sum of the numerical values associated with the selected clusters subject to the constraint that these clusters cannot overlap. Possible meanings for such numerical values are not relevant at this point.

As an example, consider the cluster tree illustrated in Fig. 3a. To get a flat solution, no other cluster can be selected in the subtree rooted at a given selected cluster. Among the solutions that satisfy this condition, the one composed of $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$ can be shown to be optimal in the sense that it maximizes the sum of the values displayed beside the clusters in the figure. To find such a solution efficiently, it is worth noticing that the sub-selection of clusters in any subtree represents a sub-problem of the very same nature of the original problem (i.e., the one that refers to the complete tree). From this observation, a dynamic programming strategy can be applied that incrementally solves sub-problems (subtrees) of increasing sizes, starting from the leaves and aggregating the intermediate solutions upwards in the tree. At this very high level of abstraction, this strategy is analogous to the one used in error based pruning of decision-tree classifiers.

In the example of Fig. 3a, starting from the deepest leaves, it can be seen that $\mathbf{C}_8$ and $\mathbf{C}_9$ must be discarded as their values sum up to 2.9 and, therefore, they are worse than $\mathbf{C}_5$ (21.1). At the level above, $\mathbf{C}_5$, as the best sub-selection in its subtree, and $\mathbf{C}_4$, as a subtree on its own, are together (37.5) better than $\mathbf{C}_2$ (32.7), which is then discarded. In the other branch, clusters $\mathbf{C}_6$ and $\mathbf{C}_7$ are also discarded as they are together (2.4) worse than $\mathbf{C}_3$ (36.9). In the end, clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$ remain, which is the optimal global solution in this example, with a total aggregated value of 74.4. Further in this paper, this basic solution strategy will be specialized, extended, and formalized as an algorithm for applications in different contexts of clustering.

2.1.2 Hierarchies with noise objects

It is worth remarking that many clustering hierarchies are not only able to represent parent-child relationships involving groups of objects with valid cluster labels, but they can also model the fact that some objects in a given parent cluster may become noise and be given a null label before this cluster is split into its children (from a top-down perspective of the cluster tree), thus no longer belonging to any cluster below the respective hierarchical level. Noise objects are a natural consequence when using density-based hierarchical algorithms (e.g. Ankerst et al. 1999; Sander et al. 2003;



(a) data set **(b)** dendrogram

Fig. 2 **a** Illustrative data set and **b** its average-linkage dendrogram (Euclidean distance)

Gupta et al. 2010), as such algorithms produce hierarchies in which the hierarchical levels are associated with different density thresholds, and objects whose density is below the threshold are deemed noise at the corresponding levels. The concept of noise, however, can be easily extended to other types of hierarchies, for instance, by using a *minimum cluster size*. In fact, in many practical applications of clustering (not just hierarchical clustering) a minimum cluster size is used as a user-specified parameter in order to prevent algorithms from finding very small clusters of objects that may be highly similar to each other just by chance, i.e., as a consequence of the natural randomness associated with the use of a finite data sample. Requiring a minimum cluster size, $m_{clSize} \geq 1$, allows only clusters with at least m_{clSize} objects to be reported, and the case where a subcomponent with fewer than m_{clSize} objects is disconnected from a cluster top-down along the hierarchy can be treated as not being a "true" split of that cluster, but as a simple reduction in its size.

As an example, Fig. 2 shows a simple data set and its average-linkage dendrogram, where there are 27 clusters (including 14 singletons), 9 of which have been highlighted. Table 1 shows the application of the idea described above to the dendrogram in Fig. 2b, with $m_{clSize} = 2$. When a single object—as a subcomponent with fewer than m_{clSize} objects—is disconnected from a cluster top-down along the hierarchy (which reads from left to right along the table), the original cluster is just regarded as having reduced in size, so it keeps its original label. This procedure reduces the 27 clusters in the dendrogram to 9 more prominent ones (labeled "1" to "9" in Table 1)[2], which however may exhibit different configurations along different hierarchical levels (e.g., cluster "4" with and without objects $\mathbf{x}_4$ and $\mathbf{x}_2$). In our framework, such different configurations can be modeled as parent-child nodes in the tree of candidate clusters, which in this case would not be strictly binary. Optionally, the cluster tree and the corresponding cluster extraction problem can be simplified by considering as significant hierarchical levels only those in which a cluster is truly split, giving rise to new sub-clusters. The clusters at such hierarchical levels are those highlighted in Fig. 2b ($\mathbf{C}_1, \ldots, \mathbf{C}_9$) and refer back to Hartigan's concept of *rigid clusters* (Hartigan 1975), which is adopted in this paper and has also been explored, e.g., by Herbin et al. (2001) and Gupta et al.

[2] Note that such a reduction may be even more noticeable for higher values of m_{clSize}.

Table 1 Hierarchy for the data in Fig. 2a with $m_{clSize} = 2$

$\mathbf{x}_1$	1	2	4	4	4	4	4	4	4	4	0	0	0	0
$\mathbf{x}_2$	1	2	4	4	4	4	4	0	0	0	0	0	0	0
$\mathbf{x}_3$	1	2	4	4	4	4	4	4	4	4	0	0	0	0
$\mathbf{x}_4$	1	2	4	4	0	0	0	0	0	0	0	0	0	0
$\mathbf{x}_5$	1	2	5	5	5	0	0	0	0	0	0	0	0	0
$\mathbf{x}_6$	1	2	5	5	5	5	5	5	8	8	8	8	8	0
$\mathbf{x}_7$	1	2	5	5	5	5	5	5	9	9	9	9	0	0
$\mathbf{x}_8$	1	2	5	5	5	5	5	5	8	8	8	8	8	0
$\mathbf{x}_9$	1	2	5	5	5	5	5	5	9	9	9	9	0	0
$\mathbf{x}_{10}$	1	3	3	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{x}_{11}$	1	3	3	3	3	3	6	6	6	6	6	0	0	0
$\mathbf{x}_{12}$	1	3	3	3	3	3	6	6	6	6	6	0	0	0
$\mathbf{x}_{13}$	1	3	3	3	3	3	7	7	7	0	0	0	0	0
$\mathbf{x}_{14}$	1	3	3	3	3	3	7	7	7	0	0	0	0	0
Scale	9.42	5.78	2.8	2.56	2.02	1.83	1.72	1.44	1.28	1.22	1.17	0.78	0.6	0

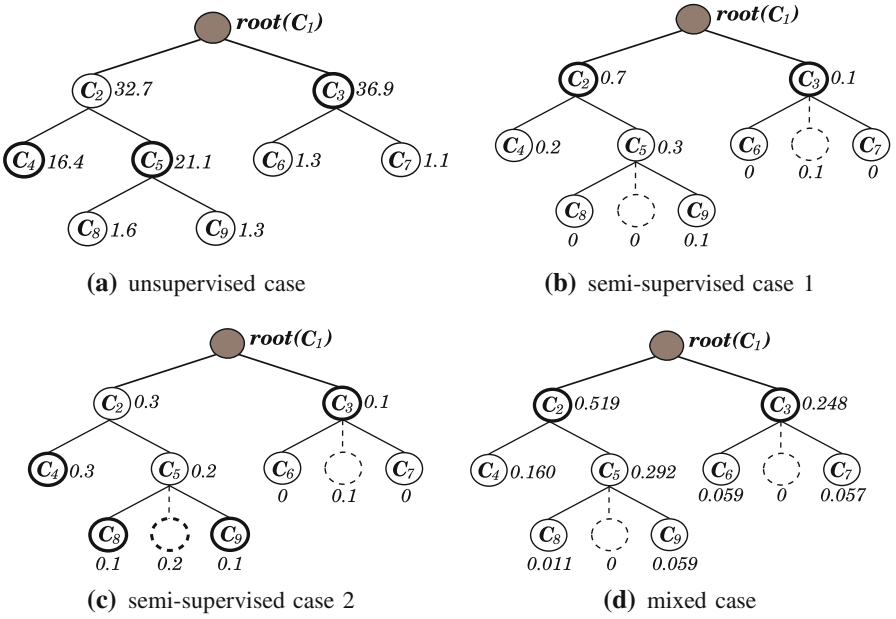
Higher (lower) hierarchical levels are on the left (right). Scale values (bottom bar) are the average-linkage distances. The remaining values are labels: a non-null value i in the jth row means that object $\mathbf{x}_j$ belongs to cluster $\mathbf{C}_i$ at the corresponding level, whereas a null value denotes noise

(2010). The corresponding cluster tree is displayed in Fig. 3a (the actual meaning of the values beside the nodes in that figure will be discussed later).

Note that, in principle, the cluster tree as represented in Fig. 3a does not explicitly account for the fact that object $\mathbf{x}_{10}$ of cluster $\mathbf{C}_3$ is not part of either of $\mathbf{C}_3$'s children, namely, $\mathbf{C}_6$ ($\mathbf{C}_{3_l}$) and $\mathbf{C}_7$ ($\mathbf{C}_{3_r}$), as $\mathbf{x}_{10}$ is already deemed noise at the level at which $\mathbf{C}_6$ and $\mathbf{C}_7$ appear as clusters and below. The same holds true w.r.t. object $\mathbf{x}_5$ and clusters $\mathbf{C}_5$, $\mathbf{C}_8$ ($\mathbf{C}_{5_l}$), and $\mathbf{C}_9$ ($\mathbf{C}_{5_r}$). We will see later that these objects can affect the optimal extraction of clusters in the semi-supervised case. For this reason and for the sake of generality, we assume that the information about objects that are part of a cluster yet not part of its sub-clusters (if any) is available in the hierarchy to be processed. This information can be represented in the cluster tree by means of *"virtual" nodes*, like the ones displayed in dashed lines in Fig. 3b–d.

In order to simplify the formulation of the problem, we assume that every internal node of the cluster tree has one virtual child node, even though only some of them are actually associated with noise objects. The remaining ones store no information at all and play no practical role for the algorithmic solution of the problem, though they simplify the notation used to describe it.[3] The virtual child node of a cluster $\mathbf{C}_i$ in the cluster tree will be denoted hereafter as $\mathbf{C}_i^{\emptyset}$, no matter whether it actually stores information or not.

[3] In the example of Fig. 3b–d, these nodes would be virtual children of $\mathbf{C}_1$ and $\mathbf{C}_2$, which have been omitted for the sake of clarity.



(a) unsupervised case **(b)** semi-supervised case 1

(c) semi-supervised case 2 **(d)** mixed case

Fig. 3 Simplified cluster trees corresponding to the hierarchy in Table 1. Figures beside the nodes denote: **a** cluster quality (measured by means of stability); **b**, **c** fractions of constraint satisfactions for different constraints; **d** convex combination of stability and fractions of constraint satisfactions. *Boldfaced solid nodes* indicate optimally selected clusters. *Dashed circles* in **b**–**d** represent virtual nodes associated with noise objects

2.2 Problem formulation as a mathematical programming task

It is assumed that we are given a cluster tree of a data set $\mathbf{X}$ as described in Sect. 2.1. It is also assumed that we are given a collection of constraints representing previous knowledge about the data set and being violated or satisfied to different degrees by the clusters in the cluster tree; the case where such a collection of constraints is empty will be discussed later as a particular, unsupervised extraction case. Recalling that the focus here is not on the construction of the hierarchy itself, but on the extraction of a flat clustering from it, we consider that the constraints used in the extraction stage are of the (instance-level) type *should-link* and *should-not-link*, which are those most commonly adopted in the broad context of semi-supervised clustering, and actually those most easily encountered in practical applications as well. Such constraints encourage pairs of objects to be grouped together in the same cluster or separated apart in different clusters (Wagstaff 2002).

Our primary objective is to extract a flat solution from the cluster tree that is globally optimal w.r.t. the constraints. Specifically, the objective is to extract a collection of clusters that altogether maximize the number of constraint satisfactions computed over the whole data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Unsupervised measures of cluster quality will in principle be considered as a secondary objective, to be discussed further. Formally, the primary objective function can be written as

$$J = \frac{1}{2n_c} \sum_{j=1}^{n} \gamma(\mathbf{x}_j), \tag{1}$$

where n_c is the number of available constraints (should- and should-not-link) and $\gamma(\mathbf{x}_j)$ is the number of constraints involving object $\mathbf{x}_j$ that are satisfied (not violated). The scaling constant $1/2$ is due to the fact that a single constraint involves a pair of objects and, as such, it is taken into account twice in the sum. Term n_c in the denominator is used to normalize J in the unit interval. Thus, J is the fraction of constraints that are satisfied or, equivalently, the complement of the fraction of constraints that are violated. Evidently, maximizing J is equivalent to minimizing the number of constraint violations.

In principle, the candidate clusters are all the clusters in the cluster tree. Whether the all-embracing "cluster" $\mathbf{C}_1$ (root) should be kept as a viable candidate or not is up to the user and does not change the essence of the problem or its solution. In the following, we consider that the user does require a partitioning of the data. So, we exclude $\mathbf{C}_1$ from the collection of eligible clusters, which is therefore $\mathbf{E} = \{\mathbf{C}_2, \ldots, \mathbf{C}_\kappa\}$.

When selecting clusters for a flat clustering solution, nested clusters in the tree must be mutually exclusive, i.e., each object can only be assigned a single label (possibly null, representing noise). This condition can be formulated as a set of constraints for the problem of maximizing J in Eq. (1). By noticing that terms $\gamma(\cdot)$ in Eq. (1) are a function of the clusters that will be selected, the optimization problem can be written as

$$\begin{array}{ll} \underset{\delta_2, \ldots, \delta_\kappa}{\text{maximize}} & J \text{ in Eq. (1)} \\ \text{subject to} & \begin{cases} \delta_i \in \{0, 1\}, & i = 2, \ldots, \kappa \\ \sum_{j \in \mathbf{I}_h} \delta_j = 1, & \forall h \text{ such that } \mathbf{C}_h \text{ is a leaf cluster,} \end{cases} \end{array} \tag{2}$$

where δ_i ($i = 2, \ldots, \kappa$) is an indicator that denotes whether cluster $\mathbf{C}_i$ is included in the flat solution ($\delta_i = 1$) or not ($\delta_i = 0$) and $\mathbf{I}_h$ is the set of indexes for those clusters on the path from leaf cluster $\mathbf{C}_h$ (included) to the root (excluded). The constraints in Problem (2) enforce that exactly one cluster will be selected along any branch from the root to a leaf. Accordingly, they ensure that each data object will be either assigned to a single cluster or labeled as noise for not being part of any selected cluster. It is worth remarking that, since a noise object is, by definition, not clustered with any other object, a should-not-link constraint involving one or both objects labeled as noise is deemed satisfied, while a should-link constraint is deemed violated.

Problem (2) is mathematically well-defined, but if the objects of a parent cluster are not involved in any constraints, then it is not possible to discriminate between this cluster and its sub-clusters in terms of constraint satisfiability. This is a particular case of a tie between nested (and therefore "competing") clusters, which can also occur in more general scenarios, even for clusters involved in constraints (as we will see in an example in Sect. 3.1). In these cases, there will be multiple valid cluster selections with the same globally optimal value for the objective function in Problem (2). In practice, this means that, instead of a unique flat clustering, one might return as a

result a clustering that is only partially flat, possibly including subtrees of nested clusters whose selection is undecidable. If the user does require a truly flat solution, the ties can be decided arbitrarily or a secondary objective must be considered to solve such subtrees. As a secondary objective, we consider an overall aggregation of the individual qualities of the composing clusters. Specifically, let $S(\mathbf{C}_i)$ be a given unsupervised measure of quality for cluster $\mathbf{C}_i$. Then, an overall aggregation of the qualities of those clusters selected from the cluster tree to compose a flat solution can be written as

$$J = \sum_{i=2}^{\kappa} \delta_i \, S(\mathbf{C}_i). \tag{3}$$

Using J given by Eq. (3) in lieu of Eq. (1) allows for resolving ties in subtrees.[4] By doing so, when the whole tree is indistinguishable w.r.t. constraint satisfiability (because it has been successfully constructed by enforcing that all clusters satisfy all the constraints or because there are no constraints at all), the problem falls into the particular, unsupervised case and clusters are purely extracted based on their qualities $S(\mathbf{C}_i)$.

2.3 Unsupervised measures of cluster quality

A suitable unsupervised measure of cluster quality depends on the characteristics of the cluster tree at hand. Cluster trees can be of varied natures and be produced by algorithms based on different paradigms. It is beyond the scope of this paper to discuss appropriate measures for cluster trees of all possible types. Instead, we will introduce a measure that can be applied to any algorithm that produces hierarchies of the same nature as that illustrated in Table 1. This includes, for instance, density-based methods (e.g. Sander et al. 2003; Gupta et al. 2006) and all the methods that produce traditional dendrograms as a result (e.g., see Jain and Dubes 1988; Everitt et al. 2001), and to which the hierarchy simplification technique based on minimum cluster size and noise labels described in Sect. 2.1 can be applied.

The measure considered here is based on the premise that more prominent clusters will survive longer after they appear, which is essentially the rationale behind the definition of *cluster lifetime* as a measure of cluster stability in classic cluster analysis (Jain and Dubes 1988). The lifetime of a given cluster in a traditional dendrogram is defined as the length of the dendrogram scale along those hierarchical levels in which that cluster exists. For hierarchies like the one in Table 1, this concept can be adjusted to account for the fact that objects of a cluster may have different lifetimes. The adjustment we consider here, which will be used in all the experiments reported in Sect. 5, is summing up the lifetimes of the objects belonging to the cluster. For example, from a bottom-up perspective of the hierarchy in Table 1, cluster $\mathbf{C}_4$ appears with 2 objects at level 1.22 of the scale and disappears when it is merged with $\mathbf{C}_5$,

[4] An alternative setting, in which both objectives in Eqs. (1) and (3) are combined into a single, mixed one, will be discussed further, in Sect. 3.2.

giving rise to $\mathbf{C}_2$ at level 5.78. In between these levels, a third object joins the cluster at level 1.72 and a fourth one joins it at level 2.56. Then, the stability of this cluster is given by $S(\mathbf{C}_4) = 2 * (5.78 - 1.22) + (5.78 - 1.72) + (5.78 - 2.56) = 16.4$. The values for the other clusters can be computed analogously and are displayed beside the nodes of the tree in Fig. 3a.

Cluster stability as defined above is endowed with two important properties that are required for a quality measure to qualify for use in the framework proposed in this paper. First, it is *local* in the sense that it can be computed for each cluster independently of the clusters that will be selected to compose the final flat solution, which are obviously unknown beforehand. Second, it is *additive* w.r.t. the objects that compose the cluster. This means that adding the values corresponding to the clusters to be selected is meaningful and compatible with the aggregation operator used in the objective function of the optimization problem, which is the sum operator in the case of Eq. (3).

3 Optimal cluster extraction

3.1 Problem solution and algorithm

Recalling that $\mathbf{E} = \{\mathbf{C}_2, \ldots, \mathbf{C}_\kappa\}$ is the whole collection of eligible clusters, let $\mathbf{F} \subseteq \mathbf{E}$ be any candidate flat clustering solution satisfying the structural constraints of Problem (2). Furthermore, let $\mathbf{X}_L \subseteq \mathbf{X}$ be the subset of data objects that have a non-null label in such a candidate flat solution, i.e., $\mathbf{X}_L = \{\mathbf{x}_j \mid \exists\, \mathbf{C}_i \in \mathbf{F} \colon\ \mathbf{x}_j \in \mathbf{C}_i\}$. Finally, let $\overline{\mathbf{X}}_L$ be the subset of data objects that have a null label (noise) in this candidate solution, i.e., $\overline{\mathbf{X}}_L = \mathbf{X} - \mathbf{X}_L$. Then, the objective function in Eq. (1) can be rewritten as

$$\begin{aligned} J &= \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \mathbf{X}_L} \gamma(\mathbf{x}_j) \;+\; \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \overline{\mathbf{X}}_L} \gamma(\mathbf{x}_j) \\ &= \sum_{i=2}^{\kappa} \delta_i \, \Gamma(\mathbf{C}_i) \;+\; \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \overline{\mathbf{X}}_L} \gamma(\mathbf{x}_j), \end{aligned} \tag{4}$$

where $\Gamma(\mathbf{C}_i) = \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \mathbf{C}_i} \gamma(\mathbf{x}_j)$ is the fraction of constraint satisfactions involving the objects of cluster $\mathbf{C}_i$ (which is zero for clusters whose objects are not involved in constraints). For example, let us consider three should-link constraints, $(\mathbf{x}_1, \mathbf{x}_6)$, $(\mathbf{x}_2, \mathbf{x}_5)$, and $(\mathbf{x}_5, \mathbf{x}_8)$, as well as two should-not-link constraints, $(\mathbf{x}_4, \mathbf{x}_9)$ and $(\mathbf{x}_3, \mathbf{x}_{10})$, for the data set in Fig. 2a. Given the clusters in Table 1, we can compute $\Gamma(\mathbf{C}_9) = \frac{1}{2\times 5} (\gamma(\mathbf{x}_7) + \gamma(\mathbf{x}_9)) = \frac{1}{10}(0 + 1) = 0.1$. The values for the other clusters can be computed analogously and are displayed beside the (non-virtual) nodes of the tree in Fig. 3b.

Similarly to the first term of Eq. (4), we can also decompose its second term, which refers to objects labeled as noise, into a sum of fractions, each of which is associated with a virtual node of the cluster tree. Every object labeled as noise in a flat solution extracted from the tree is associated with one of the virtual nodes. Hence, we can

rewrite Eq. (4) as[5]

$$J = \sum_{i=2}^{\kappa} \delta_i \, \Gamma(\mathbf{C}_i) + \sum_{l=1}^{m} \varphi_l \, \Gamma(\mathbf{C}_l^{\emptyset}), \tag{5}$$

where m is the number of virtual nodes in the tree ($m = \frac{\kappa-1}{2}$ for binary trees, i.e., the number of internal nodes), φ_l is an indicator that denotes whether the objects associated with the virtual node $\mathbf{C}_l^{\emptyset}$ (the virtual child of $\mathbf{C}_l$) are labeled as noise in the flat solution ($\varphi_l = 1$) or not ($\varphi_l = 0$), and $\Gamma(\mathbf{C}_l^{\emptyset}) = \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \mathbf{C}_l^{\emptyset}} \gamma(\mathbf{x}_j)$ is the fraction of constraint satisfactions involving the noise objects associated with $\mathbf{C}_l^{\emptyset}$. For instance, considering the same set of constraints as in the example above (and recalling that a should-not-link constraint involving one or both objects labeled as noise is deemed satisfied, while a should-link constraint is deemed violated), one has $\Gamma(\mathbf{C}_3^{\emptyset}) = \frac{1}{10}\gamma(\mathbf{x}_{10}) = 0.1$ and $\Gamma(\mathbf{C}_5^{\emptyset}) = \frac{1}{10}\gamma(\mathbf{x}_5) = 0$, which are the values displayed beside the respective virtual nodes in Fig. 3b. For virtual nodes not associated with any noise object at all (those omitted in Fig. 3b), $\Gamma(\mathbf{C}_l^{\emptyset})$ is defined as zero.

Notice that the objects associated with a virtual node $\mathbf{C}_l^{\emptyset}$ will end up being labeled as noise ($\varphi_l = 1$) if and only if any descendant of $\mathbf{C}_l$ is included into the final solution. This means that φ_l ($l = 1, \ldots, m$) is a function of the original decision variables δ_i of Problem (2), which can then be rewritten as

$$\begin{array}{ll} \text{maximize} & J \text{ in Eq. (5)} \\ \delta_2, \ldots, \delta_\kappa & \\ \text{subject to} & \left\{ \begin{array}{l} \text{the same constraints as in (2)} \\ \varphi_l = \left\{ \begin{array}{l} 1 \text{ if } l \in \mathbf{A} \\ 0 \text{ otherwise,} \end{array} \right. \end{array} \right. \end{array} \tag{6}$$

where $\mathbf{A} = \{l \mid \exists\, i \neq l : \delta_i = 1 \ \wedge \ \mathbf{C}_l \text{ is an ancestor of } \mathbf{C}_i\}$ are the indexes of clusters that are ancestors of any cluster $\mathbf{C}_i$ to be selected as part of the final flat solution ($\delta_i = 1$). In brief, Problem (6) formulates the maximization of the constraint satisfactions, written as a sum of fractions associated with the original and virtual nodes of the tree, provided that: (a) clusters located along a common branch of the tree (nested clusters) must be mutually exclusive; and (b) if a cluster is selected, then the objects associated with all virtual nodes of this cluster's ancestors must be labeled as noise.

To solve Problem (6), we process every cluster node except the root, starting from the leaves (bottom-up), deciding at each node $\mathbf{C}_i$ whether $\mathbf{C}_i$ or the best-so-far selection of nodes in $\mathbf{C}_i$'s subtrees should be selected. To be able to make this decision locally

[5] Notice that Γ has the same properties as S discussed in Sect. 2.3: it can be computed locally for each node and it is additive w.r.t. the objects in the node, so it is compatible with the aggregation operator (sum) used in the objective function in Eqs. (1) and (5).

Algorithm 1: Solution to Problem (6)

Input: Cluster tree and the fraction of constraint satisfactions for each node, $\Gamma(\cdot)$.
1. Initialize $\delta_2 = \cdots = \delta_\kappa = 1$.
2. Initialize the $\hat{\Gamma}$ values of the leaf nodes as $\hat{\Gamma}(\mathbf{C}_h) = \Gamma(\mathbf{C}_h)$.
3. For every internal node $\mathbf{C}_i$ (except the root $\mathbf{C}_1$), starting from the deepest levels and going up the tree, do:
   3.1 If $\Gamma(\mathbf{C}_i) < \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
      Set $\hat{\Gamma}(\mathbf{C}_i) = \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$ and remove $\mathbf{C}_i$ from the list of candidate clusters by setting $\delta_i = 0$.
   3.2 Else If $\Gamma(\mathbf{C}_i) > \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
      Set $\hat{\Gamma}(\mathbf{C}_i) = \Gamma(\mathbf{C}_i)$ and remove from the list of candidates (by setting their $\delta_{(\cdot)}$ values to 0) all the clusters in $\mathbf{C}_i$'s subtrees.
   3.3 Otherwise, resolve the tie by performing either the actions in Step 3.1 or those in Step 3.2, according to a secondary criterion.
4. **Return**: $\delta_2, \ldots, \delta_\kappa$.

at $\mathbf{C}_i$, we propagate and update the sum of constraint satisfactions $\hat{\Gamma}(\mathbf{C}_i)$ of nodes provisionally selected in the subtree rooted at $\mathbf{C}_i$ in the following, recursive way:

$$\hat{\Gamma}(\mathbf{C}_i) = \begin{cases} \Gamma(\mathbf{C}_i), & \text{if } \mathbf{C}_i \text{ is a (non-virtual) leaf node} \\ \max\{\Gamma(\mathbf{C}_i),\ \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})\}, & \text{if } \mathbf{C}_i \text{ is an internal node,} \end{cases} \tag{7}$$

where $\mathbf{C}_{i_l}$ and $\mathbf{C}_{i_r}$ are the sub-clusters of $\mathbf{C}_i$, while $\mathbf{C}_i^{\emptyset}$ is its virtual child.

Algorithm 1 gives the pseudo-code. Note that Step 3.3 does not specify a particular criterion to resolve ties. Different decisions when resolving a tie will lead to different flat solutions, but the (globally optimal) value of the primary objective function in Eq. (5) will necessarily be the same for any of these solutions. In other words, the choice of the secondary objective does not affect the optimality of the final solution w.r.t. the primary objective. Technically, therefore, in order to provide a single optimal solution to Problem (6), ties (if any) can be resolved arbitrarily. From a practical perspective, however, a more justified approach is to consider a suitable measure of cluster quality as a secondary objective. Specifically, a tie can be decided analogously to Steps 3.1–3.2, but replacing $\Gamma(\cdot)$ with an unsupervised measure of cluster quality $S(\cdot)$ (Sect. 2.3) and considering it to be null for virtual nodes ($S(\mathbf{C}_i^{\emptyset}) = 0$). Under these conditions, J in Eq. (5) becomes structurally equivalent to the unsupervised objective function given by Eq. (3). In this case, the solution of a sub-problem that refers to a subtree involved in a tie will therefore be optimal with respect to this secondary, unsupervised objective function. The detailed pseudo-code is given in Algorithm 2.

When there are no constraints, Algorithm 2 processes the whole tree in an unsupervised way, as in the example displayed in Fig. 3a. We have preliminary shown in Sect. 2.1.1 that the optimal solution in that example consists of clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$, with J in Eq. (3) equal to 74.4.

When the collection of five constraints previously used for illustration purposes in this section is considered, which refers to the example in Fig. 3b, the solution changes to $\mathbf{C}_2$ and $\mathbf{C}_3$, with J in Eq. (5) equal to 0.8 (i.e., 80 % of the constraints satisfied, 20 % violated). In details, $\mathbf{C}_8$ and $\mathbf{C}_9$ are discarded as they are together (and along with $\mathbf{C}_5^{\emptyset}$)

Algorithm 2: Solution to Problem (6) with quality-based tiebreaker

Input: Cluster tree, the fraction of constraint satisfactions for each node, $\Gamma(\cdot)$, and the unsupervised quality of each non-virtual node, $S(\cdot)$ (null for virtual nodes).

1. Initialize $\delta_2 = \cdots = \delta_\kappa = 1$.
2. Initialize $\hat{\Gamma}$ and $\hat{S}$ for the leaf nodes as $\hat{\Gamma}(\mathbf{C}_h) = \Gamma(\mathbf{C}_h)$ and $\hat{S}(\mathbf{C}_h) = S(\mathbf{C}_h)$.
3. For every internal node $\mathbf{C}_i$ (except the root $\mathbf{C}_1$), starting from the deepest levels and going up the tree, do:
   - 3.1 If $\Gamma(\mathbf{C}_i) < \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
     Set $\hat{\Gamma}(\mathbf{C}_i) = \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$ and remove $\mathbf{C}_i$ from the list of candidate clusters by setting $\delta_i = 0$. Also, set $\hat{S}(\mathbf{C}_i) = \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$.
   - 3.2 Else If $\Gamma(\mathbf{C}_i) > \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
     Set $\hat{\Gamma}(\mathbf{C}_i) = \Gamma(\mathbf{C}_i)$ and remove from the list of candidates (by setting their $\delta_{(\cdot)}$ values to 0) all the clusters in $\mathbf{C}_i$'s subtrees. Also, set $\hat{S}(\mathbf{C}_i) = S(\mathbf{C}_i)$.
   - 3.3 Otherwise, resolve the tie in an unsupervised manner:
     - 3.3.1 If $S(\mathbf{C}_i) < \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$, then do the same as if the condition in Step 3.1 had been satisfied.
     - 3.3.2 If $S(\mathbf{C}_i) >= \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$, then do the same as if the condition in Step 3.2 had been satisfied.
4. **Return**: $\delta_2, \ldots, \delta_\kappa$.

worse than $\mathbf{C}_5$. At the level above, the pair $\mathbf{C}_4$ and $\mathbf{C}_5$ is also discarded as $\mathbf{C}_2$ is better. In the other subtree of the root, there is a tie w.r.t. constraint satisfiability: either $\mathbf{C}_3$ or its children ($\mathbf{C}_6$, $\mathbf{C}_7$, and $\mathbf{C}_3^{\emptyset}$) would add the same amount of 0.1 to the objective function. A particular choice does not affect the global optimality of the final solution w.r.t. Eq. (5), but $\mathbf{C}_3$ is kept in this case because its unsupervised quality (36.9) is higher than that of $\mathbf{C}_6$ and $\mathbf{C}_7$ together (2.4), which are thence discarded. Note that, in what concerns the secondary objective in Eq. (3), this sub-selection is optimal for the subtree rooted at $\mathbf{C}_3$.

As an additional example, if we change, e.g., two of the should-link constraints to should-not-link ones, namely, $(\mathbf{x}_2, \mathbf{x}_5)$ and $(\mathbf{x}_5, \mathbf{x}_8)$, then we get the scenario illustrated in Fig. 3c. In this case, the optimal solution changes to $\mathbf{C}_3$, $\mathbf{C}_4$, $\mathbf{C}_8$, and $\mathbf{C}_9$, once again with $J = 0.8$ (which includes the fraction of 0.2 corresponding to the virtual child of $\mathbf{C}_5$).

3.2 Alternative formulation: single, mixed objective function

So far it has been assumed that, in the semi-supervised scenario, one wants to give priority to satisfy the available constraints and use an unsupervised measure of cluster quality solely to decide ties. Alternatively, one may prefer to balance the relative importances of these two objectives in the optimization problem. This can be undertaken in the proposed optimization framework by means of a convex combination of the corresponding objective functions, i.e., $J = \alpha J_U + (1 - \alpha) J_{SS}$, where J_U is the unsupervised objective function in Eq. (3) and J_{SS} is the semi-supervised one in Eq. (1). Based on the very same reasoning previously described in Sect. 3.1, all we need to do now is to store, in the nodes of the tree, the convex combination of the values of unsupervised quality and constraint satisfactions of the corresponding clusters,

i.e., $\Omega(\mathbf{C}_i) = \alpha S(\mathbf{C}_i) + (1-\alpha)\Gamma(\mathbf{C}_i)$. For virtual nodes, S is null and the value to be stored is therefore $\Omega(\mathbf{C}_i^{\emptyset}) = (1-\alpha)\Gamma(\mathbf{C}_i^{\emptyset})$. An auxiliary variable $\hat{\Omega}(\mathbf{C}_i)$ can then be recursively defined in the very same way as in Eq. (7), but replacing Γ and $\hat{\Gamma}$ with Ω and $\hat{\Omega}$, respectively. This way, Algorithm 2 can be directly applied by using Ω and $\hat{\Omega}$ in place of Γ and $\hat{\Gamma}$, respectively.

Note that the semi-supervised and unsupervised scenarios discussed in Sect. 3.1 now become special cases of this new, mixed one. On the one hand, by setting $\alpha = 1$ the values stored in the non-virtual nodes reduce to the measure of cluster quality, S, whereas the values in the virtual nodes become zero, which leads to the purely unsupervised scenario. On the other hand, by setting $\alpha = 0$ the values stored in all non-virtual and virtual nodes reduce to the corresponding fractions of constraint satisfactions, Γ, all of them multiplied by the same scalar term $1-\alpha$ (which, therefore, does not affect the solution), thus leading to the semi-supervised scenario originally discussed.

Intermediate values of α allow one to perform exploratory data analysis w.r.t. a trade-off between the relative importances of constraint satisfaction and unsupervised quality in the resulting clustering solution. As a weighting factor, the interpretation of this parameter is clearer if both objective functions are normalized within a common interval. In the case of J_{SS} and J_U, the former is by definition within $[0, 1]$, but the latter has no predefined upperbound. Nevertheless, it is possible to normalize J_U within $[0, 1]$ by dividing it by its maximum possible value for the problem at hand, J_U^*, which is the optimal solution to the purely unsupervised problem. Solving this particular problem a priori does not change the asymptotic complexity of the method. By doing so, the precomputed value of J_U^* can be used to normalize J_U, implicitly, when computing Ω, as $\Omega(\mathbf{C}_i) = \alpha \frac{S(\mathbf{C}_i)}{J_U^*} + (1-\alpha)\Gamma(\mathbf{C}_i)$.

Let us recall the example of the hierarchy in Table 1, whose cluster tree and the corresponding values of unsupervised cluster quality (stability) are displayed in Fig. 3a. We already know that the optimal value of the objective function for this example is $J_U^* = 74.4$ in the purely unsupervised case. If we now consider a collection of three should-link constraints, $(\mathbf{x}_1, \mathbf{x}_6)$, $(\mathbf{x}_2, \mathbf{x}_5)$, and $(\mathbf{x}_5, \mathbf{x}_8)$, as well as two should-not-link constraints, $(\mathbf{x}_4, \mathbf{x}_9)$ and $(\mathbf{x}_{12}, \mathbf{x}_{14})$, we have fractions of constraint satisfactions computed as $\Gamma(\mathbf{C}_2) = 0.6$, $\Gamma(\mathbf{C}_4) = \Gamma(\mathbf{C}_6) = \Gamma(\mathbf{C}_7) = \Gamma(\mathbf{C}_9) = 0.1$, and $\Gamma(\mathbf{C}_5) = 0.3$ (for the other nodes of the tree the value is zero). In this case, by computing the values of Ω as a function of α, it can be shown that we have three different possible solutions to the mixed optimization problem, depending on the value of α. Specifically, clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$ are selected for values of α greater than ≈ 0.756, while $\mathbf{C}_2$, $\mathbf{C}_6$, and $\mathbf{C}_7$ are selected for values of α smaller than ≈ 0.3. For intermediate values of α, a third solution composed of clusters $\mathbf{C}_2$ and $\mathbf{C}_3$ is obtained. The scenario for $\alpha = 0.5$ is illustrated in Fig. 3d.

3.3 Efficient implementation and asymptotic complexity analysis

Note that Step 3.2 of Algorithm 1 (or Algorithm 2) can be implemented in a more efficient way by not setting $\delta_{(\cdot)}$ values to 0 for discarded clusters down in the subtrees

(which could happen multiple times for nodes present in multiple subtrees of its ascendant nodes). Instead, in a simple post-processing procedure, the tree can be traversed top-down in order to find, for each branch, the shallowest cluster that has not been discarded ($\delta_{(\cdot)} = 1$) by Step 3.1. This way, the solution can be found with two traversals of the tree, one bottom-up and another one top-down. This means that the complexity of the algorithm, having as inputs the cluster tree and the values of $\Gamma(\cdot)$ and $S(\cdot)$ associated with its nodes, is $O(\kappa)$, i.e., *it is linear w.r.t. the number of nodes (or candidate clusters) in the tree, both in terms of running time and memory space*.

It is worth noticing that, if the cluster tree results from a hierarchy simplification procedure like the one described in Sect. 2.1, κ is typically much smaller than the number of data objects ($\kappa \ll n$). It is theoretically possible, however, that a cluster split is observed at each of the n hierarchical levels, leading to the worst case in which $\kappa = 2n - 1$, i.e., κ is $O(n)$.

The pre-computation of the input values $\Gamma(\cdot)$ and $S(\cdot)$ depends on the nature of the particular hierarchy at hand. $S(\cdot)$ also depends on the particular cluster quality measure to be adopted. For the type of hierarchy considered in the examples and experiments provided in this paper, which are dendrogram-like hierarchies possibly modeling noise, it follows that: (i) it is straightforward to compute $\Gamma(\cdot)$ associated with both the original and virtual nodes of the tree with a single pass through the hierarchical levels for each constraint, checking the labels of the pair of objects involved in that constraint. So, the complexity to compute $\Gamma(\cdot)$ for all nodes of the tree is $O(n_c\, n)$, where n_c is the number of constraints; and (ii) similarly, it is also straightforward to compute $S(\cdot)$ for all cluster nodes of the tree—as the stability measure described in Sect. 2.3—by means of a single bottom-up screening of the hierarchy, which is $O(n^2)$ (i.e., no greater than the complexity of computing the clustering hierarchy itself).

4 Related work

Apart from important research towards the semi-supervised *construction* of clustering hierarchies (Kim and Lee 2002; Klein et al. 2002; Davidson and Ravi 2005, 2009; Kestler et al. 2006; Bade and Nürnberger 2006, 2008; Bade et al. 2007; Kraus et al. 2007; Struyf and Džeroski 2007; Zhao and Qi 2010; Miyamoto and Terami 2010; Zheng and Li 2011; Xiong et al. 2011; Gilpin and Davidson 2011; Hamasuna et al. 2012; Ahmed et al. 2012), we are not aware of an approach in the literature for the semi-supervised *extraction* of clusters based on optimal cuts through a generic cluster tree. The algorithm SS-DBSCAN (Lelis and Sander 2009) uses semi-supervision in the form of partially labeled objects and implicitly provides as a result a collection of clusters that would be equivalent to local cuts through the OPTICS hierarchy (Ankerst et al. 1999) performed so as to force clusters' maximality and purity. SS-DBSCAN is an improvement over HISSCLU (Böhm and Plant 2008), which obtains clusters based on a simple horizontal cut through a related hierarchy and whose cut level (single density threshold) depends on a user-specified parameter. An underlying, restrictive assumption of SS-DBSCAN is that there should be at least one labeled object of each data category to be discovered as a cluster; the method cannot discover natural clusters whose objects are not involved in the partial information provided a priori by the user.

In the unsupervised domain, a few methods are able to explicitly or implicitly provide local cuts through some kind of clustering hierarchy. Boudaillier and Hébrail (1997, 1998) described an interactive tool for manual local cuts in traditional dendrograms based on exploratory visualization. In regard to automated methods, Ferraretti et al. (2009) proposed a greedy heuristic approach to iteratively select clusters to be split, top-down through a traditional dendrogram, attempting to improve the Dunn's validity index as a global measure of cluster quality; the remaining clusters are extracted as the final flat solution. Stuetzle (2003) and Stuetzle and Nugent (2010) proposed algorithms to detect clusters of spatial point sets as modes of continuous-valued density estimates. These algorithms are equivalent to performing local cuts through a single-linkage-like dendrogram, but in both cases the cuts are based on criteria that are critically dependent upon a user-specified threshold. Gupta et al. (2010) proposed a heuristic approach to extract a flat solution from density-based hierarchies produced by the HDS algorithm (Gupta et al. 2006), based on an iterative greedy procedure guided by the stabilities of the candidate clusters. Sander et al. (2003) presented two different proposals. The first one is an algorithm to transform an OPTICS reachability plot into an equivalent density-based dendrogram. The second one is a method to derive from those plots a compacted tree containing only significant clusters, which was claimed to be less sensitive to the user settings than the original method of Ankerst et al. (1999). However, the only (ad-hoc) approach to the problem of extracting a flat solution from local cuts in the cluster tree was to arbitrarily take all the leaf clusters and simply discard the others.

5 Experimental evaluation

5.1 Data sets

We used real data sets with a variety of characteristics (no. of objects, dimensionality, and no. of clusters) and from different domains, namely, biology, text, image, and UCI data sets. Two data sets, "Articles-5" and "Cbrilpirivson", consist of very high dimensional representations of text documents. They are formed by 253 and 945 articles represented by 4636 and 1431 dimensions, respectively, both with 5 classes. Articles-5 is made available upon request by Naldi et al. (2011) and Cbrilpirivson (Paulovich et al. 2008) is available at http://infoserver.lcad.icmc.usp.br/infovis2/PExDownload. We used the Cosine measure as dissimilarity function for these data sets. Two data sets, "CellCycle-237" [made public by Yeung et al. (2001)] and "YeastGalactose" [used by Yeung et al. (2003)] represent gene-expression data and contain 237 resp. 205 objects (genes), 17 resp. 20 dimensions (conditions), and 4 known classes. For these data sets we used Euclidean distance on the z-score normalized objects, which is equivalent to using Pearson correlation on the original data. Two data sets, "Wine" and "Ecoli", are from the UCI Repository (Frank and Asuncion 2010). They contain 178 resp. 336 objects in 13 resp. 7 dimensions, with 3 resp. 8 classes. For these data sets we used Euclidean distance.

In addition to individual data sets, we also report average performance on two data set collections based on the Amsterdam Library of Object Images (ALOI) (Geusebroek

et al. 2005). Image sets were created as in (Horta and Campello 2012) by randomly selecting k ALOI image categories as class labels 100 times for each $k = 2, 3, 4, 5$, then sampling (without replacement), each time, 25 images from each of the k selected categories, thus resulting in 400 sets, each of which contains 2, 3, 4, or 5 clusters and 50, 75, 100, or 125 images (objects). The images were represented using 6 different descriptors: color moments, texture statistics from the gray-level co-occurrence matrix, Sobel edge histogram, 1st order statistics from the gray-level histogram, gray-level run-length matrix features, and gray-level histogram, with 144, 88, 128, 5, 44, and 256 attributes, respectively. We report results for the texture statistics (as a typical case), denoted by "ALOI-TS88", and for a 6-dimensional representation combining the first principal component extracted from each of the 6 descriptors using PCA, denoted by "ALOI-PCA". We used Euclidean distance in both cases.

5.2 Evaluation measures

We report the Adjusted Rand Index (ARI) (Hubert and Arabie 1985) in all experiments, considering noise objects as singletons. We have also computed the Overall F-measure (Larsen and Aone 1999), but the results are omitted for the sake of compactness as the conclusions that can be drawn are precisely the same as for ARI. Both ARI and F-measure are measures commonly employed in the literature and they have been computed here *not* taking into account those data objects involved in the constraints used by the algorithms under evaluation.

5.3 Algorithms

Our general method, denoted here as "Framework for Optimal Selection of Clusters" (*FOSC*), is compared with the following specialized algorithms: (i) the greedy method by Ferraretti et al. (2009), designed for traditional dendrograms and denoted here as "ACS"; (ii) the heuristic method by Sander et al. (2003), referred to here as "OPTICS-AutoCl", which consists of the extraction of the leaf nodes of a compacted, density-based cluster tree from an OPTICS reachability plot; and (iii) Semi-Supervised DBSCAN (SS-DBSCAN) (Lelis and Sander 2009), which is also density-based.

5.4 Experimental settings

We performed experiments in three different scenarios. In the first scenario, we consider the extraction of clusters from density-based hierarchies. Specifically, FOSC has been applied to the equivalent dendrogram that can be constructed from an OPTICS reachability plot by using the transformation algorithm described by Sander et al. (2003). We refer to this approach as "FOSC-OPTICSDend". The results have been compared with those of SS-DBSCAN (semi-supervised) and OPTICS-AutoCl (unsupervised, as a baseline), which are both also based on OPTICS. In the second scenario, we consider traditional hierarchical clusterings. We report the results of FOSC when applied to average-linkage dendrograms and compare them with those obtained by

ACS (unsupervised). These approaches are referred to here as "FOSC-AvLink" and "ACS-AvLink", respectively. In this scenario, we are not aware of a semi-supervised method to compare with.

In the third scenario, we compare the results of FOSC when the framework is applied to dendrograms that are obtained in an unsupervised versus in a semi-supervised way. Specifically, we compare FOSC-AvLink described above for the second scenario against FOSC itself, but now applied to modified average-linkage dendrograms obtained in a semi-supervised way. The resulting approach is referred to here as "FOSC-SAT", because it takes as an input dendrograms that have been computed using the SAT-solver-based method described by Gilpin and Davidson (2011). Specifically, this method incorporates propositional logic solvers into the construction loop of agglomerative hierarchical algorithms and can cope with constraints of varied natures, including instance-level constraints of the types used here in this paper. It tries to find a dendrogram in which no cluster violates any constraints. However, when both (so-called global) should-link and should-not-link constraints take place at the same time, such a dendrogram does not exist. To overcome this drawback and perform our experiments, we have allowed constraints to be violated in the root of the cluster tree (i.e., in "cluster" $\mathbf{C}_1$), which is not deemed a valid candidate by FOSC anyway. In this case, the method of Gilpin and Davidson (2011) starts constructing the cluster tree by satisfying the should-link constraints at the bottom and goes upwards by iteratively performing the best legal join (according to the linkage policy and the available should-not-link constraints), until no more legal joins exist. At this point, the current clusters are finally merged into a single "cluster" at the root.

The methods based on OPTICS demand a parameter *MinPts*, which is a non-critical smoothing factor also used by other density-based clustering algorithms (e.g. Ester et al. 1996; Gupta et al. 2010; Sun et al. 2010). We set it to $MinPts = 4$ in all experiments, which is a value commonly used in the literature. The speed up control value ϵ in OPTICS was not used ($\epsilon =$ "infinity"). The parameters required by OPTICS-AutoCl and ACS were set as suggested by the authors. Finally, in all the experiments involving dendrograms, we applied the procedure in Sect. 2.1 with $m_{clSize} = MinPts = 4$, also as a smoothing factor. We also ran experiments for other values of m_{clSize} and *MinPts*, and the main conclusions do not change.

The partial information provided to the semi-supervised methods was obtained in the form of labeled objects randomly selected from the data sets. These objects were not considered when assessing the quality of the results. SS-DBSCAN uses the labels explicitly. The should-link and should-not-link constraints that can be derived from the labels are used by FOSC. We set the number of labeled objects to 0, 2, 4, 6, 8, and 10 % of the number of objects in each data set, where 0 corresponds to the particular, unsupervised case. In the other cases, the results reported are averages over 100 random selections of label sets. Note that ACS-AvLink and OPTICS-AutoCl are unsupervised methods, therefore their results do not change with the number of labeled objects available.

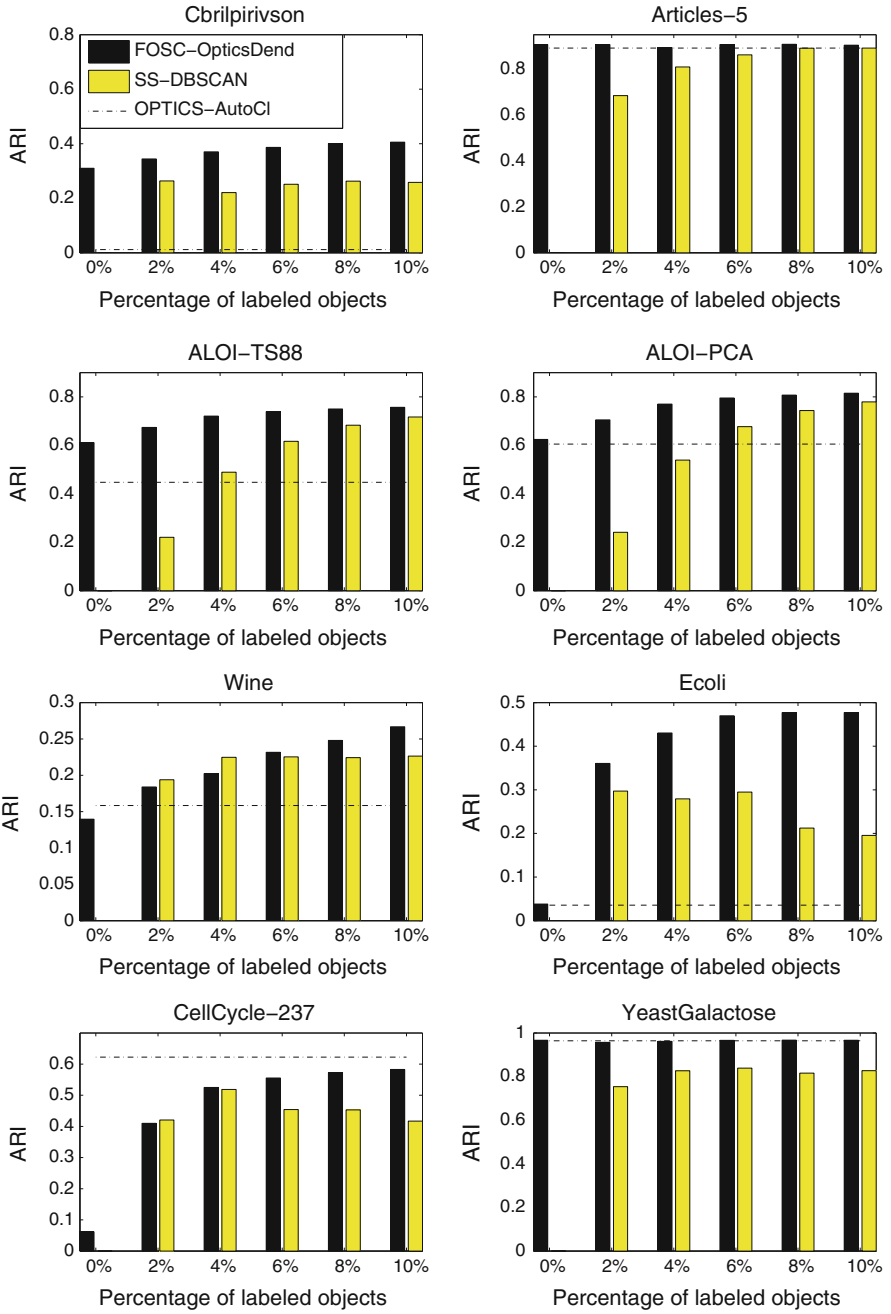


Fig. 4 ARI results for FOSC-OPTICSDend, SS-DBSCAN, and OPTICS-AutoCl methods. SS-DBSCAN is not shown when there are no objects labeled as this algorithm does not work in an unsupervised way

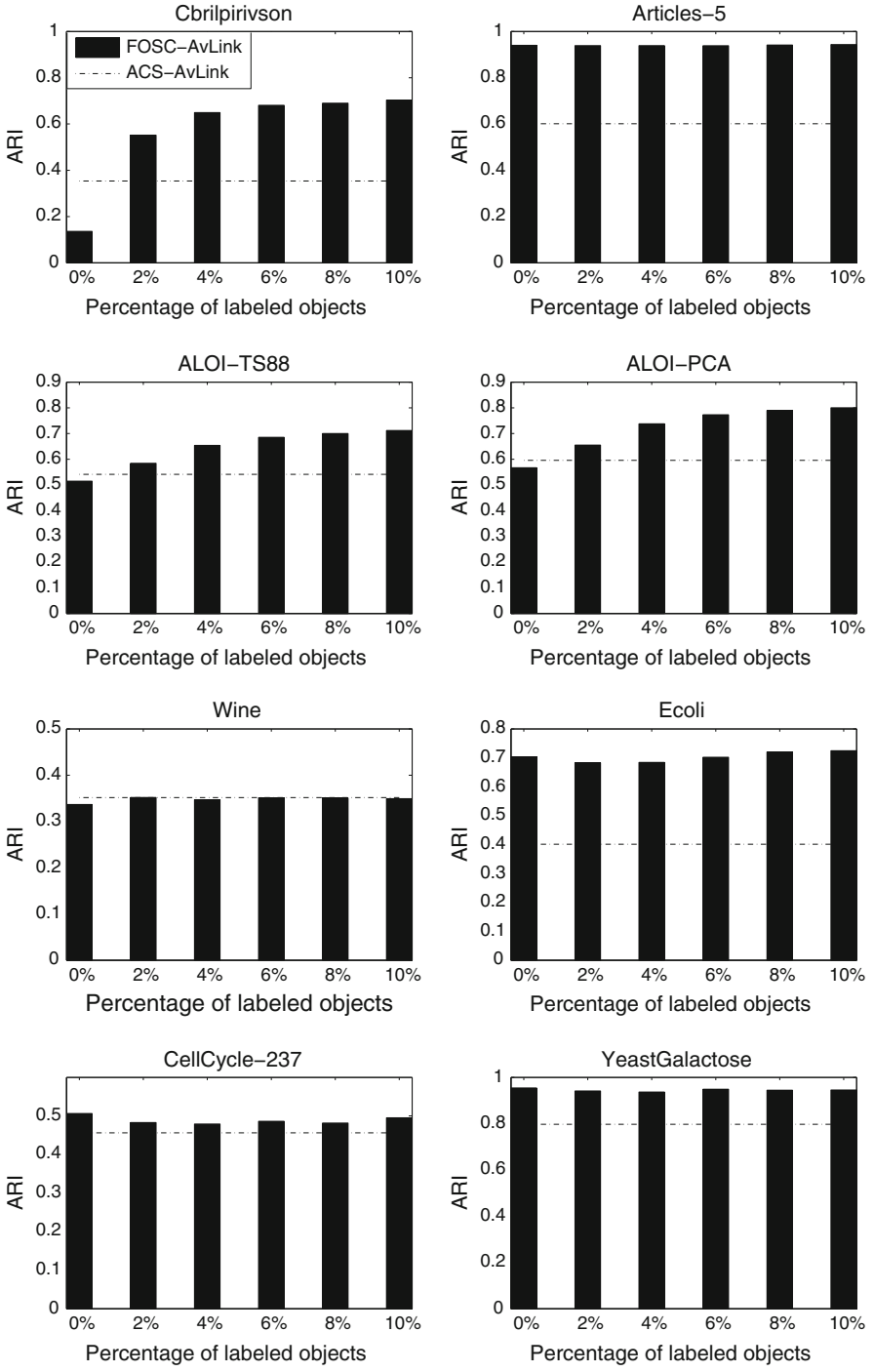


Fig. 5 ARI results for FOSC-AvLink and ACS-AvLink

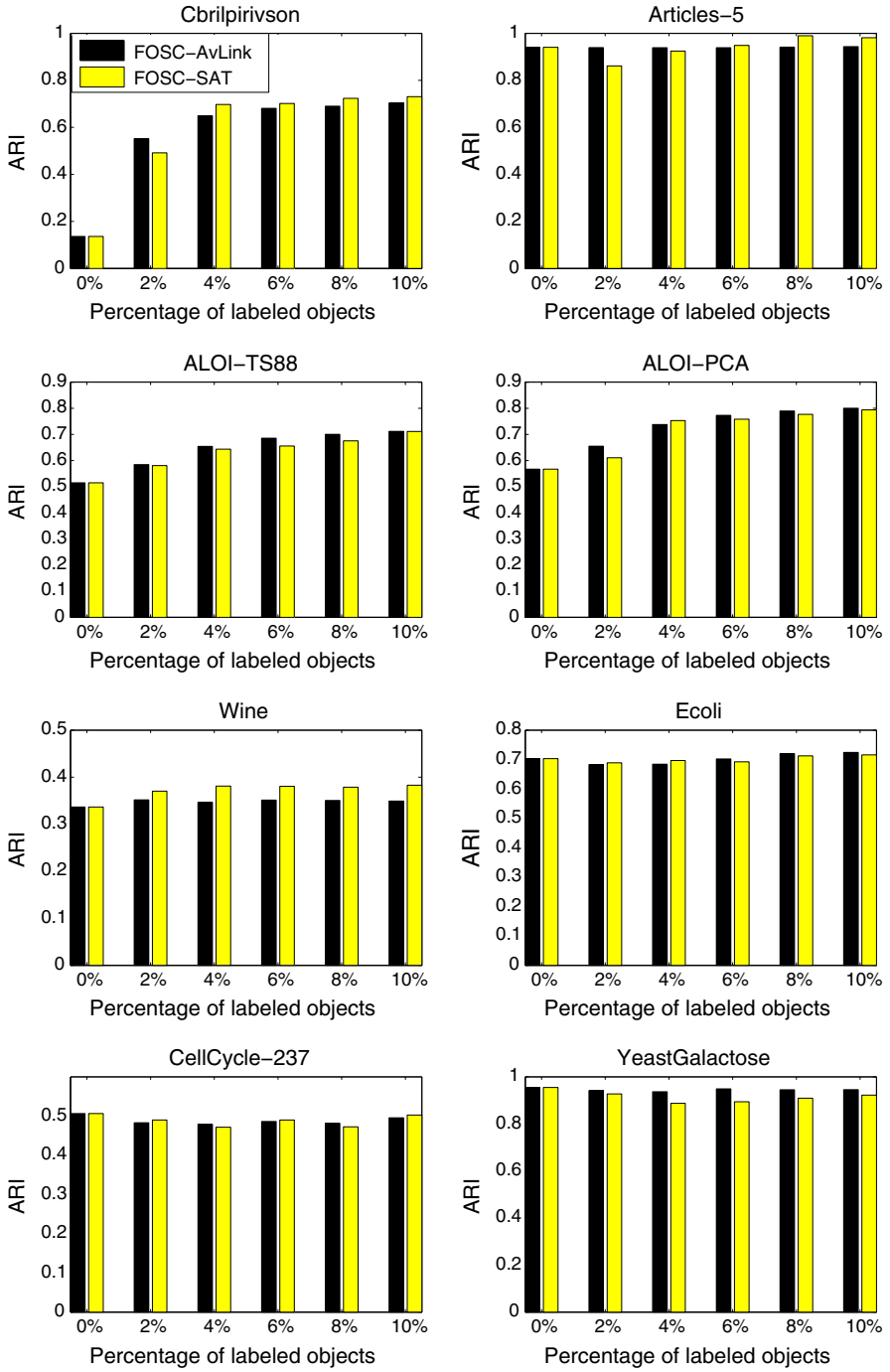


Fig. 6 ARI results for FOSC-AvLink and FOSC-SAT

5.5 Clustering results

The results are shown in Figs. 4, 5, and 6. When there are no labeled objects, FOSC operates in an unsupervised way based on cluster quality only. Notice from Figs. 4 and 5 that, even though FOSC is a general framework, its overall results in the unsupervised case are comparable to those of the specialized algorithms OPTICS-AutoCl and ACS-AvLink. In some cases, such as Articles-5 and YeastGalactose, the unsupervised performance leaves no much room for improvements, and the addition of constraints does not have effect. In many other cases, the performance of FOSC improves by adding constraints. This is to an extent that, as expected, FOSC outperforms the baseline, unsupervised methods in almost all cases involving labeled objects, often by a large margin (e.g. Cbrilpirivson and ALOI).

When compared with SS-DBSCAN, FOSC provides better results virtually in all cases, in many cases by a large margin, and more prominently with reduced amounts of labeled objects. Having the unsupervised selection method based on cluster quality as a secondary objective, which can help the algorithm in the absence of constraints, FOSC operates very well even with smaller amounts of labeled objects. Furthermore, FOSC improves faster than SS-DBSCAN as the number of labeled objects increases. For Articles-5, for example, it can be seen in Fig. 4 that SS-DBSCAN can only achieve the same quality as FOSC when it uses about 8% of labeled objects. For YeastGalactose, FOSC with no semi-supervision outperforms SS-DBSCAN even when the latter uses 10% of labeled objects.

In some of the data sets (e.g. Ecoli), the performance of SS-DBSCAN dropped when larger amounts of constraints were added. As a matter of fact, it has been observed and discussed in the semi-supervised clustering literature that adding constraints may possibly decrease the performance of clustering algorithms (Davidson et al. 2006). In our experiments, considering that we have excluded the objects involved in constraints when computing the evaluation measures, this is particularly plausible. In the case of FOSC, however, such a behavior is more rarely observed and is noticeably less pronounced.

In the experiments displayed in Figs. 4 and 5 the cluster trees provided as an input to FOSC have been obtained in an unsupervised way. In other words, external constraints have not been used in the first (preprocessing) stage in which the hierarchical clusterings are built from data. Instead, only *implicit* constraints following from the inductive biases of the respective clustering models (e.g. the concept of density-based clusters underlying the algorithms in Fig. 4) have been considered in this stage. When available, external constraints have only been used in the second stage in which clusters are extracted by FOSC. This particular approach can be interpreted as a strategy that gives priority to satisfy the implicit model constraints when constructing the cluster hierarchy and then uses constraints provided by the user as *preferences* (rather than hard requirements) to extract a flat solution. A different approach is to give priority to the external constraints when building the hierarchy in the first stage and, then, extract a flat solution in the second stage mostly based on unsupervised cluster quality. Whether or not a particular approach will be more appropriate than the other in a particular application scenario seems to depend on different factors, such as the data set, the quality of the constraints,

and the clustering model adopted. For example, when constructing the unsupervised average-linkage dendrograms used in the experiments displayed in Fig. 5, if an important should-not-link constraint is violated at the bottom then the mistake will not be fixed above in the tree. On the other hand, if such a constraint is one of those that could actually deteriorate the quality of the clustering solution, then forcing it to be satisfied might produce adverse results. The decision on which approach should be adopted in a given application depends on how the (model and external) constraints should be treated. This decision refers to the type of hierarchy that will be provided as an input to FOSC and, as such, it precedes the use of the proposed framework. It is not our intention in this paper to promote either approach. We will, however, show experiments that suggest that FOSC can be reasonably robust to this choice.

In Fig. 6 we contrast the results of FOSC-AvLink (as shown in Fig. 5) with those of FOSC-SAT, which, as previously described, refers to FOSC applied to average-linkage dendrograms obtained in a semi-supervised way. Notice that, apart from small differences observed in particular experimental settings (data sets and percentages of labeled objects), the results are rather comparable and no particular approach can be deemed superior based on this collection of data sets.

6 Conclusions and perspectives

We have introduced a framework for the semi-supervised or unsupervised extraction of flat clusterings from optimal local cuts through cluster hierarchies. The extraction of a flat clustering from a cluster tree has been formulated as an optimization problem and a linear complexity algorithm (w.r.t. time and memory) has been presented that provides the globally optimal solution to this problem. Unlike most non-hierarchical (partitioning-like) algorithms for clustering, our method provides not only an optimal solution w.r.t. the different criteria it optimizes, but also the number of clusters as a by-product, rather than an explicit or implicit input parameter.

The proposed framework can, in principle, be applied to hierarchies of varied natures, so it is not hooked on a particular clustering inductive bias. In our experiments, we applied it to density-based cluster trees and average-linkage dendrograms (built in unsupervised and semi-supervised ways). We observed very promising results for real data sets in both scenarios, especially when contrasting our general framework with algorithms specialized in the respective scenarios.

In the semi-supervised extraction case, the proposed framework needs only a cluster tree and the degree to which the clusters in the tree satisfy instance-level constraints provided by the user. We have not considered weighted constraints, but all the developments in the paper can be straightforwardly generalized to do so. When the user provides no constraints at all (unsupervised case) or part of the clusters in the hierarchy cannot be distinguished in terms of constraint satisfactions/violations, then unsupervised measures of cluster quality can be applied. Such measures can also be used if one wants to balance the relative importance of constraint satisfaction and unsupervised cluster quality when solving the problem. The study of quality measures suitable for clustering hierarchies of varied natures, eventually constructed in a semi-supervised

way by using constraints of different types (possibly other than should- and should-not-link—e.g., see Davidson and Ravi 2009; Gilpin and Davidson 2011; Zheng and Li 2011), is an important topic for future research.

Acknowledgments This study was supported by Fapesp / CNPq (Brazil) and NSERC (Canada).

References

Ahmed EB, Nabli A, Gargouri F (2012) Shacun: semi-supervised hierarchical active clustering based on ranking constraints. In: Proceedings of the 12th industrial conference on data mining. Springer, Berlin, pp 194–208

Ankerst M, Breunig MM, Kriegel HP, Sander J (1999) Optics: ordering points to identify the clustering structure. SIGMOD Rec 28:49–60

Bade K, Nürnberger A (2006) Personalized hierarchical clustering. In: IEEE/WIC/ACM international conference on web intelligence (WI)

Bade K, Nürnberger A (2008) Creating a cluster hierarchy under constraints of a partially known hierarchy. In: SIAM international conference on data mining (SDM), Atlanta

Bade K, Hermkes M, Nürnberger A (2007) User oriented hierarchical information organization and retrieval. In: European conference on machine Learning (ECML), Corvallis, pp 518–526

Basu S, Davidson I, Wagstaff K (eds) (2008) Constrained clustering: advances in algorithms applications and theory. CRC Press, Boca Raton

Benkhalifa M, Mouradi A, Bouyakhf H (2001) Integrating wordnet knowledge to supplement training data in semi-supervised agglomerative hierarchical clustering for text categorization. Int J Intell Syst 16:929–947

Blockeel H, De Raedt L, Ramon J (1998) Top-down induction of clustering trees. In: International conference on machine learning (ICML), pp 55–63

Böhm C, Plant C (2008) Hissclu: a hierarchical density-based method for semi-supervised clustering. In: International conference on extending database technology (EDBT)

Boudaillier E, Hébrail G (1997) Interactive interpretation of hierarchical clustering. In: Principles of data mining and knowledge discovery, LNCS, vol 1263, Springer, Heidelberg, pp 288–298

Boudaillier E, Hébrail G (1998) Interactive interpretation of hierarchical clustering. Intell Data Anal 2:229–244

Brecheisen S, Kriegel HP, Kröger P, Pfeifle M (2004) Visually mining through cluster hierarchies. In: SIAM international conference on data mining (SDM)

Davidson I, Ravi S (2005) Agglomerative hierarchical clustering with constraints: theoretical and empirical results. In: European conference on principles and practice of knowledge discovery in databases (PKDD)

Davidson I, Ravi S (2009) Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. Data Min Knowl Disc 18:257–282

Davidson I, Wagstaff KL, Basu S (2006) Measuring constraint-set utility for partitional clustering algorithms. In: European conference on principles and practice of knowledge in databases (PKDD)

Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: International conference on knowledge discovery and data mining (KDD)

Everitt BS, Landau S, Leese M (2001) Cluster analysis, 4th edn. Arnold, London

Ferraretti D, Gamberoni G, Lamma E (2009) Automatic cluster selection using index driven search strategy. In: International conference of the Italian association for artificial intelligence (AI*IA)

Frank A, Asuncion A (2010) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 1 Dec 2011

Geusebroek JM, Burghouts G, Smeulders A (2005) The Amsterdam library of object images. Int J Comput Vis 61:103–112

Gilpin S, Davidson I (2011) Incorporating SAT solvers into hierarchical clustering algorithms: an efficient and flexible approach. In: ACM SIGKDD international conference on knowledge discovery and data mining (KDD)

Gupta G, Liu A, Ghosh J (2006) Hierarchical density shaving: a clustering and visualization framework for large biological datasets. In: IEEE ICDM workshop on data mining in bioinformatics (DMB)

Gupta G, Liu A, Ghosh J (2010) Automated hierarchical density shaving: a robust automated clustering and visualization framework for large biological data sets. IEEE/ACM Trans Comput Biol Bioinformatics 7(2):223–237

Hamasuna Y, Endo Y, Miyamoto S (2012) On agglomerative hierarchical clustering using clusterwise tolerance based pairwise constraints. J Adv Comput Intell Intell Inform 16(1):174–179

Hartigan JA (1975) Clustering algorithms. Wiley, New York

Herbin M, Bonnet N, Vautrot P (2001) Estimation of the number of clusters and influence zones. Pattern Recognit Lett 22(14):1557–1568

Hinneburg A, Keim DA (1998) An efficient approach to clustering in large multimedia databases with noise. In: International conference on knowledge discovery and data mining (KDD)

Horta D, Campello RJGB (2012) Automatic aspect discrimination in data clustering. Pattern Recognit 45:4370–4388

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218

Jain AK, Dubes RC (1988) Algorithms for clustering data. Englewood Cliffs, Prentice Hall

Kestler H, Kraus J, Palm G, Schwenker F (2006) On the effects of constraints in semi-supervised hierarchical clustering. In: IAPR workshop on artificial neural networks in pattern recognition (ANNPR)

Kettenring JR (2006) The practice of cluster analysis. J Classif 23:3–30

Kim HJ, Lee SG (2002) An effective document clustering method using user-adaptable distance metrics. In: ACM symposium on applied computing (SAC)

Klein D, Kamvar SD, Manning CD (2002) From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In: International conference on machine learning (ICML)

Kraus JM, Palm G, Kestler HA (2007) On the robustness of semi-supervised hierarchical graph clustering in functional genomics. In: 5th International workshop on mining and learning with graphs (MLG), pp 1–4

Kriegel HP, Kröger P, Sander J, Zimek A (2011) Density-based clustering. Wiley Interdiscip Rev Data Min Knowl Discov 1(3):231–240

Larsen B, Aone C (1999) Fast and effective text mining using linear-time document clustering. In: International conference on knowledge discovery and data mining (KDD)

Lelis L, Sander J (2009) Semi-supervised density-based clustering. In: International conference on data mining (ICDM)

Milligan GW, Cooper MC (1985) An examination of procedures for determining the number of clusters in a data set. Psychometrika 50(2):159–179

Miyamoto S, Terami A (2010) Semi-supervised agglomerative hierarchical clustering algorithms with pairwise constraints. In: IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1–6

Naldi M, Campello R, Hruschka E, Carvalho A (2011) Efficiency issues of evolutionary k-means. Appl Soft Comput 11(2):1938–1952

Paulovich F, Nonato L, Minghim R, Levkowitz H (2008) Least square projection: a fast high-precision multidimensional projection technique and its application to document mapping. IEEE Trans Vis Comput Graph 14(3):564–575

Sander J, Qin X, Lu Z, Niu N, Kovarsky A (2003) Automatic extraction of clusters from hierarchical clustering representations. In: Pacific-Asia conference on knowledge discovery and data mining (PAKDD)

Skarmeta AG, Bensaid A, Tazi N (2000) Data mining for text categorization with semi-supervised agglomerative hierarchical clustering. Int J Intell Syst 15(7):633–646

Struyf J, Džeroski S (2007) Clustering trees with instance level constraints. In: European conference on machine learning (ECML), pp 359–370

Stuetzle W (2003) Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. J Classif 20:25–47

Stuetzle W, Nugent R (2010) A generalized single linkage method for estimating the cluster tree of a density. J Comput Graph Stat 19(2):397–418

Sun H, Huang J, Han J, Deng H, Zhao P, Feng B (2010) gSkeletonClu: density-based network clustering via structure-connected tree division or agglomeration. In: IEEE international conference on data mining (ICDM)

Tan PN, Steinbach M, Kumar V (2006) Introduction to data mining. Addison-Wesley, Boston

Wagstaff KL (2002) Intelligent clustering with instance-level constraints. Ph.D. thesis, Department of Computer Science, Cornell University

Xiong T, Wang S, Mayers A, Monga E (2011) Semi-supervised parameter-free divisive hierarchical clustering of categorical data. In: Pacific-Asia conference on knowledge discovery and data mining (PAKDD), pp 265–276

Yeung KY, Fraley C, Murua A, Raftery AE, Ruzzo WL (2001) Model-based clustering and data transformations for gene expression data. Bioinformatics 17(10):977–987

Yeung KY, Medvedovic M, Bumgarner R (2003) Clustering gene-expression data with repeated measurements. Genome Biol 4(5):R34

Zhao H, Qi Z (2010) Hierarchical agglomerative clustering with ordering constraints. In: International conference on knowledge discovery and data mining (WKDD)

Zhao Y, Karypis G (2005) Hierarchical clustering algorithms for document datasets. Data Min Knowl Discov 10:141–168

Zheng L, Li T (2011) Semi-supervised hierarchical clustering. In: IEEE international conference on data mining (ICDM)