Knowl Inf Syst (2013) 35:1–32
DOI 10.1007/s10115-012-0522-9

REGULAR PAPER

Overlapping correlation clustering

Francesco Bonchi · Aristides Gionis · Antti Ukkonen

Received: 22 December 2011 / Revised: 23 April 2012 / Accepted: 16 June 2012 /
Published online: 22 July 2012

Abstract We introduce a new approach for finding overlapping clusters given pairwise similarities of objects. In particular, we relax the problem of *correlation clustering* by allowing an object to be assigned to more than one cluster. At the core of our approach is an optimization problem in which each data point is mapped to a small set of labels, representing membership in different clusters. The objective is to find a mapping so that the given similarities between objects agree as much as possible with similarities taken over their label sets. The number of labels can vary across objects. To define a similarity between label sets, we consider two measures: (i) a 0–1 function indicating whether the two label sets have non-zero intersection and (ii) the Jaccard coefficient between the two label sets. The algorithm we propose is an iterative local-search method. The definitions of label set similarity give rise to two non-trivial optimization problems, which, for the measures of set-intersection and Jaccard, we solve using a greedy strategy and non-negative least squares, respectively. We also develop a distributed version of our algorithm based on the BSP model and implement it using a Pregel framework. Our algorithm uses as input pairwise similarities of objects and can thus be applied when clustering structured objects for which feature vectors are not available. As a proof of concept, we apply our algorithms on three different and complex application domains: trajectories, amino-acid sequences, and textual documents.

Keywords Algorithms · Clustering · Overlapping clustering · Correlation clustering · Pregel

F. Bonchi · A. Gionis · A. Ukkonen (✉)
Yahoo! Research, Avinguda Diagonal 177, 08018 Barcelona, Spain
e-mail: aukkonen@yahoo-inc.com

F. Bonchi
e-mail: bonchi@yahoo-inc.com

A. Gionis
e-mail: gionis@yahoo-inc.com

1 Introduction

In many real-world applications, it is desirable to allow overlapping clusters as data points may intrinsically belong to more than one cluster. For example, in social networks, users belong to numerous communities. In biology, a large fraction of proteins belong to several protein complexes simultaneously, and genes have multiple coding functions and participate in different metabolic pathways. In information retrieval and text mining, documents, news articles, and web pages can belong to different categories.

In this paper, we formulate overlapping clustering as the problem of mapping each data point to a small set of labels that represent cluster membership. The number of labels does not have to be the same for all data points. The objective is to find a mapping so that the similarity between any pair of points in the dataset agrees as much as possible with the similarity of their corresponding sets of labels.

While this idea is general and could be instantiated in different clustering frameworks, in this paper, we apply it to the setting of *correlation clustering* [6], a clustering paradigm defined as follows: given a complete graph with positive and negative edges, the objective is to partition the graph so as to minimize the number of positive edges cut by the partition plus the number of negative edges not cut.

In our formulation, we still require a complete graph as input, but every edge is associated with a weight, which is a number in [0, 1]. Weights represent similarity between data points and the extent to which data points should be assigned to the same cluster. For defining distances between sets of labels, we consider two measures: a set-intersection indicator function and the Jaccard coefficient. We also constrain the maximum number of cluster labels allowed, either globally or per data point. These alternatives, together with the possibility of having fractional or binary edge weights, produce a whole family of problems.

In more detail, we make the following contributions:

- We define OVERLAPPING- CORRELATION- CLUSTERING, an optimization problem that extends the framework of correlation clustering to allow overlaps (Sect. 2). We show that the problem we define is **NP**-hard. We also discuss interesting connections of our problem with graph coloring and dimensionality reduction (Sect. 3).
- We propose to solve the OVERLAPPING- CORRELATION- CLUSTERING problem using a simple local-search algorithm. The iterative step of the local-search algorithm optimizes the labels of one object, given the labels of all other objects. Applying this local optimization, we iteratively improve the cost of the solution, until no further improvement can be made. We apply this general framework to both variants of the problem, Jaccard coefficient and set intersection (Sect. 4).
- In the case of the Jaccard coefficient, the iterative step of the local-search algorithm corresponds to a new problem, which we call JACCARD- TRIANGULATION. We prove that JACCARD- TRIANGULATION is **NP**-hard, and we devise a method based on non-negative least squares, followed by post-processing of the fractional solution (Sect. 4.2). In the case of set-intersection, the sub-problem is named HIT- N- MISS. This is a set-cover type of problem, which we solve using a greedy algorithm (Sect. 4.3).
- We present a distributed version of our algorithms, implemented on a map-reduce architecture, which can be used to find overlapping clustering in extremely large datasets (Sect. 5).
- We evaluate our algorithms on a collection of real-world datasets, showing that our algorithms produce overlapping clusters that resemble the ground truth. We also experiment with the idea of speeding up the algorithms by randomly eliminating pairs of data points

from consideration. Our results show that a significant amount of pruning can be achieved without degrading the quality of the solution. Finally, we test on a very large social-media dataset the scalability of the distributed implementation of our methods (Sect. 6).
– We illustrate the generality of our approach and the good quality of the results we obtain, by applying our methods on three different application domains: spatio-temporal trajectory analysis, bioinformatics, and document analysis. Our experiments provide ample evidence that our algorithms find meaningful overlapping clusters (Sect. 7).

The presentation of the paper is completed by surveying the related literature in Sect. 8 and discussing future work in Sect. 9.

2 Problem definition

We are given a set of n objects $V = \{v_1, \ldots, v_n\}$. We are also given a *similarity* value $s(u, v)$ for each pair of objects $(u, v) \in V \times V$. For instance, if the objects V represent documents, then $s(u, v)$ may be defined as the cosine between the vectors that represent the documents u and v; if the objects V represent records of a database table, then $s(u, v)$ may be defined as the fraction of attributes that records u and v agree on, etc. Hereinafter, we simply consider the values $s(u, v)$ as input to our problem, and we do not make any assumption on how to obtain those values. In this paper, we consider that the similarity function s takes values in the interval [0, 1]. We also study the special case in which the similarity function takes only values in the set {0, 1}.

2.1 Non-overlapping clustering

In non-overlapping clustering, we have at our disposal k cluster labels, denoted by $L = \{1, \ldots, k\}$, and the task is to assign cluster labels for each object in V. In other words, the clustering is defined by a labeling function $\ell : V \to L$. The objective is to assign labels to objects so that, to the highest possible degree, similar objects get assigned the same label. To make the previous statement precise, many formulations have been proposed in the literature, among which a very general and principled one is *correlation clustering*.

Problem 1 **(CORRELATION-CLUSTERING)** Given n objects $V = \{v_1, \ldots, v_n\}$, a pair-wise similarity function s over $V \times V$ finds a labeling function $\ell : V \to L$ that minimizes the cost

$$C_{\text{cc}}(V, l) = \sum_{\substack{(u,v)\in V\times V \\ \ell(u)=\ell(v)}} (1 - s(u, v)) + \sum_{\substack{(u,v)\in V\times V \\ \ell(u)\neq\ell(v)}} s(u, v). \tag{1}$$

The intuition underlying the above problem definition is that if two objects u and v are assigned to the same cluster, we should pay the amount of their dissimilarity $1-s(u, v)$, while if they are assigned to different clusters, we should pay the amount of their similarity $s(u, v)$. In the binary case, which is the most widely studied setting for CORRELATION- CLUSTERING, Eq. (1) says that we should pay for the number of pairs of objects that have similarity 0 and are clustered together and the number of pairs of objects that have similarity 1 and are clustered in different clusters.

In the traditional setting, no constraint on the maximum number of clusters is given, that is, $|L| = \Theta(n)$. This is indeed one of the advantages of CORRELATION- CLUSTERING: the number of clusters is not an input parameter, but "discovered" by the method.

2.2 Allowing overlaps

We now discuss how we extend the definition of CORRELATION- CLUSTERING in order to take into account overlapping of clusters. The main idea is to redefine the mapping function ℓ. Instead of mapping each object to a single cluster label $c \in L$, we relax the function ℓ so that it can map objects to any subset of cluster labels. Let $\mathcal{L}_+$ be the collection of all subsets of L except the empty set, that is, $\mathcal{L}_+ = 2^L \setminus \{\emptyset\}$, we now define the multi-labeling function ℓ as $\ell : V \to \mathcal{L}_+$. If an object v is mapped under ℓ to a set of cluster labels $\ell(v) = \{c_1, \ldots, c_s\} \in \mathcal{L}_+$, then we say that v participates in all the clusters $c_1, \ldots, c_s$.

For a good clustering, similar objects should be mapped to similar sets of cluster labels. Thus, to evaluate a solution to overlapping clustering, we define a similarity function H between sets of cluster labels, that is, $H : \mathcal{L}_+ \times \mathcal{L}_+ \to [0, 1]$. We now have the necessary ingredients to define the problem of overlapping correlation clustering.

Problem 2 (OVERLAPPING-CORRELATION-CLUSTERING) Given n objects $V = \{v_1, \ldots, v_n\}$, a pair-wise similarity function s over $V \times V$, and a similarity function H between sets, find a multi-labeling function $\ell : V \to \mathcal{L}_+$ that minimizes the cost

$$C_{\text{occ}}(V, \ell) = \sum_{(u,v) \in V \times V} |H(\ell(u), \ell(v)) - s(u, v)|. \tag{2}$$

Our definition of clustering aims at finding a multi-labeling that, to the highest possible degree, maintains the similarities between objects. Note that considering the error term $|H - s|$ is meaningful since both H and s are similarity functions that take values in the range $[0, 1]$.

To make our problem concrete, we need to define the similarity function H between two sets E, F of cluster labels. In this paper, we consider two such functions: the Jaccard coefficient $J(E, F) = \frac{|E \cap F|}{|E \cup F|}$, and the set-intersection indicator function I:

$$I(E, F) = \begin{cases} 1 & \text{if } E \cap F \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

The Jaccard coefficient is a very natural function to measure similarity between sets, and it has been used in a wide range of applications. On the other hand, in certain applications, we might be interested in whether the cluster-label sets intersect or not: for these, we use the set-intersection indicator.

2.3 Constraints

So far we have assumed a finite alphabet of labels and hence a maximum number of clusters $|L| = k$. This can be seen as the typical constraint in which one needs to specify an upper bound in the total number of clusters. However, for many applications, while we may have in our disposal a large number of clusters, we may not want to assign an object to all those clusters. For example, when clustering the users of a social network, we may want to use hundreds or even thousands of clusters; however, we may want to assign each user only in a handful of clusters.

Thus, we consider a second type of constraint in which we require that each object v should be mapped to at most p clusters, that is, $|\ell(v)| \leq p$ for all $v \in V$.

3 Problem characterization

In this section, we discuss the different variants of our problem, we establish its computational complexity, and we investigate connections with other problems.

First note that, based on our discussion in the previous section, when defining an instance of Problem 2, we have the following options:

- the similarity function s between objects may take values in the range $[0, 1]$, or it may take binary values in $\{0, 1\}$;
- the similarity function H between sets of cluster labels may be the Jaccard coefficient J or the intersection indicator I; and
- we may impose the local constraint of having at most p cluster labels for each object or we may not.

Any combination of the above options gives rise to a valid problem instance. We systematically refer to any of these problems using the notation (r, H, p), where: $r \in \{\mathtt{f}, \mathtt{b}\}$ refers to the range of the function s: $\mathtt{f}$ for fractional values and $\mathtt{b}$ binary; $H \in \{J, I\}$ refers to the similarity function H: J for Jaccard coefficient and I for set-intersection; and p refers to the value of p of the local constraint, so $p = k$ means that there is no local constraint.[1]

As an example of our notation, by $(\mathtt{b}, H, k)$, we refer to two different problem instances, where s takes binary values, H can be either J or I, and there is no local constraint.

3.1 Hardness results

Our first observation is that all instances specified by $(r, H, 1)$ correspond to the CORRELATION- CLUSTERING problem defined in Problem 1. The reason is that when $|\ell(v)| = 1$ for all v in V, then both the Jaccard coefficient and the intersection indicator take just 0 or 1 values. In particular, $|H(\ell(u), \ell(v)) - s(u, v)|$ becomes $1 - s(u, v)$ when $\ell(u) = \ell(v)$ and $s(u, v)$ when $\ell(u) \neq \ell(v)$. Thus, we easily establish that our problem is a generalization of the standard CORRELATION- CLUSTERING problem. Since CORRELATION- CLUSTERING is **NP**-hard, and since $p = 1$ is a special case of any (r, H, p) problem, the previous observation implies that the general OVERLAPPING- CORRELATION- CLUSTERING problem is also **NP**-hard. However, in order to show that the complexity of our problems does not derive exclusively from the hardness of the special case $p = 1$, we provide **NP**-hardness results that do not rely on such special case.

Theorem 1 *The problem instances* (r, I, p)*, with* $p > 1$*, are* **NP***-hard.*

Proof We show that the $(\mathtt{b}, I, p)$ problem is **NP**-hard, which also gives the **NP**-hardness for the $(\mathtt{f}, I, p)$ problem. We obtain the reduction from the problem of COVERING- BY-CLIQUES [27, GT17], which is the following: given an undirected graph $G = (V, E)$ and an integer $C \leq |E|$, decide whether G can be represented as the union of $c \leq C$ cliques. We can show that a zero-cost solution to the $(\mathtt{b}, I, C)$ problem identifies graphs having a covering by at most C cliques, and solutions with a cost larger than zero identify graphs that do not admit a covering by at most C cliques. Given an undirected graph $G = (V, E)$, we construct an instance of the $(\mathtt{b}, I, C)$ problem by simply setting the set of objects to be the set of nodes V. For each edge $(u, v) \in E$, we set $s(u, v) = 1$, while if $(u, v) \notin E$, we set $s(u, v) = 0$. We also set the total number of cluster labels $k = C$. It is easy to verify our claim: a zero-cost solution of $(\mathtt{b}, I, C)$ on input (V, s) corresponds to a covering of G by at most C cliques. □

[1] Note that we always assume that the value of k is given as input, although in our implementations this is interpreted as an upper bound as well; the resulting clustering may have less clusters.

In addition, due to the inapproximability of the COVERING- BY- CLIQUES problem, we can deduce that the problem instances (r, I, p) do not admit polynomial-time constant factor approximation algorithms unless $\mathbf{P} = \mathbf{NP}$.

We now turn our attention in the problems (r, I, k), that is, using set-intersection and no local constraint. Moreover, we consider the case in which we are allowed to use a very large number of cluster labels, in particular $k = \Theta(n^2)$.

Proposition 1 *For the problem instances* $(r, I, \Theta(n^2))$*, the optimal solution can be found in polynomial time.*

Proof We start by giving each object a unique cluster label. Then, we process each pair of objects for which $s(u, v) \geq \frac{1}{2}$. For any such pair of objects, we make a new label, which we assign to both objects, and never use again. Thus, for pairs with $s(u, v) \geq \frac{1}{2}$, the intersection of $\ell(u)$ and $\ell(v)$ is not empty, and thus, we pay $|1 - s(u, v)| \leq \frac{1}{2}$. On the other hand, for the pairs with $s(u, v) \leq \frac{1}{2}$, the intersection of $\ell(u)$ and $\ell(v)$ is empty, and thus, we pay $|s(u, v)| \leq \frac{1}{2}$. Since I takes only 0/1 values, no other solution can cost less, and thus, the previous process gives an optimal solution. □

When we have binary similarities, the above process straightforwardly provides a 0 cost solution.

Corollary 1 *The problem instance* $(\mathfrak{b}, I, \Theta(n^2))$ *always admits a 0 cost solution that can be found in polynomial time.*

3.2 Connection with graph coloring

Given that the problem instance $(\mathfrak{b}, I, k)$ always admits a solution of 0 cost if we allow enough cluster labels, we next ask which is the minimum number of cluster labels k needed for a 0 cost solution. We characterize this number by pointing out a connection with the GRAPH- COLORING problem, whose formulation we recall next. A proper coloring of a graph $G = (V, E)$ is a function $c : V \to \{1, \ldots, k\}$ so that for all $(u, v) \in E$, we have $c(u) \neq c(v)$. The GRAPH- COLORING problem asks to find the smallest number k, known as the chromatic number $\chi(G)$ of G, for which a proper coloring of G exists.

Going back to the binary $(\mathfrak{b}, I, k)$ instance of the OVERLAPPING- CORRELATION- CLUSTERING problem, given the set of objects V and similarity function s, we consider *similar pairs* $P^+ = \{(u, v) \in V \times V \mid s(u, v) = 1\}$ and *dissimilar pairs* $P^- = \{(u, v) \in V \times V \mid s(u, v) = 0\}$. Using these, we define the graph $\widehat{G} = (P^+, \widehat{E})$, with similar pairs as nodes, and the set of edges $\widehat{E}$ given by the dissimilar pairs as follows:

$$\widehat{E} = \{((u, v), (x, y)) \in P^+ \times P^+ \mid \{(u, x), (u, y), (v, x), (v, y)\} \cap P^- \neq \emptyset\}.$$

We have:

Proposition 2 *The chromatic number* $\chi(\widehat{G})$ *of* $\widehat{G}$ *is equal to the minimum number of cluster labels required by a zero-cost solution to the* $(\mathfrak{b}, I, k)$ *problem with input* (V, s).

Proof We observe that a color in $\widehat{G}$ corresponds to a cluster in our problem. The colors are assigned to pairs of objects in V, which ensures that the positive pairs P^+ are satisfied.

On the other hand, the constraint of having a proper coloring ensures that the negative pairs P^- are also satisfied. Thus, a proper coloring on $\widehat{G}$ corresponds to a zero-cost solution on our problem. □

Although the previous result is theoretically interesting, it has limited practical relevance, as we are interested in minimizing the error given a specific number of clusters.

To make the connection practically useful, we would need to relax the GRAPH- COLORING problem, so that it allows for a less strict definition of coloring. Namely, we would like to allow for colorings that for certain cost may allow the following relaxations: (i) $(u, v) \in E$ not necessarily implies $c(u) \neq c(v)$—corresponding to violations on P^-; and (ii) nodes may be left uncolored—corresponding to violations on P^+. We believe that this is an interesting path that may lead to novel algorithms for OVERLAPPING- CORRELATION- CLUSTERING. We plan to investigate this research direction in our future work.

3.3 Connection with dimensionality reduction

We finally remark that our problem can be seen as an instance of *dimensionality-reduction*. We briefly remind the problem of dimensionality reduction here: Given a set of points in a high-dimensional space, the goal is to map each point x in the original space to a point $h(x)$ in space of lower dimensionality, so that for any pair of points x and y, their distance $d(x, y)$ in the high-dimensional space is preserved as well as possible by the distance $d(h(x), h(y))$ in the lower-dimensional space. Dimensionality reduction is a problem that has been studied, among other areas, in theory, data mining, and machine learning and has many applications, for example, in proximity search, feature selection, component analysis, visualization, and more. The connection between dimensionality reduction and the OVERLAPPING- CORRELATION- CLUSTERING problem, defined by Eq. (2), is apparent. However, the difference is that dimensionality reduction methods are typically defined for geometric spaces. Alternatively, they operate by hashing high-dimensional or complex objects in a way that similar objects have high collision probability [12,34]. To the best of our knowledge, the case that the projected space is a set-system and similarities are measured by a set distance function has not been considered before.

4 Algorithms

We propose a local-search algorithm that optimizes the labels of one object in the dataset, when the labels of all other objects are fixed. We apply this framework both for the Jaccard coefficient and the intersection-function variants of the problem, proposing novel algorithms for these two local optimization problems in Sects. 4.2 and 4.3, respectively.

4.1 The local-search framework

A typical approach for multivariate optimization problems is to iteratively find the optimal value for one variable *given* values for the remaining variables. The global solution is found by repeatedly optimizing each of the variables in turn until the objective function value no longer improves. In most cases, such a method will converge to a local optimum. The algorithm we propose falls into this framework. At the core of our algorithm is an efficient method for finding a good labeling of a single object given a fixed labeling of the other objects. We can guarantee that the value of Eq. 2 is non-increasing with respect to such optimization steps. First, we observe that the cost of Eq. (2) can be rewritten as

Algorithm 1 `LocalSearch`

1: initialize ℓ to a valid labeling;
2: **while** $C_{\text{occ}}(V, \ell)$ decreases **do**
3: **for** each $v \in V$ **do**
4: find the label set L that minimizes $C_{v,p}(L \mid \ell)$;
5: update ℓ so that $\ell(v) = L$;
6: **return** ℓ

$$C_{\text{occ}}(V, \ell) = \frac{1}{2} \sum_{v \in V} \sum_{u \in V \setminus \{v\}} |H(\ell(v), \ell(u)) - s(v, u)|$$
$$= \frac{1}{2} \sum_{v \in V} C_{v,p}(\ell(v) \mid \ell),$$

where

$$C_{v,p}(\ell(v) \mid \ell) = \sum_{u \in V \setminus \{v\}} |H(\ell(v), \ell(u)) - s(v, u)| \tag{3}$$

expresses the error incurred by node v when it has the labels $\ell(v)$, and the remaining nodes are labeled according to ℓ. The subscript p in $C_{v,p}$ serves to remind us that the set $\ell(v)$ should have at most p labels. Our general local-search strategy is summarized in Algorithm 1.

Line 4 is the step in which `LocalSearch` seeks to find an optimal set of labels for an object v by solving Equation (3). This is also the place that our framework differentiates between the measures of Jaccard coefficient and set-intersection.

4.2 Local step for Jaccard coefficient

Problem 3 **(JACCARD-TRIANGULATION)** Consider the set $\{\langle S_j, z_j \rangle\}_{j=1\ldots n}$, where S_j are subsets of a ground set $U = \{1, \ldots, k\}$, and z_j are fractional numbers in the interval $[0, 1]$. The task is to find a set $X \subseteq U$ that minimizes the distance

$$d(X, \{\langle S_j, z_j \rangle\}_{j=1\ldots n}) = \sum_{j=1}^{n} |J(X, S_j) - z_j|. \tag{4}$$

The intuition behind Equation (4) is that we are given sets S_j and "target similarities" z_j and we want to find a set whose Jaccard coefficient with each set S_j is as close as possible to the target similarity z_j. A moment's thought can convince us that Eq. (4) corresponds exactly to the error term $C_{v,p}(\ell(v) \mid \ell)$ defined by Eq. (3), and thus, in the local-improvement step of the `LocalSearch` algorithm.

To our knowledge, JACCARD- TRIANGULATION is a new and interesting problem, which has not been studied before, in particular in the context of overlapping clustering. The most-related problem that we are aware of is the problem of finding the Jaccard median, which was recently studied by Chierichetti et al. [16]. The Jaccard-median problem is a special case of the JACCARD- TRIANGULATION problem, where all similarities z_j are equal to 1. Chierichetti et al. provide a PTAS for the Jaccard-median problem. However, their techniques seem mostly of theoretical interest and do not extend beyond the special case where all $z_j = 1$.

However, since JACCARD- TRIANGULATION is a generalization of the Jaccard-median problem that has been proven **NP**-hard [16], the following is immediate.

Theorem 2 JACCARD- TRIANGULATION *is* **NP**-*hard.*

We next proceed to discuss our proposed algorithm for the JACCARD- TRIANGULATION problem. The idea is to introduce a variable x_i for every element $i \in U$. The variable x_i indicates if element i belongs in the solution set X. In particular, $x_i = 1$ if $i \in X$ and $x_i = 0$ otherwise. We then assume that the size of set X is t, that is,

$$\sum_{i \in U} x_i - t = 0. \tag{5}$$

Now, given a set S_j with target similarity z_j, we want to obtain $J(X, S_j) = z_j$, for all $j = 1, \ldots n$, or

$$J(X, S_j) = \frac{\sum_{i \in S_j} x_i}{|S_j| + t - \sum_{i \in S_j} x_i} = z_j,$$

which is equivalent to

$$(1 + z_j) \sum_{i \in S_j} x_i - z_j t = z_j |S_j|, \tag{6}$$

and we have one Equation of type (6) for each pair $\langle S_j, z_j \rangle$. We observe that Eqs. (5) and (6) are linear with respect to the unknowns x_i and t. On the other hand, the variables x_i and t take integral values, which implies that the system of Eqs. (5) and (6) cannot be solved efficiently. Instead, we propose to relax the integrality constraints to non-negativity constraints $x_i, t \geq 0$ and solve the above system in the least-squares sense. Thus, we apply a non-negative least-squares optimization method (NNLS), and we obtain estimates for the variables x_i and t.

The solution we obtain from the NNLS solver has two drawbacks: (i) it does not incorporate the constraint of having at most p labels, and (ii) most importantly, it does not have a clear interpretation as a set X, since the variables x_i may take any non-negative value, not only 0–1. We address both of these problems with a greedy post-processing of the fractional solution: We first sort the variables x_i in decreasing order, breaking ties arbitrarily. We then obtain a set X_q by setting the first q variables x_i to 1 and the rest to 0. We vary q from 1 to p. Out of the p different sets X_q that we obtain this way, we select the one that minimizes the cost $d(X_q, \{\langle S_j, z_j \rangle\})$ and return this as the solution to the JACCARD- TRIANGULATION problem.

An alternative approach could be to optimize the sum of squares of differences of Equation (6), for all j, subject to the constraint of Equation (5), the constraint $\sum_{i \in U} x_i \leq p$, and the constraints $0 \leq x_i \leq 1$. This formulation leads to a quadratic program, which can be also solved by standard solvers, albeit in way less efficient than non-negative least squares. Since this computation is performed for each object in the inner loop of Algorithm 1, in this paper, for efficiency reason, we adopt the non-negative least-squares formulation.

4.3 Local step for set-intersection function

Following the approach of the previous section, we formulate the problem that we need to solve for the local-improvement step of the `LocalSearch` algorithm (line 4 of Algorithm 1) in the case of the set-intersection function I.

Problem 4 **(HIT-N-MISS)** Let $\mathcal{C}$ be a collection of n tuples of the from $\langle S_j, h_j, m_j \rangle$, with $j = 1 \ldots n$, where S_j are subsets of a ground set $U = \{1, \ldots, k\}$, while h_j and m_j are non-negative numbers. A set $X \subseteq U$ partitions $\mathcal{C}$ in $\mathcal{C}_X = \{S_j \mid I(X, S_j) = 1\}$ and $\mathcal{C}_{\bar{X}} = \{S_j \mid I(X, S_j) = 0\}$. The task is to find a set X in order to minimize the distance

$$d(X, \{\langle S_j, h_j, m_j\rangle\}) = \sum_{j|S_j \in \mathcal{C}_X} h_j + \sum_{j|S_j \in \mathcal{C}_{\bar{X}}} m_j. \quad (7)$$

Once again, we should be able to verify that Eq. (7) corresponds to the cost $C_{v,p}(\ell(v) \mid \ell)$ defined by Eq. (3) in the case that the cluster-label similarity function H is the set-intersection function I. In fact, for the problem instances defined by Eq. (3), we always have $h_j + m_j = 1$. However, since we do not know how to take advantage of the additional structure $h_j + m_j = 1$, we just formulate Problem 4 in its generality.

The HIT- N- MISS problem is related to set-cover type of problems. As in set-cover, we are given a collection $\mathcal{C}$ of sets S_j. Each set is accompanied with two penalty scores, a *hit* penalty p_j and a *miss* penalty n_j. Our task is to find a new set X in order to either hit or miss the sets S_j, as dictated by their penalty scores h_j and m_j. In particular, for each set S_j that X hits, we have to pay its hit penalty h_j, while for each set S_j that X misses, we have to pay its miss penalty m_j. The HIT- N- MISS problem is isomorphic to the *positive-negative partial set-cover* problem, studied by Miettinen [42], who showed that the problem is not approximable within a constant factor, but it admits an $O(\sqrt{n}\log n)$ approximation.

In our setting, we solve the HIT- N- MISS problem with a simple greedy strategy: Starting from $X_0 = \emptyset$, let X_t the current solution and let $A = U \setminus X_t$ be the set of currently available items (cluster labels). Then, for the next step of the greedy, we pick the item i from the set of available items A that yields the lowest distance cost, evaluated as $d(X_t \cup \{i\}, \{\langle S_j, h_j, m_j\rangle\})$. We terminate when there is no further decrease in the cost or when we reach the maximum number of cluster labels allowed, that is, $t = p$.

4.4 Initialization

The local-search algorithm described above requires an initial labeling of the nodes. This can be done in several ways. In the experiments, we always use a random initialization. However, in our future work, we plan to investigate an initialization based on solving a standard graph coloring problem on the graph $\widehat{G}$ of Proposition 2.

5 Overlapping clustering of big data

In this section, we present the distributed version of our algorithm, implemented on a *map-reduce* architecture [20], which can be used to cluster really large problem instances.

The main idea for a distributed implementation of Algorithm 1 is to note that in Step 4 of the algorithm, in each iteration, the labels of a nodes $v \in V$ can be computed with access to the input similarity matrix and the labels at the neighboring nodes at the current iteration. Thus, in each iteration of the algorithm, the new labels for each node $v \in V$ can be computed *independently*, provided that each node has *read access* to the input similarity matrix and the labels of its neighboring nodes. In order to organize the distributed computation for new per-node labels, and the read access to the input similarity matrix and the labels from other nodes, we use Pregel [40], a distributed-computing paradigm for graph algorithms.

5.1 Pregel basics

Pregel is a framework for distributed computing for graph algorithms, which is based on the Bulk Synchronous Parallel model of computation [55]. Like *map-reduce* programs [20], Pregel programs are designed to run on clusters with thousands of nodes but are nonetheless

very simple for a programmer to express. We make use of `Giraph`,[2] an implementation of the Pregel framework for `Hadoop`,[3] the well-known open-source implementation of the map-reduce model. `Giraph` programs run on `Hadoop` as regular map-reduce applications, which makes deploying them on an existing `Hadoop` installation rather easy.

Next, we give a brief overview of Pregel. For full details, the reader is referred to the original paper that introduced the framework [40]. Pregel is motivated by the family of graph algorithms in which computation can be expressed as a relatively simple operation that is executed on the nodes of the graph over a number of iterations. For example, in the standard iterative algorithm for PageRank [11], on each iteration, the score of every node is updated given the scores of its neighboring node. The crucial property is that the iterative update operation can be performed in parallel for all the nodes of the graph. We can thus assign all nodes of the graph to separate processing units, which communicate with each other the updated node scores between the iterations of the computation.

Consequently, the basic unit of computation in a Pregel program is the *node*. As stated above, all nodes can in theory execute their update operation simultaneously, but in practice, a Pregel program is run using a number of worker machines, each of which processes sequentially a subset of the nodes of the graph. Computation takes place in *supersteps*, which in our case correspond to iterations of the `for` loop of Algorithm 1. To write a Pregel application, the programmer only needs to specify the algorithm that a single node executes on every superstep. All remaining functionalities of the distributed application, such as message passing and checkpointing, are handled by the framework.

Communication between nodes in a Pregel program is straightforward. At the start of a superstep, every node may receive a number of *messages* from its neighbors. Likewise, at the end of a superstep, the nodes may send messages to their neighbors. The Pregel framework routes messages to their recipients once all nodes have completed their update operations in the current superstep. The messages are delivered at the start of the next superstep. A node can also *vote to halt* the computation if it considers itself finished. For example, in our case, a node will vote to halt if its set of labels did not change during the current superstep. A node that has voted to halt no longer executes in subsequent supersteps unless it receives a message from one of its neighbors. Receiving a message re-activates a halted node. The distributed computation finishes when all nodes have voted to halt.

5.2 Overlapping correlation clustering with Pregel

We now discuss in more detail how our general local-search algorithm (Algorithm 1) can be implemented as a Pregel program. We only need to specify the algorithm that each node executes during every superstep. The outline of this algorithm is given in Algorithm 2. In the first superstep (superstep 0), all nodes simply initialize themselves with a random set of labels and send these labels to their neighbors. In subsequent supersteps, every node first checks whether certain stopping conditions are satisfied and votes to halt if this is the case. We use two stopping conditions. The first one is a threshold on the cost. If the cost drops below this value, all nodes vote to halt. The second is a simple upper bound on the number of iterations. Otherwise, the node reads the messages it has received and uses the updated label sets of its neighbors to recompute its own label set. If the label set changes, the node sends this set to its neighbors. Otherwise, the node votes to halt.

[2] http://incubator.apache.org/giraph/.

[3] http://hadoop.apache.org/.

Algorithm 2 Computation at a Pregel node

```
1: if superstep = 0 then
2:   initialize labels at random
3:   send label set to neighbors
4: else
5:   if stopping conditions met then
6:     vote to halt
7:   else
8:     receive updated label sets from neighbors
9:     recompute new labels
10:    if labels did not change or previous solution had lower cost then
11:      vote to halt
12:    else
13:      assign new labels as our label set
14:      send label set to neighbors
```

An important difference between the sequential and the distributed implementations of Algorithm 1 is that the sequential variant updates the label sets of one object at a time, while the distributed algorithm updates all objects simultaneously. That is, when updating labels of object v in iteration i, the sequential algorithm uses the newest possible labels of the neighboring objects. In contrast, when the distributed implementation updates labels of v in iteration i, it can only use neighbor labels from iteration $i-1$. This has the drawback that the cost is not guaranteed to decrease at every step, and the distributed algorithm is more prone to get stuck at a non-optimal solution.

We tackle this problem with a simple approach based on randomization. On every iteration, the distributed algorithm updates only a *random subset of the objects*. Our randomized approach is motivated by the following consideration: Suppose that the distributed variant updates only *one* object in each iteration. Then, the distributed algorithm mimics the computation of the sequential algorithm albeit in a very inefficient way and loses all the benefits of parallelization. By updating a randomly chosen fraction of the objects, we explore variants that lie between the fully parallelized (update all) and the fully sequential algorithm (update only one). In Sect. 6.3, we conduct experiments to quantify the performance of the distributed algorithm as a function of the fraction of nodes that are updated.

6 Experimental evaluation

In this section, we present our experimental evaluation regarding performance and quality of the obtained clustering. In the next section, we show how to apply our framework and obtain meaningful overlapping clusterings in three different application domains.

6.1 Recovering ground-truth overlapping clusterings

Our first experiment is aimed at assessing under what conditions our algorithms, named OCC- JACC (Jaccard) and OCC- ISECT (set-intersection indicator), can reconstruct a ground-truth clustering.

We consider two publicly available datasets (EMOTION and YEAST), originally used in the context of multi-label classification. In these datasets, each example is associated with multiple labels, as opposed to the single-label classification paradigm in which is example is assigned to only one class. Such a labeling can be interpreted as a *ground-truth* overlapping

clustering g, where each label induces a cluster. In this experiment, the input to our algorithms is the Jaccard coefficients between the labels of every object pair. For OCC- ISECT, we convert the weights to positive and negative edges by labeling the edge as positive unless the Jaccard coefficient is equal to zero. The EMOTION dataset has 593 objects and 6 labels. The YEAST dataset has 2,417 objects and 14 labels. More information on the datasets can be found at http://mulan.sourceforge.net/datasets.html.

Performance is evaluated by comparing the labeling ℓ produced by the algorithms, with the ground-truth clustering g, using the measures of precision and recall, as adapted for the overlapping-clustering task by Banerjee et al. [5]:

$$\mathrm{prec}_g(\ell) = \frac{|P(\ell) \cap P(g)|}{|P(\ell)|} \quad \text{and} \quad \mathrm{rec}_g(\ell) = \frac{|P(\ell) \cap P(g)|}{|P(g)|},$$

where $P(x) = \{(u, v) : x(u) \cap x(v) \neq \emptyset\}$ is the set of pairs of objects with at least one common label in labeling x. We also report the average cost-per-edge, that is, the cost of the solution $C_{\mathrm{occ}}(V, \ell)$ as in Eq. (2), divided by the number of edges in the input.

We consider both the case where the algorithm is given only the total number of clusters k, as well as the variant where also the node-specific bound on the number of labels, denoted p, is given. In the former case, p was not specified, while in the latter case, k was fixed to the true number of distinct labels.

Medians of the metrics over 30 trials together with 90 % confidence intervals are shown in Figs. 1 and 2 for the OCC- JACC and OCC- ISECT algorithms, respectively. With the EMOTION dataset OCC- JACC performs better than OCC- ISECT, with YEAST the situation is reversed.

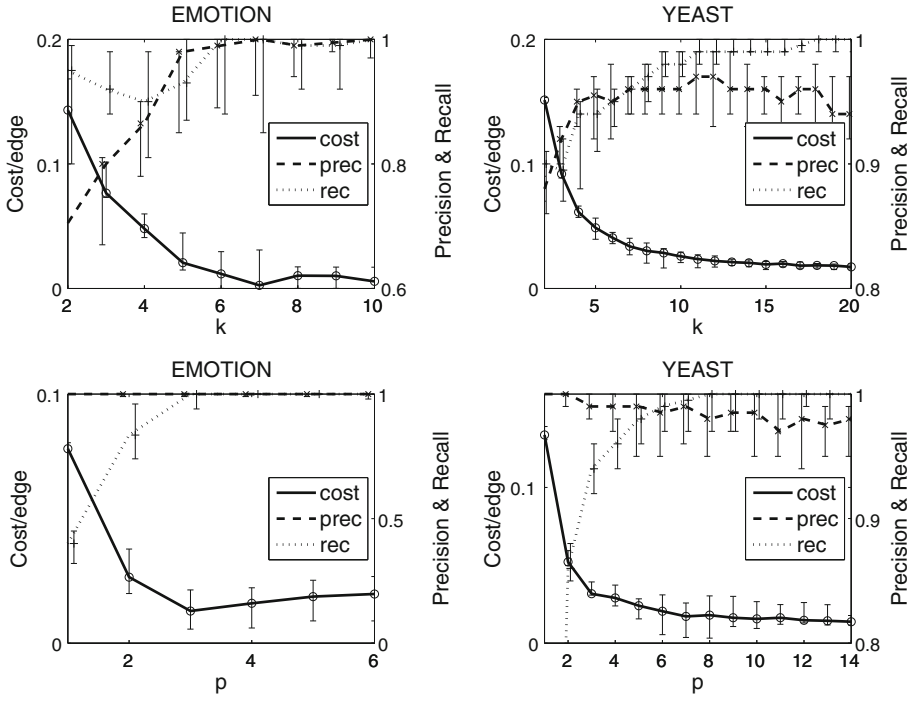

Fig. 1 Cost-per-edge, precision, and recall of OCC- JACC as a function of k, the total number of distinct labels (*top row*), and as a function of p, the maximum number of labels per node (*bottom row*)

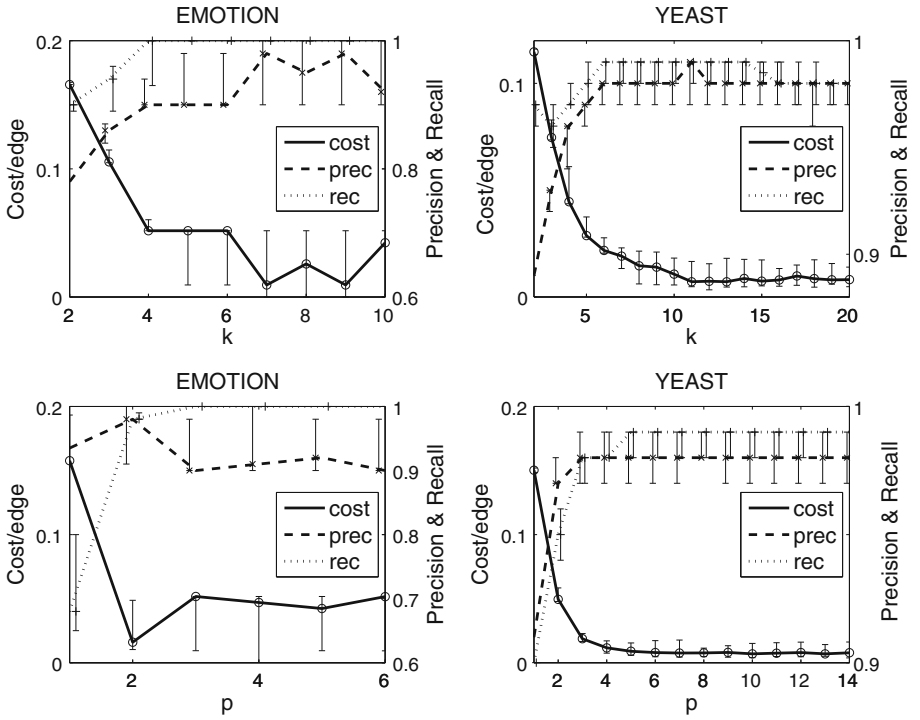

Fig. 2 Same as in Fig. 1, but for the OCC- ISECT algorithm

Observe that in the case where p is varied, precision is high already for low values. As p increases recall also increases, which is an expected behavior, as the number of pairs that belong to the same cluster tends to increase when overlaps are allowed.

6.1.1 Comparison with non-negative matrix factorization methods

We also compare the overlapping clusterings produced by our methods with a state-of-the-art method for symmetric non-negative matrix factorization (SNMF) [33]. The connection between our problem and SNMF is discussed in the related work, Sect. 8.4. The comparison is performed again on the basis of precision and recall in recovering ground-truth overlapping clusterings, in the EMOTION and YEAST datasets.

We implemented in Matlab the α-SNMF algorithm by He et al. [33] and initialized it with vectors containing a uniform random sample in the interval [0, 1]. In all experimental settings, the algorithm was run for 500 iterations; by checking the error plots, we verified that the method always converged in less than 100 iterations. The number of clusters k was set to the number of clusters for the ground truth, that is, $k = 6$ for EMOTION and $k = 14$ for YEAST. For the OCC- JACC and OCC- ISECT algorithms, the results shown are obtained by considering the median over 30 runs, as in the previous experiments.

As the α-SNMF algorithm produces fractional solutions, we have to use a rounding threshold for deciding cluster membership. In Fig. 3, we report the results in terms of precision and recall as a function of the rounding threshold. As OCC- JACC and OCC- ISECT algorithms do not need this threshold, their corresponding lines are flat.

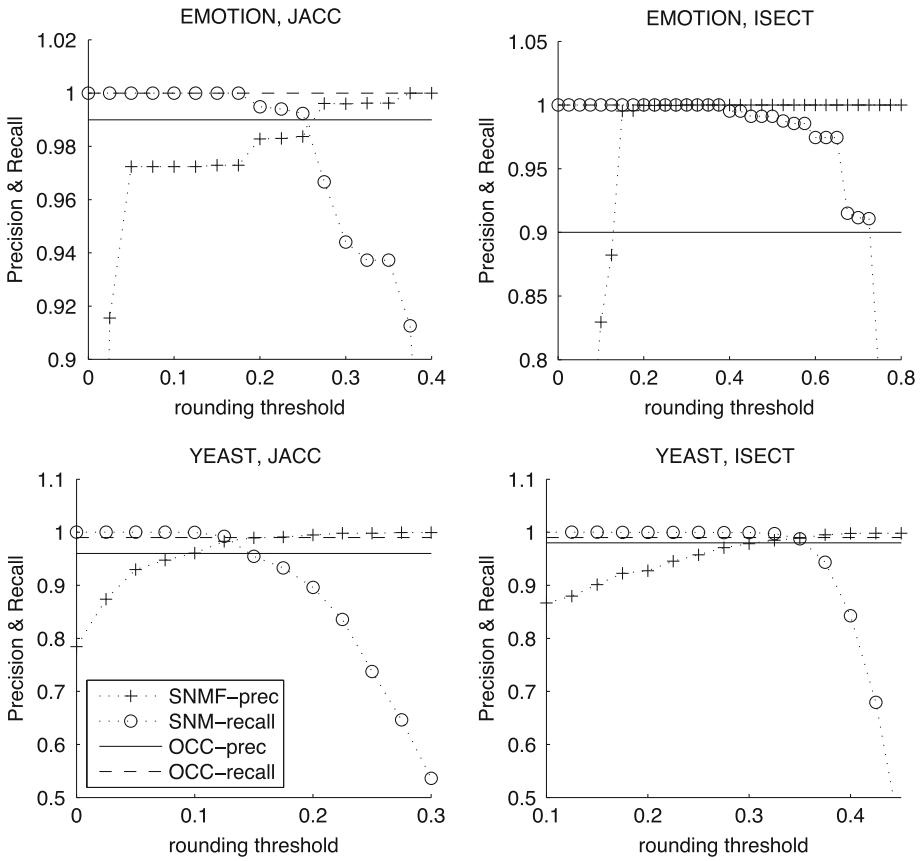

Fig. 3 Comparison with symmetric non-negative matrix factorization (SNMF) [33] in recovering ground-truth overlapping clusterings. For the SNMF algorithm, the precision and recall is shown as a function of the rounding threshold for deciding the cluster assignment

From the results shown in Fig. 3, we conclude that in most cases our overlapping-clustering methods outperform the SNMF method. Additionally, our methods perform robustly without having to set an additional parameter, such as the rounding threshold. In contrast, the SNMF algorithm is quite sensitive to the rounding threshold, especially for the YEAST dataset. The only case that the SNMF algorithm performs somewhat robustly and it is not so sensitive to the rounding threshold parameter is for the EMOTION dataset with the set-intersection measure—upper right plot in Fig. 3.

6.2 Pruning the input

In the basic form, the input to our algorithms contains all $|V| \times |V|$ pairwise similarities. However, it turns out that there can be a lot of redundancy in this input. Often we can prune most of the pairwise comparisons with negligible loss in quality. This is an important characteristic, as it allows us to apply the algorithms also for larger datasets. Selecting the best set of edges to prune is an interesting problem in its own right. In this experiment, we took the simple

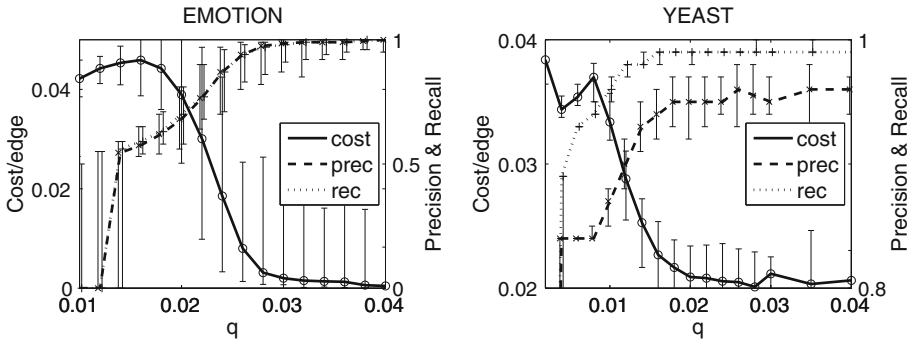

Fig. 4 Pruning experiment using OCC- JACC. Cost-per-edge and precision and recall as a function of the pruning threshold q

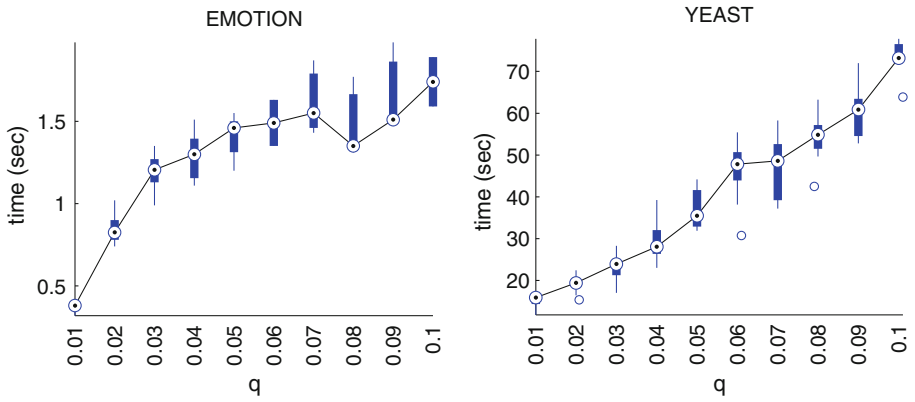

Fig. 5 Running time of OCC- JACC as a function of the pruning threshold q

approach and prune edges at random: an edge is taken in consideration with probability q (called the *pruning threshold*) independently of the other edges.

In Fig. 4, we show edge-specific cost as well as precision and recall as a function of q for the OCC- JACC algorithm (the curves are again medians over 30 trials). Clearly with these example datasets, the pruning threshold can be set very low. Also, there is a noticeable "threshold effect" in the cost-per-edge that may serve as an indicator to find the pruning threshold in a setting where a ground truth is not available. This suggests that in practice it is not necessary to use all pairwise comparisons, a sample of the graph may be enough. In fact, the results for YEAST shown in Figs. 1 and 2 were computed with $q = 0.05$.

In terms of computational speedup, pruning has a very positive effect. Figure 5 shows the running time of OCC- JACC (Python implementation on a 1.8 GHz CPU) as a function of q for both datasets. As expected, the dependence between running time and q is roughly linear. For EMOTION, we observe a weaker effect due to the small size of the data. With YEAST, the advantages are more evident. Without pruning ($q = 1.0$) OCC- JACC clusters YEAST in roughly 6 min. We can reduce this to about 35 s by setting $q = 0.05$ and still get very good results as was seen in Fig. 4.

It is also of interest to see how fast the algorithms converge, and how the convergence depends on the pruning threshold q. That is, how does the cost evolve over the execution of

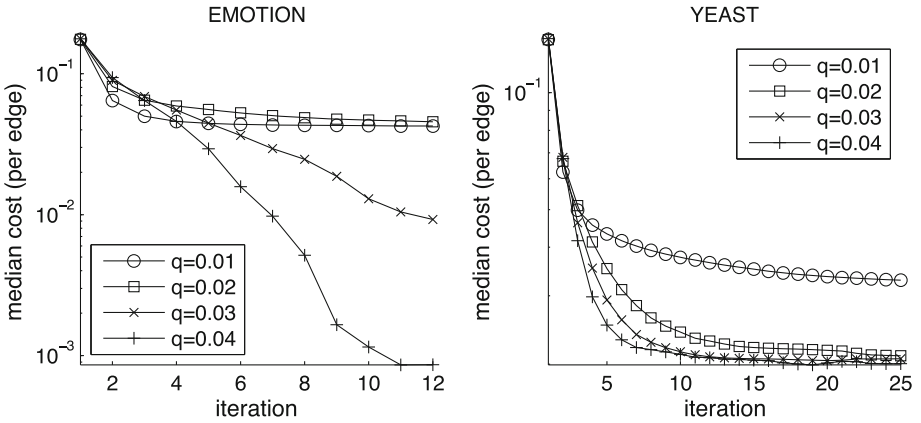

Fig. 6 Median (over 30 trials) cost-per-edge as a function of the iteration number of the OCC- JACC algorithm

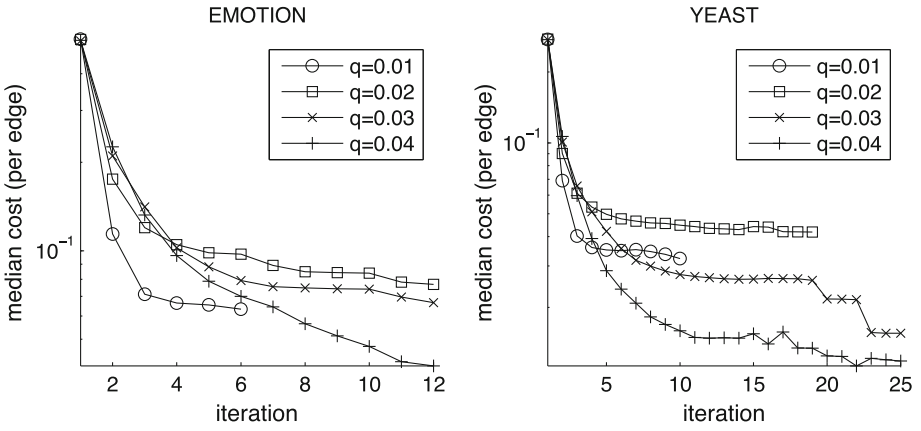

Fig. 7 Same as in Fig. 6, but for the OCC- ISECT algorithm

the algorithm? This is shown in Figs. 6 and 7 for OCC- JACC and OCC- ISECT, respectively. As expected the plots confirm that the stronger the pruning the sooner the methods converge.

6.3 Distributed overlapping correlation clustering

We implemented the OCC- ISECT algorithm as a Pregel program as described in Sect. 5. We refer to this distributed implementation of the OCC- ISECT algorithm as OCC- MR. The purpose of the experiments described in this section is twofold. First, we show how the our overlapping-clustering algorithms can be scaled up by using a distributed implementation. Second, we study special characteristics of the distributed implementation, which was discussed in Sect. 5.2. In particular, we show that selecting the fraction of vertices that are updated on each superstep can have a considerable effect on the performance of the algorithm.

For these purposes, we use a dataset obtained from Flickr,[4] a popular online community for image and video sharing. We consider a set of 78.5 million users who are affiliated to some

[4] http://www.flickr.com/.

interest groups, that is, group of users with a common interest in photos regarding a specific subject or technique (for example, "`Nikon Selfportrait`" or "`HDR Panoramas`"). For each pair of users, we compute the Jaccard coefficient of the interest groups they belong to. This process produces approximately 26.2 million non-zero similarities that are used as input for our clustering.

6.3.1 Degree of parallelism

As a first experiment, we studied how much parallelization actually speeds up computation. The OCC- MR algorithm is prone to communication overhead, as the worker machines must communicate updated label sets to each other after every superstep. For dense graphs, such communication overhead may result in a lot of messages being routed from node to node. Moreover, simply loading the input from the distributed file system may introduce a lot of overhead.

The running time of the algorithm is proportional to both the global number of labels (k), as well as the local bound (p). By increasing these two parameters, we can make the computation at a single vertex more costly. Thus, to test the performance of the algorithm with respect to these two parameters, we run the OCC- MR algorithm on the FLICKR dataset with $k = 50$ and $p = 5$ (labeled as "easy" instance), as well as $k = 200$ and $p = 20$ (labeled as "hard" instance). We expect the benefits of a distributed implementation to be stronger when the local optimization step takes a longer time to compute, as is the case with the "hard" instance.

Figure 8 shows the running time of the OCC- MR algorithm as a function of the number of worker nodes. The time required for actual computation (including message passing), and the time required for the data-loading phase is shown separately. For both "easy" and "hard" instances, we observe a clear decrease in the running time as the number of worker nodes increases from 25 to 200—a factor of 8. However, while in the "easy" instance, the running time decreases by a factor of 3.2, in the "hard" instance, it decreases by a factor of 5.5. This effect verifies our hypothesis that parallelization is more useful for certain parameter settings. For the "easy" instance, the running time starts to be dominated by communication overhead with 100 worker machines or more, as indicated by the negligible decrease in computation time from 100 to 200 worker nodes.

Moreover, as the number of workers increases, loading and distributing the input slows down. Since both "easy" and "hard" instances use the same input, there is obviously no difference between their loading steps. It is worth noting, however, that when running "easy" on 75 or more machines, merely loading and distributing the input takes longer than actual computation. With "hard" this effect does not happen until 200 or more workers.

Clearly, these results are to a certain degree specific to the Pregel framework that is being used, in our case `Giraph`. However, in practice, all Pregel programs are subject to similar issues.

6.3.2 Fraction of updated nodes

The distributed implementations of our overlapping-clustering algorithms are likely to perform better if only a subset of the objects is updated on each iteration. We study this empirically by running the OCC- MR algorithm on the FLICKR dataset with different update probabilities. Evidently, low update probabilities should lead to slow convergence, as only a few nodes are updated in each superstep. Even if the algorithm is likely to converge to a very good solution eventually, this might take too many iterations in practice. By increasing the

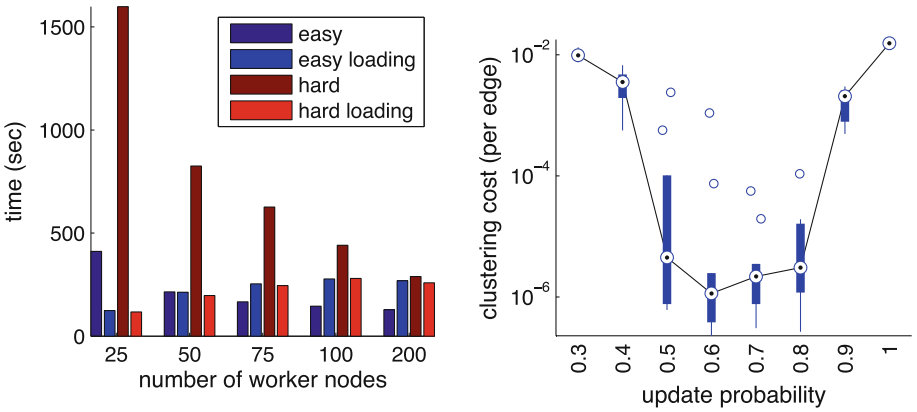

Fig. 8 *Left* Running time of the OCC- MR algorithm (without the time spent on loading the input) using the FLICKR dataset as a function of cluster size for two parameter settings, where easy: $k = 50$, $p = 5$, and hard: $k = 200$, $p = 20$. Time spent on loading and distributing the input across worker nodes is shown separately. *Right* Clustering cost (per edge) as a function of the probability of a node being updated in an iteration of the OCC- MR algorithm. (Note the logarithmic scale on the y axis.) The experiment was run with the FLICKR dataset, setting $k = 100$ and $p = 10$. Computation was stopped after 75 iterations, or when the clustering cost was below 10^{-7}. The plot is based on 10 independent trials

update probability, we allow the algorithm to take "longer" steps but also increase the risk of getting stuck in a local optima. The main purpose of this experiment is to show that by setting the update probability to a suitable value, the algorithm will find a very good solution in a reasonable number of supersteps. To this end, we set an upper bound of 75 on the number of supersteps and observe the cost of the solution at this point.

The results of this experiment are shown in Fig. 8, please see the caption of the figure for details on the other parameters. Overall, we observe that the update probability plays a very important role. Clearly, the "fully parallelized" variant (update probability 1.0) is unable to find good solutions. And as expected, with low update probabilities, the allotted 75 supersteps are not enough for the algorithm to converge. With this particular dataset (FLICKR) and the given parameters, a suitable update probability seems to be around 0.6, although the differences between 0.5 and 0.8 are very small.

Figure 9 illustrates this phenomenon in more detail for different update probabilities. We observe that during the first 10–15 supersteps, the clustering cost decreases roughly by one order of magnitude independent of the update probability. After this, the algorithm enters a phase with very slow convergence, until it starts to head toward a low-cost solution. However, the precise moment when this happens strongly depends on the update probability, with 0.6 having the best performance in this particular example. Note that this experiment cannot rule out the possibility of the algorithm eventually breaking out of the phase of slow convergence also when the update probability is equal to 1.0. It merely suggests that given a budget on the number of supersteps performance can be considerably improved by adjusting the update probability to a suitable value.

7 Applications

In this section, we discuss the results of applying our method for clustering three different datasets: movement trajectories, proteins, and documents. Our objective is to provide evidence that our algorithms not only give high-quality clusters but they also find meaningful overlaps.

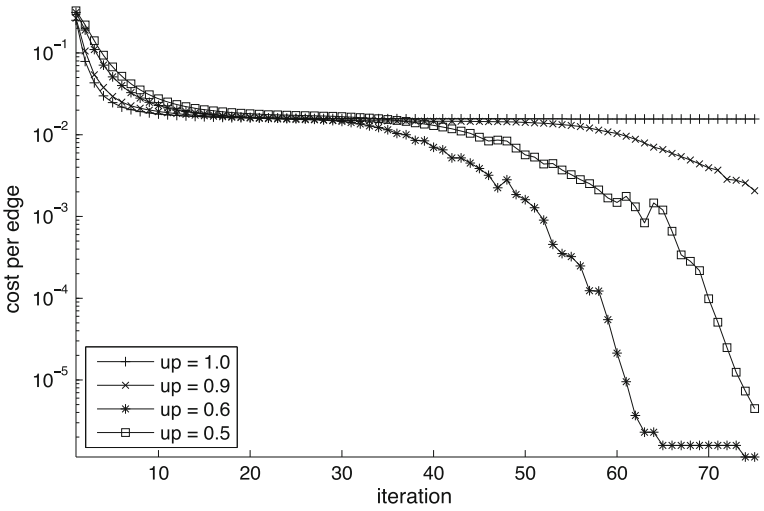

Fig. 9 Medians of 10 iterations of cost-per-edge as a function of iteration number when running the OCC- MR algorithm on the FLICKR dataset using different update probabilities. Note the logarithmic scale on the y axis

7.1 Overlapping clustering of trajectories

Spatio-temporal data mining is an active research area with a wide variety of applications, such as mobility management, video surveillance, mobile social networks, and ecology. The basic entities in analysis are usually trajectories of moving objects, that is, sequences $\langle (x_0, y_0, t_0), \ldots, (x_n, y_n, t_n) \rangle$ of observations in space and time. Trajectories may have different lengths, and therefore, it is not feasible to simply view them as vectors and use standard distance functions. Moreover, the different nature of space and time implies different granularity and resolution issues. While there has been extensive research on clustering trajectories [26,38,44], to the best of our knowledge, the problem of overlapping clustering of trajectories has been largely left unexplored.

As a motivating example of why overlapping clustering of trajectories is a meaningful task, consider a well-shaped cluster C_i of trajectories of GPS-equipped cars going from a south-west suburb to the city center between 8 and 9 am, and another cluster C_j moving inside the city center along a specific path, from 3 to 4 pm. Now consider a trajectory that moves from the south-west suburb to the city center in the morning, and within the city center in the afternoon: it is very natural for this trajectory to belong to both clusters C_i and C_j.

Developing overlapping clustering of trajectories within our framework is fairly straightforward. We just need to compute a distance (or similarity) for every pair of trajectories. We choose the edr distance [14], which has the important feature of being *time-tolerant*, that is, it is defined even between two trajectories of different length. We normalize the edr distance to stay in the range [0, 1], and to make it suitable for our method, we convert distances to similarities: $\mathrm{sim}(u, v) = 1 - \mathrm{edr}(u, v)$.

We use animal movement data generated by the Starkey project and analyzed over the years by many zoologists.[5] The dataset contains the radio-telemetry locations of elks, deer, and cattle from 1993 to 1996. We select to work on all the three species together and only

[5] http://www.fs.fed.us/pnw/starkey/.

Table 1 Result of clustering on the STARKEY93 dataset

C_1	C_2	C_3	C_4	C_5	Clusters
16E 6D	3E 2D	3E	5E	3E 2D	C_1
	4E 2D 38C	∅	1E	30C	C_2
		13E 6D	9E	∅	C_3
			21E 2D	∅	C_4
				3E 2D 33C	C_5

On the diagonal, we report the population of each cluster, using C, D, and E to distinguish between the number of cattle, deer and elks, respectively. In the non-diagonal cells of the matrix, we report the population of the overlap among two clusters

for 1993. The dataset contains only 88 trajectories, corresponding to 33 elks, 14 deer, and 41 cattle, but each trajectory is very long. The whole dataset has 79,987 (x, y, t) observations; an average of 909 observations per trajectory.[6]

Studying overlapping trajectory clusters in this context might be of zoological interest, as discussed by Coe et al. [17]:

> These three species have important social, ecological, and economic values. Understanding their interspecific interactions may clarify two recurring issues in their management: competition for food and competition for space, both of which may result in decreased animal fitness. […] Accurate predictions of ungulate distributions over time and space may help managers regulate densities and understand effects of specific ungulates on ecosystem processes. […] Overlapping distributions could be evidence for competition or dependence. Non-overlap could be an expression of active avoidance or ecological separation, which occurs when two species evolved together.

In the following, we report the results of a clustering obtained with our overlapping correlation-clustering framework with $k = 5$ and $p = 2$ (i.e., 5 clusters, and each trajectory can belong to at most 2 clusters). Interestingly, in this context, both definitions of similarity between sets of elements (i.e., Jaccard and set-intersection indicator) yield very consistent clusterings, with just few elements assigned to a different cluster.

In Table 1, we report the results obtained with the Jaccard definition. It is important to recall that we are only clustering the trajectories, that is, sequences of spatio-temporal observations without any additional information. In particular, we do not use information of the species to which a trajectory corresponds. However, we can observe that two clusters C_2 and C_5 contain mostly cattle, and few individuals of the other species, while the other 3 clusters do not contain any cattle. In particular, cluster C_4 contains mainly elks.

This is in line with the zoological domain knowledge. Cattle are introduced in late spring or early summer for the grazing season. During summer elks and deer avoid cattle, but in late summer and fall, the three species overlap in some areas, due to competition for forage resources that have become depleted [17].

It is interesting to observe that cluster C_5 is almost perfectly covered by clusters C_1 and C_2. In Fig. 10, we plot the trajectories of each cluster in space and time. We can see that the clusters C_3 and C_4, which contain only elks and deer, cover most of the available area, while clusters C_1, C_2, and C_5 contain individuals that almost never enter in the area with

[6] We computed the edr distance using the following space- and time-tolerance parameters: $\Delta x = \Delta y = 2.5$K and $\Delta t = 500$K.

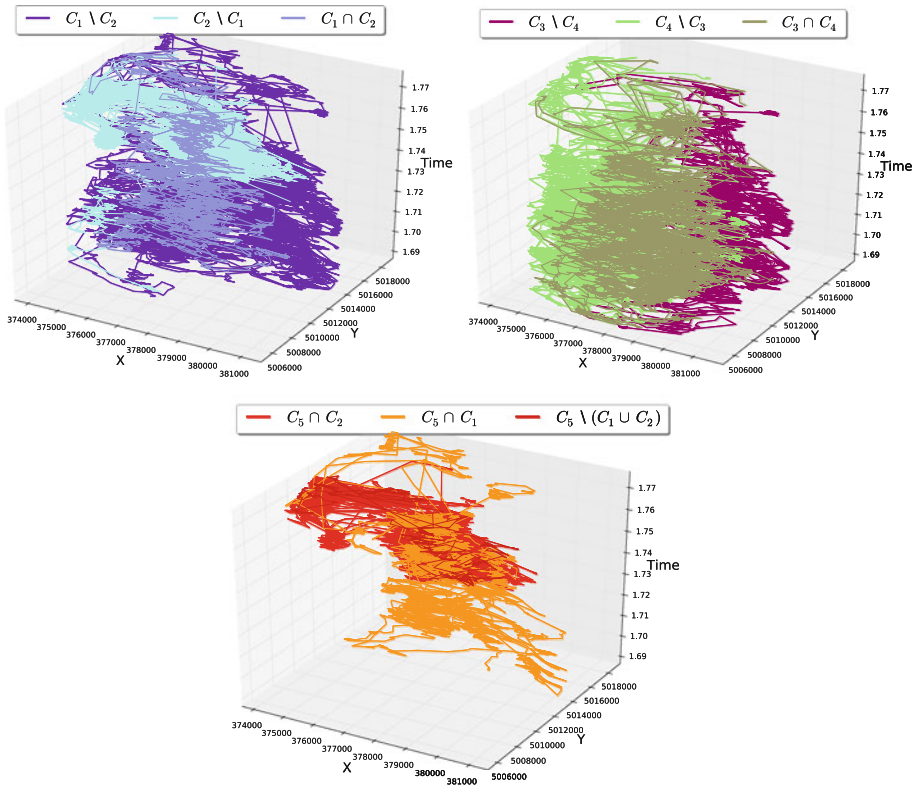

Fig. 10 Plots in space and time of all the trajectories in the five clusters obtained in the STARKEY93 dataset. Time axis is $\times 10^8$ and it counts elapsed seconds since 12/31/87, which represents a starting point of the Starkey Ungulate Research project

Y smaller than 5,014K. The cluster C_2 contains almost all cattle that enter the area only in late spring (time greater than 1.724), plus few elks and deer that belong to C_1 as well, which move closer to the cattle than what the others in C_3 and C_4 do.

Summarizing: clusters C_3 and C_4 contain elks and deer that stay away from cattle; cluster C_1 also contains only elks and deer, but those moves in higher Y coordinates where cattle also move; cluster C_2 is the cattle cluster and it contains few elks and deer; finally cluster C_5 is a mixed cluster that overlaps with C_2 only for the cattle and with C_1 for the elks and deer.

7.2 Overlapping clustering of protein sequences

An important problem in genomics is the study of evolutionary relatedness of proteins using sequence data. We use our algorithms to cluster proteins to homologous groups given pairwise similarities of amino-acid sequences. Such similarities are computed by the sequence alignment tool BLAST [3]. We follow the approach of Paccanaro et al. [46] and Nepusz et al. [45] and compare the computed clustering against a ground truth given by SCOP, a manually crafted taxonomy of proteins [43]. The SCOP taxonomy is a tree with proteins at the leaf nodes. The ground-truth clusters used in the experiments are subsets of the leafs, that is, proteins, rooted at different SCOP *superfamilies*. These are nodes on the 3rd level below the root.

Table 2 Precision, recall, and their harmonic mean F-score, for non-overlapping clusterings of protein sequence datasets computed using SCPS and the overlapping clustering algorithms

Dataset	BASELINE	SCPS	OCC- ISECT	OCC- JACC
	Prec	Prec/recall/F-score	Prec/recall/F-score	Prec/recall/F-score
D_1	0.21	0.56 / 0.82 / 0.664	0.70 / 0.67 / **0.683**	0.57 / 0.55 / 0.561
D_2	0.17	0.59 / 0.89 / 0.708	0.86 / 0.83 / **0.844**	0.64 / 0.63 / 0.637
D_3	0.38	0.93 / 0.88 / **0.904**	0.81 / 0.43 / 0.558	0.73 / 0.39 / 0.505
D_4	0.14	0.30 / 0.64 / 0.408	0.64 / 0.56 / **0.598**	0.44 / 0.39 / 0.412

BASELINE is the precision of a baseline that assigns all sequences to the same cluster
The highest F-score is indicated in bold

We compare our algorithm with the SCPS algorithm [45,46], a spectral method for clustering biological sequence data. The experiment is run using four datasets from Nepusz et al. [45]. Those datasets contain pre-computed sequence similarities in the range [0, 1] for various subsets of proteins, together with the ground-truth clusterings.[7] The dataset D_1 contains 669 sequences and 5 ground-truth clusters, D_2 contains 587 sequences and 6 clusters, D_3 contains 567 sequences and 5 clusters, and D_4 contains 654 sequences and 8 clusters.

To compare our algorithms with the SCPS algorithm, we first computed non-overlapping clusterings of all four datasets. All algorithms were given the correct number of clusters as a parameter. Results of comparing the resulting clusters with the ground-truth clusters, in terms of precision, recall, and F-score are shown in Table 2. The SCPS algorithm has a higher recall in every case, but with datasets D_1, D_2, and D_4, the OCC- ISECT algorithm achieves a substantially higher precision. In practice, this means that if OCC- ISECT assigns two sequences to the same cluster, they belong to the same cluster also in the ground truth with higher probability than when using SCPS.[8]

We also conduct a more fine-grained analysis of the results using the SCOP taxonomy. In fact, different cluster "errors" should have different cost, depending on the distance on the taxonomy. If we misplace a protein in two clusters that are extremely close in the taxonomy, the error should have a small cost. Following this intuition, we define the SCOP similarity between two proteins as follows:

$$\mathrm{sim}(u, v) = \frac{\mathrm{d}(\mathrm{lca}(u, v))}{\max(\mathrm{d}(u), \mathrm{d}(v)) - 1}, \tag{8}$$

where $\mathrm{d}(u)$ is the depth of a node in the tree (the root is at depth 0), and $\mathrm{lca}(u, v)$ denotes the lowest common ancestor of the nodes u and v. The above similarity has a value of zero if $\mathrm{lca}(u, v)$ is the root, and a value of one if $\mathrm{lca}(u, v)$ is the common parent of u and v. Based on $\mathrm{sim}(u, v)$, we define the cost of a clustering by paying $1 - \mathrm{sim}(u, v)$ for two proteins that ends in the same cluster, and $\mathrm{sim}(u, v)$ for two proteins belonging to different clusters, as in Eq. (1) and similarly to Eq. (2) for the overlapping clusterings.

The results of Table 3 suggest that the overlapping correlation-clustering algorithms find a clustering that is better in agreement with the SCOP taxonomy than the clusterings found by SCPS. However, while allowing overlaps is beneficial, we do not observe a significant improvement as the node-specific constraint p is increased. Moreover, we observe that only a small number of proteins are assigned to multiple clusters. We conjecture that this is due

[7] The datasets and the SCPS implementation are available at http://www.paccanarolab.org/software/scps/.

[8] Note that these numbers are not directly comparable with the ones reported by Nepusz et al. [45] as they define precision and recall in a slightly different way.

Table 3 Comparing clustering cost based on distance on the SCOP taxonomy, for different values of p, the maximum number of labels per protein

	SCPS	$p = 1$	$p = 2$	$p = 3$
OCC- ISECT				
D_1	0.231	0.196	0.194	0.193
D_2	0.188	0.112	0.107	0.106
D_3	0.215	0.214	0.214	0.231
D_4	0.289	0.139	0.133	0.139
OCC- JACC				
D_1	0.231	0.208	0.202	0.205
D_2	0.188	0.137	0.130	0.127
D_3	0.215	0.243	0.242	0.221
D_4	0.289	0.158	0.141	0.152

to the similarities produced by BLAST, which imply very well defined clusters in most of the cases. Nevertheless, it is worth noting that our methods, regardless of the parameters used, do not find unnecessarily large overlaps, when this is not dictated by the data.

7.3 Overlapping clustering of documents

As documents can belong to multiple topics, using our framework to find overlapping clusters of documents is a well-motivated application. We run our overlapping-clustering algorithms on a dataset of 1,500 papers published at NIPS workshops.[9] The bag-of-words dataset was converted to similarities as follows. First, we compute the Jaccard coefficient between the bags-of-words of every pair of documents, but we only consider terms that appear at least five times in the documents. Second, we convert the Jaccard coefficients to binary labels with a threshold of 0.05. The threshold was set to a value slightly larger than zero, because for most pairs of papers, the intersection of word sets was non-empty. We tried various combinations of k and p parameters with OCC- ISECT and found that $k = 6$ and $p = 2$ gives the lowest cost of 0.182 per edge. Allowing overlaps is essential, as with $k = 6$ and $p = 1$, we obtain a cost of 0.225 per edge.

Table 4 shows the top-10 words in terms of `tf·idf` of each cluster. Term frequency (`tf`) is the number of documents in a cluster that contain the given term, while inverse document frequency (`idf`) is defined in the standard manner over the entire collection. Clearly, the clusters represent different topics, with C_1 corresponding to *algorithms*, C_2 to *hardware*, C_3 to *pattern recognition*, C_4 to *artificial neural networks*, and C_6 to *biological neural networks*. Cluster C_5 is more difficult to label given the top-10 words, but a deeper inspection reveals it to contain keywords related to models of (visual) perception, and papers such as *A Model for Resolution Enhancement (hyperacuity) in sensory representation* (Zhang and Miller, NIPS 1988) and *Scale Mixtures of Gaussians and the Statistics of Natural Images* (Wainwright, NIPS 1999).

Table 5 shows the co-occurrence counts of each cluster together with cluster size on the diagonal. From this, we immediately see that many papers have been assigned to more than one cluster. Overlap is therefore not an exception, but indeed, most papers naturally cover two topics. We can also observe that the cluster C_5 is kind of an outlier as it has no overlap with

[9] http://archive.ics.uci.edu/ml/datasets/Bag+of+Words.

Table 4 Top-10 words in terms of `tf·idf` in each cluster computed from the NIPS data

C_1	Method algorithm problem error parameter distribution probability vector space number
C_2	Circuit chip analog voltage vlsi current implementation neuron signal transistor
C_3	Training performance data test number task recognition set error image
C_4	Unit output weight layer input training hidden pattern system information
C_5	John david michael visual natural connectionist images scale left representation
C_6	Cell neuron activity response visual cortex firing synaptic inhibitory stimulus

Table 5 Co-occurrence counts of NIPS papers in different overlapping clusters

	C_1	C_2	C_3	C_4	C_5	C_6
C_1 (algorithms)	868	1	253	472	0	3
C_2 (hardware)	1	97	0	43	0	29
C_3 (pattern recognition)	253	0	523	270	0	0
C_4 (artificial neural networks)	472	43	270	946	0	111
C_5 (perception)	0	0	0	0	52	0
C_6 (biological neural networks)	3	29	0	111	0	186

the other clusters. This might be explained by the somewhat different terminology used in sensory-perception research. Also, some combinations of clusters are very rare. For example, only one paper has been assigned to both C_1 (algorithms) and C_2 (hardware). This paper happens to be *Constrained Optimization Applied to the Parameter Setting Problem for Analog Circuits* by Kirk et al. (NIPS 1991). Likewise, the combination C_1 (algorithms) and C_6 (biological neural networks) contains only three papers, which are *Optimal Sampling of Natural Images: A Design Principle for the Visual System?* (Bialek et al., NIPS 1991), *Refractoriness and Neural Precision* (Berry and Meister, NIPS 1998), and *Population Decoding Based on an Unfaithful Model* (Wu et al., NIPS 2000).

Intuitively, the clusters found seems to be topically coherent and meaningful, as well as their overlaps.

8 Related work

We review research related to our paper, grouped in four different themes, correlation clustering, overlapping clustering, constraint clustering, and applications.

8.1 Correlation clustering

The problem of CORRELATION- CLUSTERING was first defined by Bansal et al. [6]. In their definition, the input is a complete graph with positive and negative edges, and no threshold on the maximum number of clusters is given. The objective is to partition the nodes of the graph so as to minimize the number of positive edges that are cut, and the number of negative edges that are not cut—corresponding to our problem instance $(\mathtt{b}, H, 1)$. This is an APX-hard optimization problem that has received a great deal of attention in the field of theoretical computer science [1,2,13,21,29,53].

The best-known approximation algorithm for this problem has been obtained by Charikar et al. [13] who give an LP-based algorithm that achieves an approximation factor of 4. The algorithm of Charikar et al. combined with a reduction by Bansal et al. [6, Theorem 23] provides a deterministic 9-approximation algorithm for correlation clustering when the edge weights satisfy the *probability condition* — that is, for every edge (i, j), the cost for taking that edge is $X_{ij} \in [0, 1]$, while the cost for splitting an edge is $1 - X_{ij}$—even if they do not satisfy the triangle inequality.

When the edge weights are arbitrary, the problem is equivalent to the multicut problem, as shown by Demaine et al. [21], and there is a $O(\log n)$-approximation bound. If one considers the corresponding maximization problem, that is, maximize the agreements rather than minimize disagreements, then the situation is much better. Even in the case of graphs with arbitrary edge weights, there is a 0.76-approximation algorithm using semi-definite programming [13,53].

Recently, Ailon et al. [1] considered a variety of correlation-clustering problems. They proposed an algorithm that achieves expected approximation ratio 5 if the weights obey the probability condition. If the weights X_{ij} obey also the triangle inequality, then the algorithm achieves expected approximation ratio 2.

Giotis and Guruswami [29] consider correlation clustering when the number of clusters is given, while Ailon and Liberty [2] study a variant of correlation clustering where it exists an unknown correct clustering of the data and the goal is to minimize the number of disagreements between the produced clustering and the ground-truth clustering. Correlation clustering is also closely related to the problem of clustering aggregation [28].

To the best of our knowledge, no previous work has consider the possibility of overlaps in correlation clustering, that is, (r, H, p), with $p > 1$.

8.2 Overlapping clustering

In 1979, Shepard and Arabie introduced the ADCLUS algorithm [52] for additive clustering, which perhaps can be considered the first overlapping-clustering method. Additive clustering assumes that the similarity of any two objects is a simple additive function of weights associated with properties that are shared by both objects. Based on this model ADCLUS estimates (i) which subsets of a given set of objects correspond to positively weighted properties, and (ii) the numerical values of those weights. The method, which has been later applied in the marketing domain [4], subsumes hierarchical clustering as a special case and can be regarded as a discrete analog of principal components analysis.

Regardless of these ancient roots, overlapping clustering has not seen many results in the last decades. Recently, Xiong et. al. [59] study clustering models with the aim of preserving predefined structure (patterns) in the data. This will often produce overlapping clusters, as a pattern can belong to multiple clusters in a natural way. SimClus [30] is a very recent set-cover-based clustering algorithm that can produce overlapping clusters. SimClus also uses pairwise similarities of the objects, but technical fundamentals and underlying assumptions of the algorithm are completely different from our approach. In [30], the objective is to assign an object to at least one cluster so that the similarity between every object and its cluster representative is lower bounded some predefined threshold.

One close sibling is fuzzy clustering [10,32,41], where each data point has a membership value in all the clusters. In this context, cluster membership is soft, while we are interested in hard cluster assignments. Obviously, a hard (and overlapping) cluster assignment can be obtained by thresholding membership values. The prototypical fuzzy clustering method is *fuzzy c-means* [31], which is essentially a soft version of k-means.

Driven by the need to cluster microarray gene expression data, various methods for overlapping clustering [9,50] and overlapping bi-clustering (or co-clustering) [15,25] have been produced in bioinformatics [39].

Recently, mixture models approach have been generalized to allow overlapping clusters. Banerjee et al. [5] generalize the work of Segal et al. [50] that was specifically designed for gene expression data: while the original model was working with constant variance Gaussians, Banerjee et al. generalize it to work with any regular exponential family distribution, and corresponding Bregman divergence. The work of Banerjee et al. [5] has later been extended to co-clustering by Shafiei and Milios [51]. Multiplicative mixture models have been proposed as a framework for overlapping clustering by Fu and Banerjee [24].

Our work distinguishes from this body of research as it develops within the correlation-clustering framework, and thus, it has a different input and different objective functions.

8.3 Constrained clustering

The binary version of correlation clustering $(\mathfrak{b}, H, 1)$ with positive and negative links can be seen as a soft instance of clustering with *must-link* (ML) and *cannot-link* (CL) constraints, for which exists an extensive literature [7,8,35,56,57]. However, in constrained clustering, you have distances among the objects and some ML and CL constraints, while in correlation clustering, distances and constraints coincide and they are all you have.

Interesting results by Davidson and Ravi [18,19] show that given a set of ML and CL constraints determining whether there exists a feasible clustering is **NP**-hard, as well as intractable is the problem of, given a set of constraints for which no feasible solution exists for a particular number k of clusters, identifying the minimum number of constraints to be relaxed so that a feasible solution exists.

While presented in the context of ML and CL constraints, the work by Scripps and Tan [49] essentially deals with a binary version of correlation clustering, and it adopts "cloning" to fix the problem of bridge nodes (or "bad triplets" in the terminology of Ailon et al. [1]). Cloning essentially means allowing overlaps; therefore, the problem they deal with is our binary version of overlapping correlation clustering. In the notation we introduce in this paper, their problem is exactly $(\mathfrak{b}, I, k)$ with no predefined number of clusters. However, as implied by Corollary 1, such a problem always admits a straightforward zero-cost solution. Scripps and Tan [49] are interested only in zero-cost solutions, while trying to minimize the number of clones (i.e., overlaps). Instead, we consider the problem of finding minimum cost solutions with a prefixed number of clusters, or with a constraint on the maximum number of clusters an object can belong.

8.4 Non-negative matrix factorization

In the non-negative matrix factorization (NMF) problem [37], the goal is to decompose a given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ into the product of two non-negative matrices $\mathbf{W} \in \mathbb{R}^{m \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times n}$, so that the product $\mathbf{WH}$ approximates as well as possible the original matrix $\mathbf{A}$. The quality of the decomposition is typically measured with the Frobenius matrix norm $||\mathbf{A} - \mathbf{WH}||_F^2$. It is also assumed that the dimension k is much smaller than the dimensions m and n. In the symmetric version of the non-negative matrix factorization problem (SNMF) [22,33], the input matrix A is symmetric, that is, $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{A} = \mathbf{A}^T$, and the goal is to find a non-negative matrix $\mathbf{X} \in \mathbb{R}^{k \times n}$ so that $\mathbf{A} \approx \mathbf{X}^T\mathbf{X}$.

The SNMF problem has many similarities with our problem. Consider that the input matrix $\mathbf{A} = [a_{ij}]$ represents the pairwise object similarities, that is, $a_{ij} = s(i, j)$. The factor matrix $\mathbf{X}$ can be viewed as mapping from each object $i \in V$ to a k-dimensional vector, namely, to the ith column $\mathbf{x}_i$ of the matrix $\mathbf{X} = [\mathbf{x}_1 \ldots \mathbf{x}_n]$. Should the matrix $\mathbf{X}$ have 0–1 values, the mapping of $i \in V$ to $\mathbf{x}_i$ can be interpreted as a membership vector to k clusters, and thus, the solution to the SNMF problem has a natural interpretation as an overlapping clustering. Furthermore, the objective function of the SNMF problem would be $||\mathbf{A} - \mathbf{X}^T\mathbf{X}||_F^2 = \sum_{i,j} |a_{ij} - \langle \mathbf{x}_i, \mathbf{x}_j \rangle|^2$, where $\langle \cdot, \cdot \rangle$ denotes the dot-product operation. A similar problem definition has been considered in the statistics literature, with the goal of modeling graphs using dot-product representations [48].

On the other hand, there are also many differences of the approaches described above with the problem we consider in this paper. First, we do not consider the dot-product between representation vectors; instead, we focus on the measures of set-intersection and Jaccard. Second, in the above-mentioned problems, there is no integrality constraint for the entries of the solution matrix $\mathbf{X}$, and thus, there is no immediate interpretation of the solution as overlapping clustering. One way to obtain integral solutions is to *round* the fractional solutions using a threshold parameter. This approach is actually explored in our experiments in Sect. 6.1.1, where it is shown that, compared against ground-truth, our overlapping-clustering algorithms outperform the SNMF-based algorithms for a wide range of the rounding threshold parameter.

8.5 Applications

Developments in overlapping clustering have mainly been driven by the concrete needs of applications. We already discussed the case of microarray gene expression data. Other application domains where overlapping clusters occur naturally are text mining [36], social networks, and social media.

Partitioning the nodes of a social network is a problem typically referred to as *community detection* and which has been extensively studied in particular in the literature of complex-networks analysis [23]. However, only few researchers have addressed the problem of detecting overlapping communities. For a survey on the topic please, see the work of Fortunato [23, Section 11].

The best-known approach to detecting overlapping communities is the CFinder algorithm based on *clique percolation* [47]. According to this method, communities are discovered by finding k-cliques and merging them when they share $k - 1$ nodes. As a node can belong to multiple cliques, the method generates overlapping communities.

Tang and Liu [54] start from the observation that, while users are naturally involved in more than a community, edges usually belong to a single community. Therefore, they cluster edges instead of nodes. This approach results in overlapping communities of nodes. Following up this work, Wang et al. [58] study the problem of discovering overlapping groups in social media. They propose a co-clustering framework based on users and tags. Users are not connected trough a social network, instead they are implicitly connected by their common interests that are expressed by the tags they use. The method creates co-clusters of tags and users.

9 Conclusions and future work

We present a novel formulation for overlapping clustering in the context of correlation clustering. The main idea is to map data points to a small set of labels, so that similarities between

label sets preserve the original similarities between data points. In this respect, our problem statement has a flavor of dimensionality reduction, yet the labels obtained from the mapping can be used to express membership in overlapping clusters.

To solve the resulting optimization problem, we propose a local-search algorithm, initialized with the labeling obtained by a graph coloring heuristic. The iterative step of the local-search algorithm optimizes the labels of one object, given the labels of all other objects. We apply such a local optimization until we obtain a solution that no further improvement can be made. The local step of our iterative algorithm, optimizing the labels of one single object given the labels of all other objects turns out to be an interesting sub-problem.

We experiment with the proposed algorithm on real datasets with overlapping clusters. Our evaluation shows that our algorithm produces overlapping clusters that resemble the ground truth. We also experiment with the idea of speeding up the algorithm by randomly eliminating from consideration pairs of data points. Our results show that significant amount of pruning can be achieved without degrading the quality of the solution.

We believe that the framework we introduced can be extended in many interesting directions.

First, we plan to instantiate our approach in different application domains. Beyond finding novel applications, we would like to investigate different approaches to solve the optimization problems we propose, for example, using non-local-search algorithms such as the idea based on relaxed graph coloring, mentioned in Sect. 3. Another interesting opportunity, worth of further investigation, is to apply graph coloring solutions for the initialization step, as discussed in Sect. 4.4. We plan also to study, both from a theoretical and practical points of view, the problem of actively selecting the pairs of objects for which to require the similarity value.

Furthermore, we would like to study alternative problem definitions, using different objective functions, and introducing additional constraints. Finally, an interesting open problem is to design an approximation algorithm for the JACCARD- TRIANGULATION problem.

Acknowledgments This research was partially supported by the Torres Quevedo Program of the Spanish Ministry of Science and Innovation, co-funded by the European Social Fund, and by the Spanish Centre for the Development of Industrial Technology under the CENIT program, project CEN-20101037, "Social Media" (http://www.cenitsocialmedia.es/).

References

1. Ailon N, Charikar M, Newman A (2005) Aggregating inconsistent information: ranking and clustering. In: Proceedings of the ACM symposium on theory of computing (STOC)
2. Ailon N, Liberty E (2009) Correlation clustering revisited: the "true" cost of error minimization problems. In: Automata, languages and programming, 36th international colloquium (ICALP)
3. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215(3):403–410
4. Arabie P, Carroll JD, DeSarbo W, Wind J (1981) Overlapping clustering: a new method for product positioning. J Mark Res 18(3):310–317
5. Banerjee A, Krumpelman C, Ghosh J, Basu S, Mooney RJ (2005) Model-based overlapping clustering. In: Proceedings of the 11th ACM SIGKDD international conference on knowledge discovery and data mining (KDD)
6. Bansal N, Blum A, Chawla S (2004) Correlation clustering. Mach Learn 56(1–3):89–113
7. Basu S, Banerjee A, Mooney RJ (2004) Active semi-supervision for pairwise constrained clustering. In: Proceedings of the Fourth SIAM international conference on data mining (SDM)
8. Basu S, Bilenko M, Mooney RJ (2004) A probabilistic framework for semi-supervised clustering. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD)

9. Battle A, Segal E, Koller D (2004) Probabilistic discovery of overlapping cellular processes and their regulation. In: Proceedings of the 8th international conference on research in computational molecular biology (RECOMB)
10. Bezdek JC, Pal SK (eds) (1992) Fuzzy models for pattern recognition—methods that search for structures in data. IEEE Press, New York
11. Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw 30(1–7):107–117
12. Broder AZ, Charikar M, Frieze AM, Mitzenmacher M (1998) Min-wise independent permutations. In: Proceedings of the 13th annual ACM symposium on theory of computing (STOC)
13. Charikar M, Guruswami V, Wirth A (2003) Clustering with qualitative information. In: Proceedings of the IEEE symposium on foundations of computer science (FOCS)
14. Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data (SIGMOD'05)
15. Cheng Y, Church GM (2000) Biclustering of expression data. In: Proceedings of the eighth international conference on intelligent systems for molecular biology (ISMB)
16. Chierichetti F, Kumar R, Pandey S, Vassilvitskii S (2010) Finding the jaccard median. In: Proceedings of the 21st annual ACM-SIAM symposium on discrete algorithms (SODA)
17. Coe PK, Johnson BK, Stewart KM, Kie JG (2004) Spatial and temporal interactions of elk, mule deer, and cattle. In: Transactions of the 69th North American wildlife and natural resources conference, pp 656–669
18. Davidson I, Ravi SS (2005) Clustering with constraints: feasibility issues and the k-means algorithm. In: Proceedings of the Fifth SIAM international conference on data mining (SDM)
19. Davidson I, Ravi SS (2007) Intractability and clustering with constraints. In: Proceedings of the 24th international conference on machine learning (ICML)
20. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
21. Demaine ED, Emanuel D, Fiat A, Immorlica N (2006) Correlation clustering in general weighted graphs. Theor Comput Sci 361:172–187
22. Ding C, He X, Simon HD (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proceedings of the SIAM data mining conference
23. Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174
24. Fu Q, Banerjee A (2008) Multiplicative mixture models for overlapping clustering. In: Proceedings of the 8th IEEE international conference on data mining (ICDM)
25. Fu Q, Banerjee A (2009) Bayesian overlapping subspace clustering. In: Proceedings of the 9th IEEE international conference on data mining (ICDM)
26. Gaffney S, Smyth P (1999) Trajectory clustering with mixtures of regression models. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '99
27. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., San Francisco
28. Gionis A, Mannila H, Tsaparas P (2007) Clustering aggregation. TKDD 1(1):Article 4
29. Giotis I, Guruswami V (2006) Correlation clustering with a fixed number of clusters. In: Proceedings of the seventeenth annual ACM-SIAM symposium on discrete algorithms (SODA)
30. Hasan M, Salem S, Zaki M (2011) Simclus: an effective algorithm for clustering with a lower bound on similarity. Knowl Inf Syst 28:665–685
31. Hathaway RJ, Davenport JW, Bezdek JC (1989) Relational duals of the c-means clustering algorithms. Pattern Recognit 22(2):205–212
32. Hathaway RJ, Hu Y (2009) Density-weighted fuzzy c-means clustering. IEEE T Fuzzy Syst 17(1): 243–252
33. He Z, Xie S, Zdunek R, Zhou G, Cichocki A (2011) Symmetric nonnegative matrix factorization: algorithms and applications to probabilistic clustering. IEEE Trans Neural Netw 22(12):2117–2131
34. Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the 13th annual ACM symposium on theory of computing (STOC)
35. 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: Proceedings of the nineteenth international conference on machine learning (ICML)
36. Kobayashi M, Aono M (2006) Exploring overlapping clusters using dynamic re-scaling and sampling. Knowl Inf Syst 10:295–313
37. Lee DD, Seung HS (2001) Algorithms for Non-negative Matrix Factorization. In: Advances in neural information processing systems 13:556–562
38. Lee JG, Han J, Whang KY (2007) Trajectory clustering: a partition-and-group framework. In: Proceedings of the 2007 ACM SIGMOD international conference on management of data, SIGMOD '07

39. Madeira SC, Oliveira AL (2004) Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Trans Comput Biol Bioinform 1(1):24–45
40. Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: SIGMOD conference, pp 135–146
41. Mei JP, Chen L (2010) Fuzzy clustering with weighted medoids for relational data. Pattern Recognit 43(5):1964–1974
42. Miettinen P (2008) On the positive-negative partial set cover problem. Inf Process Lett 108(4):219–221
43. Murzin A, Brenner S, Hubbard T, Chothia C (1995) Scop—a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol 247(4):536–540
44. Nanni M, Pedreschi D (2006) Time-focused clustering of trajectories of moving objects. J Intell Inf Syst 27(3):267–289
45. Nepusz T, Sasidharan R, Paccanaro A (2010) Scps: a fast implementation of a spectral method for detecting protein families on a genome-wide scale. BMC Bioinform 11(1):120
46. Paccanaro A, Casbon JA, Saqi MAS (2006) Spectral clustering of protein sequences. Nucleic Acids Res 34(5):1571–1580
47. Palla G, Derenyi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature
48. Scheinerman ER, Tucker K (2010) Modeling graphs using dot product representations. Comput Stat 25(1):1–16
49. Scripps J, Tan PN (2006) Clustering in the presence of bridge-nodes. In: Proceedings of the sixth SIAM international conference on data mining (SDM)
50. Segal E, Battle A, Koller D (2003) Decomposing gene expression into cellular processes. In: Proceedings of the 8th Pacific symposium on biocomputing (PSB)
51. Shafiei MM, Milios EE (2006) Model-based overlapping co-clustering. In: Proceedings of the fourth workshop on text mining
52. Shepard RN, Arabie P (1979) Additive clustering: representation of similarities as combinations of discrete overlapping properties. Psychol Rev 86(2):87–123
53. Swamy C (2004) Correlation clustering: maximizing agreements via semidefinite programming. In: Proceedings of the ACM-SIAM symposium on discrete algorithms (SODA)
54. Tang L, Liu H (2009) Scalable learning of collective behavior based on sparse social dimensions. In: Proceedings of the 18th ACM conference on information and knowledge management (CIKM)
55. Valiant LG (1990) A bridging model for parallel computation. Commun ACM 33(8):103–111
56. Wagstaff K, Cardie C (2000) Clustering with instance-level constraints. In: Proceedings of the 17th international conference on machine learning (ICML)
57. Wagstaff K, Cardie C, Rogers S, Schrödl S (2001) Constrained k-means clustering with background knowledge. In: Proceedings of the 18th international conference on machine learning (ICML)
58. Wang X, Tang L, Gao H, Liu H (2010) Discovering overlapping groups in social media. In: The 10th IEEE international conference on data mining (ICDM)
59. Xiong H, Steinbach M, Ruslim A, Kumar V (2009) Characterizing pattern preserving clustering. Knowl Inf Syst 19:311–336

Author Biographies

Francesco Bonchi is a senior research scientist at Yahoo! Research in Barcelona, Spain, where he is part of the Web Mining Group. His recent research interests include mining query-logs, social networks, and social media, as well as the privacy issues related to mining these kinds of sensible data. In particular, his main interest nowadays is to develop data mining methods for the analysis of information an influence spread. He gave a keynote talk at WI-IAT 2011 on this topic. He has been program co-chair of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2010), and various workshops on social web mining, privacy, and other topics in data mining.

Aristides Gionis is a senior research scientist in Yahoo! Research, Barcelona. He received his Ph.D. from the Computer Science department of Stanford University in 2003 and until 2006 he has been a researcher at the University of Helsinki, Finland. He is serving on the editorial boards of TKDE and KAIS, and he has been in the program committee of numerous premier conferences. His research interests include algorithms for data analysis and applications in the web domain.

Antti Ukkonen is a postdoctoral researcher at Yahoo! Research, Barcelona. He received his doctoral degree from the Department of Information and Computer Science of Aalto University in 2008. His research interests include algorithmic aspects of data analysis methods, as well as their application to real-world problems in science and industry.