An agent-based algorithm exploiting multiple local dissimilarities for clusters mining and knowledge discovery

Filippo Maria Bianchi1 · Enrico Maiorino1 · Lorenzo Livi2 · Antonello Rizzi1 · Alireza Sadeghian2

Abstract We propose a multi-agent algorithm able to automatically discover relevant regularities in a given dataset, determining at the same time the set of configurations of the adopted parametric dissimilarity measure that yield compact and separated clusters. Each agent operates independently by performing a Markovian random walk on a weighted graph representation of the input dataset. Such a weighted graph representation is induced by a specific parameter configuration of the dissimilarity measure adopted by an agent for the search. During its lifetime, each agent evaluates different parameter configurations and takes decisions autonomously for one cluster at a time. Results show that the algorithm is able to discover parameter configurations that yield a consistent and interpretable collection of clusters. Moreover, we demonstrate that our algorithm shows comparable performances with other similar state-of-the-art algorithms when facing specific clustering problems. Notably, we compare our method with respect to several graph-based clustering algorithms and a well-known subspace search method.

Keywords Agent-based algorithms · Data mining · Knowledge discovery · Clustering · Local dissimilarity measure · Graph conductance · Random walk

1 Introduction

Finding characterizing regularities in data is an important knowledge discovery task, which can be exploited for a multitude of purposes. When there is not any a priori knowledge on the dataset at hand, it could be useful to perform an initial exploratory analysis of the data to learn how to compare the elements in a meaningful way, so that relevant patterns in the dataset can be discovered. Clustering (Kannan et al. 2004; Nguyen and Wu 2013; Cao et al. 2013; Pedrycz 2005; Bianchi et al. 2015; Izakian et al. 2013; Zhang et al. 2014) is a well-established approach that can be used to this end. Among the many solutions available in this field, it is worth citing those clustering techniques based on graph-theoretical results and multi-agent systems (Bulò and Pelillo 2013; Tabrizi et al. 2013; Ferrer et al. 2009; Galluccio et al. 2012, 2013; Azran and Ghahramani 2006; North 2014; Agogino and Tumer 2006; Giannella et al. 2004). Graph-based techniques have the fundamental advantage of mapping the original problem onto a “dimensionless” object: the graph. Moreover, graph theory offers a plateau of theoretical results and multi-agent systems (Bulò and Pelillo 2013; Tabrizi et al. 2013; Ferrer et al. 2009; Galluccio et al. 2012, 2013; Azran and Ghahramani 2006; North 2014; Agogino and Tumer 2006; Giannella et al. 2004). Graph-based techniques have the fundamental advantage of mapping the original problem onto a “dimensionless” object: the graph. Moreover, graph theory offers a plateau of theoretical results to be exploited by effective algorithms, which easily integrate with the agent-based paradigm. Typical settings involving the interplay of both approaches include random walk (RW)-based algorithms (Alamgir and von Luxburg 2010; Gallesco et al. 2011), in which agents move and interact on the graph via specific (probabilistic) mechanisms.
An important family of clustering techniques is the subspace clustering, which consists in finding a subset of dimensions where the clusters are better defined (Parsons et al. 2004; Vidal 2010). This is performed by removing dimensions which are irrelevant or redundant in the clusters. The approach can be thought as an extension of feature selection, attempting to find clusters in different subspaces of the same dataset. In fact, while feature selection aims at compressing information by analyzing the entire dataset, subspace algorithms identify, for each cluster, the subspace that better characterizes its structure. Subspace algorithms are specifically designed to treat data represented with vectors and many implementations are not capable of treating values defined on discrete or circular domains.

When there is uncertainty about importance, interpretability or relationships of the features in the dataset at hand, a fundamental issue is the definition of the dissimilarity among the input data (Livi et al. 2014; Gisbrecht and Schleif 2015; Duin and Pękalska 2012), since the specific dissimilarity measure adopted by a data mining procedure affects the possibility of discovering meaningful regularities. Depending on the application at hand, data can be collected, transformed, and visualized using different representations. Accordingly, many (non-metric) parametric dissimilarity measures could be designed depending on the specific task (Livi et al. 2015). Recently, there is a steady increasing interest in using several, possibly heterogeneous, dissimilarity measures at the same time (Kim and Duin 2009; Pedrycz 2013; Queiroz et al. 2013; Bereta et al. 2013; Gönen and Alpaydın 2011). Regardless of the number of dissimilarity measures, the setting of their characterizing parameters is what really allows to discover the relevant information hidden in the data.

Metric learning (Shen et al. 2014; Yang et al. 2010; Yin et al. 2012; Zhang et al. 2012; Chang 2012) is an important topic in pattern recognition. Techniques in this field deal with the problem of learning an optimal setting of the parameters characterizing the particular dissimilarity for the problem at hand—usually it is assumed to be a metric distance. For a given dissimilarity measure, it is possible to distinguish two main approaches (Mu et al. 2013): those trying to determine a partition of data, and those that focus on searching for isolated clusters surrounded by uncategorised data. Local description of data is of particular interest, since it allows to characterize the input data by means of a heterogeneous collection of descriptions (Bereta et al. 2013).

In this paper, we propose the Local Dissimilarities-Agent-Based Clusters Discoverer (LD-ABCD) algorithm. LD-ABCD is designed to discover (learn) configurations of a parametric dissimilarity measure yielding one or more well-formed clusters. Cluster discovery is implemented by means of multiple RWS, which are independently performed by several agents on graphs representing the dataset. Each agent initially selects a specific Parameter Configuration (PC), which fully determines the dissimilarity measure used for building a weighted graph that encodes the proximity among the input data. The edges of the graph represent the dissimilarity among the elements in the dataset. The outcome of the dissimilarity measure depends on the specific PC used. For this reason, the edge labels are modified each time the adopted PC changes; consequently also the RW performed on the labeled graph is affected by the choice of the PC. During an RW, an agent searches and takes decisions autonomously for one cluster at a time. A suitable online mechanism is designed to decide whether a set of elements processed (i.e., “walked upon”) by an agent should be accepted as a meaningful cluster or rejected. To this end, we heavily exploit the graph conductance concept (Kannan et al. 2004).

The agent-based approach herein exploited allows the design of scalable algorithms. Scalability is an important requirement as the size of the dataset increases and when multiple computational resources are involved. In fact, the agent-based paradigm represents a straightforward solution for distributing the computation over different threads or even machines; this set the basis for handling large datasets in a distributed manner, an aspect that, however, is not directly explored in this work. Another advantage is that agents are able to perform their tasks in an autonomous, asynchronous manner and hence they can be created and removed at runtime if the availability of computational resources changes during the execution of the procedure. Moreover, multiple agents can perform different specialized tasks and autonomously modify their policy in a decentralized manner, as long as they maintain a common interface of communication with each other.

We demonstrate the validity of our approach by performing different types of experiments. First, we consider some synthetic datasets to highlight the features and the functioning of the algorithm, evaluating the capability of LD-ABCD in discovering Relevant PCs (RPCs), that is, those yielding well-formed clusters. Additionally, we provide demonstrative examples on the concept of equivalence among PCs, which will be later discussed in the paper. Successively, we processed some real datasets to compare the performances of LD-ABCD with respect to (w.r.t.) state-of-the-art algorithms, to assess the effectiveness of different features of our system. In particular, we compared the capability of LD-ABCD with other graph-based clustering algorithms in discovering clusters composed of elements belonging to the same (predefined) class. We also evaluated the accuracy of our algorithm w.r.t. a subspace clustering method in the characterization of different classes, in terms of discriminative features used in a classification task. Finally, we designed a test to appreciate a variant of the LD-ABCD algorithm, used for speeding up the generation of new candidate PCs.

The remainder of the paper is structured as follows. In Sect. 2 we introduce LD-ABCD, describing in detail all
reduces the amount of data transmitted between the various sites, i.e., to reduce the communication overhead. One of the main issues of distributed mining is how to mine across multiple heterogeneous data sources like multi-database and multi-relational data.

LD-ABCD identifies clusters on a dataset that is represented through a labeled graph: graph clustering is a well-known problem and it has been addressed in many other works (Delvenne et al. 2010; Bulò and Pelillo 2013; Tabrizi et al. 2013; Ferrer et al. 2009; Galluccio et al. 2012, 2013; North 2014). Such clusters are discovered by different agents, which operate according to a paradigm inspired by the multi-agent systems that can be found in the literature (Azran and Ghahramani 2006; Negenborn et al. 2010; Chandrasekhar and Naga 2011; Chaimontree et al. 2012; De Smet and Aeyels 2009; Alamgir and von Luxburg 2010; North 2014). Agent-based models offer a suitable framework for mining data in a distributed environment. The problem of distributed data mining consists in aggregating information collected by multiple agents in different sites and in discovering patterns in the global dataset (Cao 2009; Klusch et al. 2003). The goal here obviously would be to minimize the amount of data transmitted between the various sites, i.e., to reduce the communication overhead. One of the main issues of distributed mining is how to mine across multiple heterogeneous data sources like multi-database and multi-relational mining. Recently, the topic has received attention, especially for what concerns the distributed classification technique (Dittrich 1997; Gorodetsky et al. 2003; Provost and Hennessy 1996; Prodromidis et al. 2000; Han et al. 2011).

In the method that we present in this paper, each agent examines the elements of a dataset by performing an RW (Alamgir and von Luxburg 2010; Gallesco et al. 2011), trying to group them into a compact and populated cluster. Once a cluster is identified, it is evaluated using the well-known conductance measure (Kannan et al. 2004), which is computed using numerical approximation techniques (Arora et al. 2008; Hoory et al. 2006; Trefethen and Bau 1997; Leighton and Rao 1999; Arora et al. 2009; Madry 2010; Gkantsidis et al. 2003; Sarma et al. 2011). Each agent searches the clusters using different configurations of the adopted dissimilarity measure, seeking for the ones that better characterize the set of elements contained in the cluster. This approach can be related to the collaborative clustering technique, which was originally introduced in Pedrycz (2002) with the aim of discovering a common underlying structure in an ensemble of different data sites. The author considered a vertical partitioning scenario, and captured the collaboration between multiple clusterings via pairwise interaction coefficients. This resulted in an extended cost function to accommodate the collaboration effect in the optimization process. In Mitra et al. (2006) the authors presented an hybrid rough-fuzzy collaborative architecture where the collaboration was implemented by exchanging and even moving cluster prototypes from one site to another. In Forestier et al. (2010) the principle of collaboration is used for increasing the efficiency and the accuracy of the clustering: different methods work together and share their knowledge to reach an agreement on the partitioning of a common dataset.

Finally, it is worth stressing that, for problems where the data are represented by real-valued vectors and the task is to identify the subset of the most relevant features for each cluster, LD-ABCD behaves equivalently to a subspace clustering algorithm (Kriegel et al. 2009; Parsons et al. 2004; Vidal 2010), where the PC of the dissimilarity measure is assumed to be a set of binary weights. Our system, however, can, in principle, be generalized into any dissimilarity function, where there is not a direct relation between the parameters of the distance function and the features of the data.

2 The proposed LD-ABCD algorithm

The proposed multi-agent algorithm is designed to operate over a general input domain, $\mathcal{X}$, which may not necessarily be a subset of $\mathbb{R}^n$. Let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ be a symmetric dissimilarity measure (not necessarily metric) that depends on some parameters/weights, i.e., PCs, which we denote as $m \in \mathcal{M}$. The main goal of LD-ABCD is to identify a set of RPCs which are capable of inducing a well-formed cluster structure. In this sense, our algorithm should be intended also as a “knowledge discovery” procedure, since, in addition to the clusters discovered using local configurations of $d(\cdot, \cdot; m)$, it determines all relevant settings of the parameters characterizing the dissimilarity measure, which may be useful in terms of interpretability of the data and related clusters. Without loss of generality, here we also assume that $\mathcal{M} = [0, 1]^D$, where $D$ is the number of parameters/weights characterizing $d(\cdot, \cdot; m)$.

Figure 1 provides the overall high-level schema of LD-ABCD, together with details of the operations performed by
Fig. 1 Overall schema of the LD-ABCD algorithm. Initially, several agents are generated and each of them performs a series of tasks, which are shown in detail for the first agent. At first a new PC is generated and it is used for building a weighted graph of the dataset $S$, then an RW is performed on it. The set of visited nodes is treated as a cluster, which is evaluated by the agent. If the cluster is accepted, then it will be saved along with the other clusters found so far; otherwise if it is rejected, the agent restarts the whole procedure from the selection of a new PC. Similar clusters are aggregated into the same meta-cluster and the procedure goes on until a given number of iterations is performed. When all agents terminate their search, the meta-clusters and their associated PCs are returned a single agent within the proposed system. Each agent $a_i$ uses a different PC $m_j^{(i)}$ for evaluating the dissimilarity between the elements in the input dataset $S = \{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}$. The dataset is initially represented as a weighted complete undirected graph, $G_j = (\mathcal{V}, \mathcal{E}, w)$, where each edge $e_{kl} \in \mathcal{E}$ is characterized by a weight, $w(e_{kl}; m_j^{(i)}) \in [0, 1]$, which depends on the dissimilarity $d(x_k, x_l; m_j^{(i)})$ evaluated with the specific $m_j^{(i)}$. Each agent performs a Markovian RW (Lovász 1996) on the graph $G_j$, visiting a number of vertices (nodes) until a quantity called “energy” is not depleted.

The RW transition probabilities from one node to another are determined by the weights of the edges and hence depend on the parameter configuration associated with the agent. When an agent $a_i$, equipped with the PC $m_j^{(i)}$, runs out of energy, the vertices visited during the RW are interpreted as a cluster $c_h(m_j^{(i)}) \subset \mathcal{V}$ (or $c_{hj}$ to simplify the notation) found by the agent—which corresponds also to the subgraph $g_{hj}$. Therefore, each agent generates a single cluster at a time that is readily evaluated by the agent itself, which takes an autonomous decision on its acceptance. Since each agent generates the clusters independently from the others, the clusters returned by LD-ABCD may overlap (i.e., a given element can belong to more than one cluster) and, also, their union could not be equal to $S$; thus LD-ABCD does not generate a partition of the data (i.e., not all the elements in the data set will certainly belong to a cluster). During its lifetime, an agent performs several RWs on different versions of the same graph, which depends on the adopted PCs. Since for an agent it is possible to find similar clusters when using different PCs, these are progressively aggregated in prototypes called meta-clusters. The final output of the system is the list of discovered meta-clusters, which are returned along with the PCs used in the discovery of the set of clusters that they represent. The computation proceeds until a maximum number of iteration is reached, which is chosen by the user according to the available resources and the desired quality of the solution. In LD-ABCD, the number of agents is defined a priori by the user and it remains the same during the execution.

In the following, we provide the details about the tasks performed by a single agent during its lifetime. First, we discuss how the weighted graph is constructed over the input dataset (Sect. 2.1). Then we focus on the implementation of the RWs (Sect. 2.2) and the evaluation of the discovered clusters (Sect. 2.3). In Sect. 2.4, we describe how we manage the energy of an agent and the process of selection of the new PC.
2.1 Graph construction

Let us assume that an agent $a_i$ is equipped with the PC $m_j^{(i)}$, and let $\mathcal{S}$ be the dataset under analysis, with $n = |\mathcal{S}|$. The corresponding weighted graph $G_j = (\mathcal{V}, \mathcal{E}, w)$ is described by the vertices $\mathcal{V}$, each one representing an element in $\mathcal{S}$, and by the edges $\mathcal{E}$, which are weighted by computing $w(\cdot)$ as the exponential kernel:

$$w(e_{lk}; m_j^{(i)}) = \exp\left(-\tau_{\text{exp}} \cdot d(x_l, x_k; m_j^{(i)})\right).$$

(1)

The setting of the parameter $\tau_{\text{exp}} \geq 0$ is an important issue and it will be discussed later in Sect. 2.2. A weighted graph can be described by the $n \times n$-weighted adjacency matrix $A_j$, defined as:

$$A_j(l, k) = w(e_{lk}; m_j^{(i)}).$$

(2)

Since the vertex set is not affected by the specific PC, we keep the related data in a shared data structure, which is accessible by all agents. The edges, which instead can differ on the base of the specific PC, are stored locally by each agent, encoded in their weighted adjacency matrix.

2.2 Random walk for cluster search

To perform an RW on $G_j$ we need to define the so-called transition matrix (Lovász 1996), $M_j$, which is used by an agent to navigate among the vertices. $M_j$ is defined as follows,

$$M_j = D_j^{-1}A_j,$$

(3)

where $D_j$ is the (diagonal) degree matrix: $D_j(l, l) = \sum_{k=1}^{|\mathcal{V}|} A_j(l, k)$. An RW can be effectively characterized by exploiting the stationary distribution (SD) of the Markov process underlying the RW. The SD can be interpreted as the left eigenvector of $M_j$, associated to the largest eigenvalue, i.e., 1. Every complete and non-bipartite graph has a unique stationary distribution (Lovász 1996) corresponding to the degree distribution,

$$\pi_j(v_l) = \frac{D(l, l)}{2|\mathcal{E}|}, \quad \forall v_l \in \mathcal{V}.$$

(4)

We use the SD $\pi_j$ for selecting the starting vertex $v_l$ of an RW, since highly central vertices will have higher probability according to the SD. In this way, we let an agent start an RW from a dense region of the graph, rather than from a peripheral region in which it could be stuck or it could easily move to a vertex belonging to a different dense region (see Fig. 2).

A correct setting of $\tau_{\text{exp}}$ in Eq. (1) is crucial, since it affects the behavior of the RW performed by an agent. In fact, a higher value of $\tau_{\text{exp}}$ magnifies the edge weights between similar elements, making less likely the unwanted transitions to vertices connected by low weights (i.e., dissimilar elements). If we assign to $\tau_{\text{exp}}$ a value that is too high, the lower weights could be excessively magnified. In this case, an agent would repeatedly move on a very small set of vertices, instead of exploring a larger portion of the graph (i.e., transition probabilities become degenerate). On the other hand, assigning a too small value to $\tau_{\text{exp}}$ leads to the opposite situation, as it allows the agent to jump to different regions of the graph during the RW (i.e., transition probabilities become uniform).

As we process new datasets, different values of $\tau_{\text{exp}}$ must be determined for defining the transition matrix. In our experiments, we used an effective heuristic approach, that consists in generating a transition matrix for every value of $\tau_{\text{exp}}$ in the interval $[\tau_{\text{exp}}^{\text{min}}, \tau_{\text{exp}}^{\text{max}}]$, then performing a sufficiently high number of RWs using each matrix and evaluating the average quality and the dimension of the obtained clusters. Since the general trend is that the dimension of the cluster decreases as its quality increases (see Sect. 2.3 for the definition of cluster quality), the selected value must be a good trade-off between those two quantities. Of course, more accurate methods could be defined for estimating $\tau_{\text{exp}}$. However, since in our setting $d(\cdot, \cdot)$ may not be metric (and also not algebraic, i.e., which cannot be expressed in closed/analytic form), it is hard to find a strong relation among $\tau_{\text{exp}}$ and the transition probabilities.

2.3 Cluster quality evaluation

An agent $a_i$ generates a cluster $c_{ghj}$ during an RW performed on $G_j$ with the PC $m_j^{(i)}$, which consists in the set of vertices of the subgraph $g_{ghj}$ visited during the RW (see Fig. 2). In the following, we will refer equivalently to $g_{ghj}$ and $c_{ghj}$. Once a cluster $c_{ghj}$ is returned by an agent $a_i$, the cluster can be either accepted or rejected, depending on its quality. Intuitively, a cluster is considered to be good if it contains several elements, which are also very similar to each other according to the current PC, i.e., if it is a populated and compact cluster. A well-established measure used for evaluating the quality of a cluster associated to a subgraph of a larger graph is the conductance (Kannan et al. 2004), $\phi(c_{ghj})$, which quantifies how well knit is the subgraph internally and how many edges (with their associated weights) connected to vertices outside the cluster are cut. In terms of clustering, a subgraph with low conductance represents a compact and populated cluster, which is also well-separated from the remaining elements of the dataset. A straightforward method for evaluating the
quality of a cluster then consists in defining a function $CQ_1$ as follows:

$$CQ_1(c_{hj}) = 1 - \phi(c_{hj}).$$

A cluster $c_{hj}$ is, therefore, accepted if $CQ_1(c_{hj}) \geq \tau_{CQ}$, where $\tau_{CQ} \geq 0$ is a user-defined threshold.

However, directly using the conductance as a quality measure of clusters discovered in different datasets could be difficult to handle. In fact, the value of the conductance of a cluster depends also on the configuration of all the other elements in the dataset and thus it could fall within very diverse ranges, making the decisions and interpretations regarding its quality a difficult task. Additionally, since in our work we made no assumption on the employed dissimilarity measure used for comparing the data, it is not possible to express the variation of the conductance in closed-form as a function of the PC.

Fig. 2 RW example. Small dissimilarity values are represented with continuous lines, while the large ones with dashed lines. The probability of moving from one node to another is given by the transition matrix (3), which takes into account the magnitude of the dissimilarities. The current node is represented with dark gray color, while explored and unexplored nodes are represented, respectively, with light gray and white colors. The graph is fully connected but some edges are not shown for clarity. (a) The agent starts from node $B$, which is a central node in the graph selected using the SD $\pi$. b The agent uses the transition matrix $M$ for choosing the next node to visit. In this case, the node $A$ is selected with probability $P_{BA}$ as the next node in the walk. c The agent continues its walk moving from node $D$ to node $C$. d The resulting set of all the nodes visited by the agent during its walk.

For these reasons, we now introduce a new quantity for evaluating the quality of a cluster, which takes into account the properties of the whole graph constructed using a specific PC. In particular, we assert that the quality of a cluster $c_{hj}$ is proportional to the closeness of its conductance, $\phi(c_{hj})$, to the minimum conductance of the entire graph $G_j$ (or simply the conductance of $G_j$), denoted as $\Phi(G_j)$. The exact computation of $\Phi(G_j)$ is an NP-Hard problem (Kannan et al. 2004), and hence is not computationally feasible. As a consequence, in this paper we use an approximation for $\Phi(G_j)$, defined through a pair of real numbers, $lb(\Phi(G_j)), ub(\Phi(G_j))$, which represents, respectively, the lower and the upper bound of the interval that contains the actual value of $\Phi(G_j)$. These values can be computed exploiting the Cheeger’s inequality, by means of a procedure that is discussed in Appendix. We introduce a novel cluster quality function, $CQ_2$, defined as:
An agent-based algorithm exploiting multiple local dissimilarities

CQ2(c_h_j) = 1 - \frac{\phi(c_h_j) - lb(\Phi(G_j))}{ub(\Phi(G_j)) - lb(\Phi(G_j))}, \quad (6)

From our experiments, we observed that the use of CQ2 rather than CQ1 characterizes much better the quality of the clusters in our multi-parameter setting. To explain this fact with greater detail, let us consider an example where the two aforementioned functions are used for evaluating the cluster quality in two different datasets of \(\mathbb{R}^2\) vectors depicted in Fig. 3. We decided to consider two different datasets because the evaluation of the conductance is strictly correlated not only to the cluster itself, but also to the whole dataset to which it belongs. In both datasets, we select two different subsets of vertices of the respective graph representations: the first one is associated with a well-defined cluster, while the second one is randomly determined, which accordingly induces a low-quality cluster. In Fig. 4a, b we plot the values assumed by CQ1 and CQ2 on the well-defined clusters, which are evaluated as a function of the PCs (in this case uniformly sampled in the parameters space \([0, 1]^2\)). Instead, in Fig. 4c, d we performed the same calculations for the randomly determined clusters. As it is possible to observe, the values assumed by CQ2 fall within similar ranges in the two datasets, allowing to use comparable threshold values (i.e., \(\tau_{CQ}\)) for evaluating good clusters in different datasets. Using the CQ2 rather than CQ1, we are also able to better discriminate those PCs that better characterize the clusters—for the first dataset, these are individuated along the bisecting line, while for the second one PCs close to the \([0, 1]\) setting are preferable. In fact CQ2, in correspondence of such PCs, assumes values that better evaluate the quality of clusters: random clusters are always highly penalized while well-formed clusters are better magnified.

To conclude, since CQ2 is normalized according to the conductance of the graph, we consider Eq. (6) as an absolute quality measure that can be used for comparing clusters generated by different agents using different PCs. The soundness of such an assumption will be demonstrated by the experiments. In the following of this paper we will always use CQ2 as the function used for evaluating the quality of a cluster and, for the sake of notation, we will refer to it simply as “CQ”.

2.4 Energy update and selection of new PCs

The RW of an agent \(a_i\) is terminated when its energy \(e_i\) reaches a value lower or equal than zero. There exists then a dependency between the length of the RW and the energy quantity, in particular with its initial value \(e_{\text{init}}\) and the way it is updated. If the agent visits new vertices the energy value is decremented by 1, if the vertex has been already visited the energy is instead decremented by 0.01. This is done to ensure that paths between visited nodes come to an end if no new nodes are visited after a certain number of steps. For what concerns \(e_{\text{init}}\), if the transition probabilities are well defined, it is unlikely that an agent will walk out of an existing cluster on the graph. For this reason, even with many iterations the result will be prevalently stable and so we assigned a high value to the starting energy proportional to the dimension of the dataset, in particular we set \(e_{\text{init}} = \frac{n}{2}, n\) being the size of the dataset. The subset of vertices that have been visited during the RW forms the resulting cluster, \(c_{h_j}\), whose quality is evaluated according to Eq. (6). If CQ(\(c_{h_j}\)) is greater or equal than the threshold \(\tau_{CQ}\), the cluster is added to the collection of good clusters discovered by the agents, along
Fig. 4 With this figure we provide an example to justify our choice about the function used for evaluating the quality of a cluster. In particular, we plot the profile of the cluster quality computed using $CQ_1$ (in blue) and by considering $CQ_2$ (in red) when varying the PCs (2 parameters). In a and b we plot the profiles associated to well-defined clusters in the two datasets in Fig. 3, while in c and d we plot the profiles of the cluster quality evaluated on some random points in the same datasets. $CQ_2$ shows a better discriminative power as it generally assumes a larger range of values w.r.t. $CQ_1$, hence allowing for a better separation of relevant PCs. Furthermore, if we consider a and c or b and d, it is possible to observe a more marked difference of the values associated to different clusters. In particular, if we use $CQ_2$ the difference of the values associated to good and bad clusters results to be higher w.r.t. the values obtained with $CQ_1$ (color figure online).

with the PC $m_j^{(i)}$ used by the agent to discover such cluster. Since it is likely that a dataset contains more than one cluster of elements which are similar w.r.t. the same PC, it is reasonable to assume that if a PC $m_j^{(i)}$ has led to the identification of a good cluster, it can be further exploited to discover additional good clusters. Then, when a cluster is accepted, the initial quantity of energy of the agent is restored, i.e., it is set to $e_i = e_{\text{init}}$, and a new RW starts on the same weighted graph, enabling an agent to explore a new unseen region of the graph. To do this, we set to zero the weights of the matrix $A_j$ associated to the vertices which have been already visited by $a_i$ in the previous RW using $m_j^{(i)}$. To this end, both $\pi_j$ and $M_j$ are modified: changing $\pi_j$ has the effect that the next RW starts from another dense region of the graph, while the modification of $M_j$ prevents the agent from reaching vertices which have already been included in an accepted cluster.

Otherwise if $CQ(ch_j)$ is not high enough, $ch_j$ is rejected and the agent selects a new PC, say $m_{\text{new}}$; also in this case the energy $e_i$ is set to the default starting value, $e_{\text{init}}$. This implies the recalculation of $A_{\text{new}}, M_{\text{new}},$ and hence of $\pi_{\text{new}},$ inducing a completely new RW characterized by a possibly different behavior. The new PC $m_{\text{new}}$ is selected by considering a uniform distribution over $M$.

In Sect. 3 we describe a variant of LD-ABCD that implements a different PC selection strategy, which is more suitable for scenarios where the core dissimilarity measure is characterized by many parameters.

2.5 Aggregation of clusters/PCs

As long as the execution of LD-ABCD proceeds, an agent might find very similar (or even equal) clusters using different PCs, in the sense that they may overlap significantly. If the same cluster $ch_j$ is identified using two different PCs, $m_a$ and $m_b$, we say that such PCs are equivalent w.r.t. $ch_j$, in the sense that $ch_j$ contains elements that are characterized similarly by considering either $m_a$ or $m_b$. This is an important qualitative information that describes the cluster in terms of the parameters of the dissimilarity measure used for discriminating the elements of the cluster from the rest of the
To group similar clusters, we merge into a single meta-cluster all such clusters whose intersection, in terms of contained elements, is sufficiently high. It is, therefore, necessary to define a dissimilarity measure among clusters: to do that, we represented each cluster \( c_{hj} \) with a Boolean vector, \( c_{hj} \in \{0, 1\}^p \), where each entry of the vector represents an index to an element in \( S \), in particular the \( t \)th entry \( c_{hj}(t) = 1 \) if the \( t \)th element of \( S \) is contained in \( c_{hj} \), while \( c_{hj}(t) = 0 \) otherwise. At this point, the dissimilarity among clusters is computed through the Hamming distance \( d_H(\cdot, \cdot) \), which evaluates the distance among the two Boolean vectors that represent the clusters. Two clusters \( c_1 \) and \( c_2 \) are considered similar if their Hamming distance \( d_H(c_1, c_2) \) is less or equal to \( \theta \). The parameter \( \theta \geq 0 \) represents the maximum percentage of elements that can be different in two clusters to be considered similar.

With \( \hat{C}_x \), we call the \( x \)th meta-cluster that represents a collection of clusters \( C_x \) sufficiently similar to each other w.r.t. the Hamming distance. The meta-cluster \( \hat{C}_x \) is composed of a Boolean vector \( \mu_x \), where the value of the \( t \)th component is equal to the most frequent value among all the \( t \)th components of the clusters in \( C_x \), and a list \( L_x \) that contains all PCs used for discovering the clusters in \( C_x \). Each PC \( m_j \) in \( L_x \) is associated to a CQ value, which is derived from the conductance of the subgraph \( g_x \) of the graph \( G_x \), where \( g_x \) is the subgraph represented by \( \mu_x \) and \( G_x \) is the graph associated to the PC \( m_j \). Such CQ values are ordered in descending order and they represent a “ranking” that as confirmed in our experiments (see Sects. 4.1.1, 4.1.2) reflects the grade at which each PC in \( L_x \) characterizes \( \hat{C}_x \).

Every time a cluster \( c_{hj} \) is discovered by an agent \( a_i \), using a metric \( m_j^{(i)} \), it is compared with all the Boolean vectors \( \mu_1, \ldots, \mu_K(t) \) of the \( K(t) \) meta-clusters existing at time \( t \), and it is assigned to the most similar meta-cluster, say \( \hat{C}_x \). Then \( c_{hj} \) is added to the set \( C_x \) and \( \mu_x \) is recomputed on such set. Then, the PC \( m_j^{(i)} \) is added to \( L_x \) and its CQ is evaluated on \( g_x \).

If no meta-cluster has been generated, or if the dissimilarity value to the most similar meta-cluster is above the threshold \( \theta \), a new meta-cluster \( \hat{C}_x \) is instantiated starting from \( c_{hj} \). In this case, \( \mu_x \) is initialized with \( c_{hj} \) and the metric \( m_j^{(i)} \) used for discovering \( c_{hj} \) is inserted in the list \( L_x \), which initially will be empty.

2.6 Analysis of computational complexity

In this section, we study the time and space complexity of LD-ABCD. For what concerns the space occupancy, the upper bound consists in storing the weighted adjacency matrix, \( A \), which each agent must use to represent the graph. The space required to store the matrix is \( O(n^2) \), where \( n = |S| \).

On the other hand, the time complexity strictly depends on the number of iterations performed by each agent during the random walk. The length of a typical RW is related to the energy \( e \) of the agent and on how this quantity is modified (which is affected by the experimental setting of the algorithm and by the intrinsic random nature of the RWs). The energy variation depends also on the nature of the dataset at hand, which makes a precise analysis difficult to perform. To give an estimation of the computational time complexity, we assume here that an agent performs in average \( T \) different steps during a typical RW.

The time complexity can be estimated as the composition of several costs. The operations performed by an agent can be divided into the following categories, which scale with the input data size in different ways:

- the PC initialization step, which includes the generation of the PC given the selected policy and the evaluation of the adjacency and transition matrices and the computation of the graph conductance bounds. Sampling a random PC has a cost that scales linearly with the number of parameters of the dissimilarity measure, and so it can be generally considered negligible w.r.t. the costs depending on dataset size. Building the adjacency and transition matrices has a cost of \( O(n^2 \cdot \delta) \), where \( \delta \) is the cost of the dissimilarity measure. Evaluating the bounds of the graph conductance, used for evaluating CQ [see Eq. (6)], has the same cost of computing the second eigenvalue of the adjacency matrix, which in our study it has been approximated with the power method described in Appendix. The power method complexity scales as \( O \left( \left( n + n^2 \right) \cdot \frac{\epsilon}{\epsilon^2} \cdot \log \frac{\epsilon}{\delta} \right) \), where \( \epsilon \) represents the precision of the approximation. We refer then to the time required for initialization step with \( t_{\text{init}} = O(n^2 \cdot (\delta + \log(n))) \);
- the random walk step, which consists in selecting a new node and updating the energy \( e \) of an agent, according to the variation of the conductance of the subgraph visited so far. While the energy updating procedure can be performed in a constant time, selecting the next node in the RW is an operation which involves analyzing all the elements of the row of the adjacency matrix relative to the current node, which scales as \( O(n) \). We then define the cost \( t_{\text{step}} = O(n) \);
- the cluster quality evaluation step consists in the evaluation of the cluster conductance, which is an operation that costs \( O(n^2) \), since all the edges of the (complete) graph must be considered—see Appendix. The estimated time required for performing this step is given by \( t_{\text{eval}} = O(n^2) \).
the cluster aggregation step that consists in updating the set of existing meta-clusters with the cluster that has been accepted by the agent. This operation consists in comparing the cluster with all the \( K(t) \) meta-clusters which have been generated so far at the time \( t \), using the hamming distance. The hamming distance is linear in the number of the elements, which is the size of the dataset \( n \), since each cluster is represented in the vectorial form described in Sect. 2.5. Note that the aggregation procedure occurs only when a cluster is accepted, i.e., when its quality is sufficiently high, so this cost sometimes is equal to zero.

We can then define \( t_{aggr} = O(K(t) \cdot n) \).

To summarize, the total time \( t_{tot} \) required by an agent to evaluate a PC \( m_{ij} \) can be expressed as:

\[
t_{tot} = t_{init} + T \cdot t_{step} + t_{eval} + t_{aggr} = O(n^2 \cdot (\delta + \log(n))) + T \cdot O(n) + \cdots + O(n^2) + O(K(t) \cdot n)
\]

Since the procedure must be repeated each time a new PC is considered, the total time required for executing the whole LD-ABCD system is \( M \cdot t_{tot} \), where \( M \) is the number of PCs evaluated (we remind the reader that the number of agents is fixed in our algorithmic setting).

3 LD-ABCD with exploration–exploitation agents

Since the PC space can be extremely large in the case of dissimilarity measures with many parameters, the technique used for searching PCs described in Sect. 2.4—uniform sampling—may result ineffective. In this section, we propose an alternative approach for exploring the PC space. The search method is inspired to the well-known Metropolis–Hastings algorithm (Metropolis et al. 1953), often employed in statistical physics. In this variation, the agents operate according to two different policies, which we named, respectively, exploration and exploitation. An agent that operates according to the exploration strategy is called “explorer”. The exploration strategy coincides with the uniform search described in Sect. 2.4 and it is meant to perform an exploratory, wide-range search in the PC space. An explorer randomly evaluates several different PCs. Every time an RPC is identified by an explorer, it is stored to a shared data structure to allow successive tentative improvements via the exploitation. Accordingly, an agent that implements the exploitation strategy, instead, is called “exploiter”. The objective of the exploiters consists in trying to improve the RPCs found so far by the explorers. An exploiter randomly selects one of the available RPCs, say \( m_{ij}^{(i)} \), along with its corresponding cluster \( c_{hj} \), and initiates a search in the PC space nearby \( m_{ij}^{(i)} \), given a suitable PCs similarity measure \( d_{PC}(\cdot, \cdot) \). This search strategy is meant to discover other PCs that yield a higher CQ (6) on the same cluster \( c_{hj} \). In fact, since it is reasonable to assume that agents with similar PCs are likely to perform similar RWs (and hence accept/reject similar clusters), we keep the cluster structure fixed (i.e., the patterns that it contains) and we just recompute its CQ using the new PCs. The fact that we recompute the CQ of the cluster without performing a new RW results in a significant improvement in terms of computational resources. The implementation of the similarity measure between PCs depends on the nature of the parameters (e.g., Hamming distance for binary configurations, Euclidean distance for real-valued parameters, etc.). If an exploiter is able to select a new PC \( \bar{m}_{ij}^{(i)} \) that yields a better CQ than \( m_{ij}^{(i)} \), this latter is deleted and it is replaced by \( \bar{m}_{ij}^{(i)} \).

Every agent can exclusively assume the role of the explorer or the exploiter (Fig. 5), modifying hence its search strategy accordingly. Before starting a new RW, an agent checks the current ratio of explorers and exploiters operating

![Fig. 5 Diagram of the exploration–exploitation procedure for selecting the new PC. The thick arrows represent the read/write operations performed by the agents on the shared data structure, highlighted in gray, containing the discovered clusters and PCs](image-url)
in the system. If the ratio is above a user-defined threshold, $0 < \tau_{\text{EXP}} \leq 1$, and at least one RPC has been already discovered by an explorer, the agent adopts the exploitation policy, otherwise it behaves as an explorer. The factor $\tau_{\text{EXP}}$ controls the balance between the diversity and the accuracy of the returned RPCs and can be tuned according to the available computational resources and the particular problem at hand. The exploration–exploitation version of LD-ABCD herein discussed is designed to be able to perform a more targeted search on large PC spaces. This results, in general, in a faster convergence of the whole algorithm, with a faster discovery of the high-quality clusters and related PCs present in the data—we will provide experimental evidence of this claim later in Sect. 4.1.3. Finally, the herein presented exploration–exploitation variant is characterized by the same computational costs described in Sect. 2.6, as the operations for the explorers and exploiters are asymptotically the same.

4 Experiments

In this section, we present and discuss different typologies of experiments performed to assess the performances of LD-ABCD. First, in Sect. 4.1 we present three tests on synthetic datasets, whose purpose is to highlight the features of the algorithm and to demonstrate its effectiveness. In Sect. 4.2 we face two different kind of problems, processing some well-known benchmarking datasets to compare the capabilities of our system with other state-of-the-art algorithms. Notably, in Sect. 4.2.1 we compare LD-ABCD with other graph-based algorithms, while in Sect. 4.2.2 we evaluate the performances w.r.t. a well-known subspace search method on suitable problems of classification.

As stressed throughout the paper, our approach is dissimilarity based. Therefore, LD-ABCD is able to process virtually any input data type (e.g., graphs, sequences and so on). However, for the sake of simplicity and for an easier interpretation of the results, we decided to test only datasets of real-valued vectors (features); extensions to other settings are straightforward. The adopted dissimilarity measure is the weighted Euclidean distance; each $D_j$ is a vector in $[0, 1]^D$, where $D$ is the dimensionality of the data at hand. Our algorithm depends on a number of parameters and thresholds: $\tau_{\text{EXP}}$, $\tau_{\text{CQ}}$, $\theta$, and the maximum number of allowed iterations. In Sect. 2.2, we described a heuristic method to estimate $\tau_{\text{EXP}}$ that selects the value which generates clusters with a CQ value averagely high. The median CQ value can moreover be used to infer a suitable threshold $\tau_{\text{CQ}}$. In most of cases the chosen value is slightly higher than the measured average CQ. For what concerns $\theta$, we set it so that two clusters are merged together in the same meta-cluster if they both share at least 90% of their elements. Finally, increasing the maximum number of iterations allows the potential discovery of more clusters and PCs, to the detriment of computational time. In our experiments, with 1000 iterations we were able to discover a high number of meta-clusters having associated many equivalent PCs, while keeping the computational time under a reasonable threshold.

4.1 Tests on synthetic datasets

In this section, we report three different tests on synthetic datasets. The first, described in Sect. 4.1.1 underlines the capability of our system to discover relevant information in noisy datasets. Notably, the identification of relevant clusters, together with the PCs used for discovering such clusters, provides a semantic characterization and a high-level description of the data. The second test in Sect. 4.1.2 demonstrates the capability of LD-ABCD to discover multiple PCs which characterize individual clusters, defining then a relation among the features considered by each PCs in the data contained in the cluster. Finally, in Sect. 4.1.3 we discuss the results obtained using the exploration–exploitation technique described in Sect. 3 to speed up the selection of RPCs.

4.1.1 Discovering relevant PCs

In this test we identify a collection of meta-clusters (see Sect. 2.5) associated with the list $\mathcal{L}$ of the equivalent PCs that have been used in the process of cluster discovery. We order the PCs in $\mathcal{L}$ according to their CQ (see Sect. 2.3) evaluated on the resulting meta-cluster. Since by definition each cluster of the considered dataset is characterized by its own specific PC, we expect (i) to retrieve the correct PC and (ii) that the PC associated to the highest CQ is the one that better characterizes the cluster.

To demonstrate the capabilities of LD-ABCD, we have generated a dataset in $\mathbb{R}^4$, which contains 4 different clusters $c_1$, $c_2$, $c_3$, and $c_4$. The vectors forming each cluster are characterized by values drawn from a tensor product of a three-dimensional Gaussian distribution with spherical covariance matrix and unidimensional uniform distribution, which plays the role of the noise. For each of the 4 clusters, we select a specific dimension to add the values that come from a uniform distribution. Specifically, referring with $x[n]$ as the $n$th component of the vectors of the dataset, we insert the values drawn from the uniform distribution in $x[1]$ relatively to the patterns of $c_1$, in $x[2]$ for the patterns of $c_2$, in $x[3]$ for the patterns of $c_3$, and finally in $x[4]$ for the patterns of $c_4$.

In Fig. 6 we show the first three components of the considered patterns, omitting the 4th component, $x[4]$. As it is possible to observe from the figure, although the clusters are characterized by a narrow variance on a specific dimension,
they are clearly well separated. While the clusters $c_1$, $c_2$, and $c_3$ (plotted with blue dots) have the component containing the noise in one of the three displayed dimensions (respectively, on $x[1]$, $x[2]$, and $x[3]$), $c_4$ (plotted with red dots) has all the components with values drawn from the Gaussian distribution in $\mathbb{R}^3$ and the component containing the noise is $x[4]$. Note that the values of $x[4]$ for the blue clusters are drawn from a Gaussian distribution instead. We execute LD-ABCD using Boolean PCs only ($m_j$ are Boolean vectors), for a fixed number of iterations. As expected, LD-ABCD discovered four different meta-clusters $\hat{c}_i$, $\hat{c}_{ii}$, $\hat{c}_{iii}$, and $\hat{c}_{iv}$. In Table 1 we report the PC with higher CQ found for each meta-cluster and the relative CQ value.

As it is possible to observe, the PCs that have been found showing the highest CQ values are those that assign 1 in each cluster in correspondence of the components drawn from the Gaussian distribution (i.e., the signal), and 0 to the component drawn from uniform distribution (i.e., the noise). This demonstrates that LD-ABCD is able to discover the local structure of the relevant clusters in the dataset, identifying also the specific PC that allows such structures to emerge.

In our experiment, we reported for each cluster the first PC in the list $\mathcal{L}$ of equivalent PCs, that is the one with the highest CQ value, and thus the one that better characterizes the cluster. Such PCs are reported in Table 1. Notice that for this test the threshold $r_{CQ}$ can be set to an arbitrarily low value, because we are considering only the first PC (in terms of CQ) in $\mathcal{L}$ and ignoring the others.

### 4.1.2 Identification of equivalent PCs

In Sect. 2.5 we have introduced the concept of equivalent PCs which are associated to each meta-cluster. Such PCs are collected in the structure $\mathcal{L}$ associated to each meta-cluster in $\hat{\mathcal{C}}$. Each PC in $\mathcal{L}$ is characterized by a specific CQ value: the higher the CQ, the better the PC characterizes the meta-cluster. If a meta-cluster is associated with a set of PCs that are characterized by high and similar CQ values, we interpret them as equivalent, in the sense that they can be used interchangeably to suitably identify and locally characterize a cluster. Furthermore, we can identify relations among the parameters w.r.t. the dataset at hand.

To show this process and make it easily understandable, we have used a synthetic dataset in $[0, 1]^4$ that contains four different clusters—see Fig. 7 for an illustration. Each cluster contains data points which are very compact in two dimensions, while having uncorrelated values in the other two dimensions. More precisely, the projection of the cluster on the hyperplane formed by the first two dimensions is normally distributed with narrow variance around the center. This means that the first cluster is defined by the vectors whose first two components are extracted from two Gaussian distributions, $G_A$ and $G_B$; the second cluster is formed by vectors whose third and fourth components are drawn from the distributions $G_E$ and $G_F$, and so on. On the remaining dimensions, the vectors contain values which are drawn from a mixture of different Gaussian distributions (each one belonging to a different cluster) or noise. Since we wanted to keep the data in each cluster sufficiently isolated from the others, we drew the noise values by a random sampling considering a domain obtained by subtracting from $[0, 1]^4$ a suitable neighborhood of all the clusters.

In this sense, each cluster can be identified by PCs which assign high weights to any of the two signal components (or both), and a low weight to the others. For example, if we consider Boolean PCs, the cluster which contains vectors of the type $[A, B, \sim, \sim]$, where $\sim$ denotes either a signal different from $A$ and $B$ or a noisy component, can be identified by the following equivalent PCs: $\{1, 1, 0, 0\}$, $\{0, 1, 0, 0\}$, and $\{1, 0, 0, 0\}$.

In the herein presented experiment, we have generated a dataset of the form described above, which is exemplified in Fig. 7. Such a dataset contains 300 vectors in $[0, 1]^4$.

![Fig. 6](image-url) First three dimensions of the dataset characterized by four clusters in $[0, 1]^4$. Every cluster contains vectors with a component whose values are drawn from a uniform distribution, which plays the role of a noisy component. The blue clusters have that component in one of the displayed dimensions, while the 4th dimension of the red cluster is the one containing the noise (color figure online)

| Meta-cluster | PC | CQ      |
|--------------|----|---------|
| $\hat{c}_i$  | $\{x[1], x[2], x[3], x[4]\} = \{0, 1, 1, 1\}$ | 0.9247  |
| $\hat{c}_{ii}$ | $\{x[1], x[2], x[3], x[4]\} = \{1, 0, 1, 1\}$ | 0.9442  |
| $\hat{c}_{iii}$ | $\{x[1], x[2], x[3], x[4]\} = \{1, 1, 0, 1\}$ | 0.9181  |
| $\hat{c}_{iv}$ | $\{x[1], x[2], x[3], x[4]\} = \{1, 1, 1, 0\}$ | 0.9475  |
An agent-based algorithm exploiting multiple local dissimilarities...
Given a PC which is returned by an explorer, the exploiter randomly switches a parameter of the exploited PC equal to 1 from the original selected PC. This means that an exploiter generates similar PCs that have a hamming distance as a function of time (i.e., algorithm iterations).

By definition, the MCQ ranges from \( \tau \) to 1, which it intersects the most and the related cardinality of the intersection expressed in percentage as a function of time (i.e., algorithm iterations). Of course, after a short initial transient a higher MCQ value, at every given time step, indicates a faster identification of the RPCs. The generated dataset lies in a 30-dimensional space and it is characterized by ten well-separated clusters. The PC space consists of binary vectors of 30 parameters, so there are \( 2^{30} - 1 \) possible PCs (the all zeros configuration is never considered). We defined the Hamming distance as the dissimilarity measure \( d_{PC}(\cdot, \cdot) \) used for comparing different PCs (see Sect. 3). Given a PC which is returned by an explorer, the exploiters generate similar PCs that have a hamming distance equal to 1 from the original selected PC. This means that an exploiter randomly switches a parameter of the exploited PC to obtain the new candidate PC to be tested.

Figure 8 shows a plot of the MCQ obtained by both search methods, the uniform and the exploration–exploitation search, as a function of time. The exploration-exploitation setting has been run with the ratio \( r_{EXPL} = 3/4 \) over a total of 4 agents, i.e., 3 explorers and 1 exploiter. Such results are intended as the average of five different runs considered for each method, executed by changing the random seeds. As it is possible to observe, the MCQ obtained with the exploration–exploitation strategy rapidly assumes higher values w.r.t. those of the uniform search approach, and this behavior is preserved until convergence.

Please note that we are not reporting the results obtained by applying the exploration–exploitation strategy on the experiments described in the previous sections, since there are no significant variations worth discussing; in those problems the number of parameters of the dissimilarity measure was limited (less than 10 generally) and the improvement on the performance using the exploration–exploitation strategy can be better appreciated on high-dimensional parameter spaces.

### 4.2 Tests on Real Datasets

In this section, we applied our algorithm in two different kind of problems on real datasets, to compare our system with other existing algorithms which share some important features and fields of application with the LD-ABCD system. In Sect. 4.2.1, we compare the quality of the clusters identified by our system with the ones found by other graph-based clustering algorithms, on the basis of the uniformity of the class labels of the patterns contained within a single cluster. The second experiment (Sect. 4.2.2) consists in identifying the cluster that better represents the entire class. Successively, the PC associated to such a cluster is used for training a dissimilarity-based classifier which aims to correctly accepting/rejecting test patterns.

We have processed five different real-world classification datasets from the UCI Machine Learning Repository (Bache and Lichman 2013), which are Wine, Breast Cancer, Iris, SPECTF Heart, and E-Coli.

#### 4.2.1 Evaluating the purity of the RWs

As we described in Sect. 2.3, LD-ABCD uses the CQ [Eq. (6)]—a criterion based only on the evaluation of the conductance—for accepting or rejecting the clusters identified during the RWs. In the following experiments, we...
demonstrate the reliability of our (unsupervised) cluster acceptance criterion using the supervised information of the class labels. In this test, the PCs are defined as real-valued numbers.

We provide a comparison with the MARW algorithm (Alamgir and von Luxburg 2010) and two other algorithms therein considered, which are Nibble (Spielman and Teng 2013) and AprPageRank (Andersen et al. 2006) (in the following shortened as N and APR), relatively to the first two datasets (Wine and Breast Cancer). MARW is an agent-based and RW-based clustering algorithm. Agents perform the RW on the same graph together, with the constraint of having a (geodesic) distance of at most $l$ from each other. This corresponds to decreasing the chance that the multi-agent RW “mistakenly” merges two different clusters (low transition probabilities are easily zeroed). To make results comparable, we adopted the same performance measure described in Alamgir and von Luxburg (2010) for evaluating the purity of a cluster. The purity is the percentage of vertices visited during the RW that has the same class label of the starting vertex. Let $v_s$ be the starting vertex, $l(v)$ the true label value of $v$, and $c_h$ the accepted cluster made of vertices visited during an RW. The cluster purity (CP) is defined as:

$$r = \frac{|\{v \mid l(v) = l(v_s)\}|}{|c_h|}.$$  

In LD-ABCD, the starting node, $v_s$, is selected from the SD, $\pi$ (see Sect. 2.2). Hence, $v_s$ is selected from a central part of the graph, making its class label a reliable estimation of the class of the cluster to which $v_s$ effectively belongs.

For each processed dataset, we identify $K$ meta-clusters and their associated collection of equivalent PCs (see Sect. 2.5). From each meta-cluster, we chose the PC that has generated the cluster with the highest CQ and then we check its CP (7). We use the average value of those $K$ CPs as the performance index on the whole dataset (we report the standard deviations). The results obtained by our system are reported in Table 4, along with the results found by the other algorithms for what concerns the first two datasets.

In addition to this numerical comparison, in the following we briefly discuss the behavior of LD-ABCD in each dataset, to provide a more complete overview of its functioning. Since there is no pre-processing on the considered data, we decided to show a principal component analysis (PCA), which we use only for facilitating the comprehension of the following discussion (see Fig. 9).

**Wine** In this dataset, LD-ABCD was able to identify three different meta-clusters that correctly cover the three classes of the dataset. Each meta-cluster contains only patterns belonging to a single class and thus the CP associated to the PC with the highest CQ is 1 in every meta-cluster. According to MARW (Alamgir and von Luxburg 2010), we stopped the RWs as soon as a given number $z$ of different vertices are visited. The value of $z$ is selected proportional to the smallest class in the dataset at hand.

**Breast Cancer** This dataset contains two different classes of patterns which are characterized by a very different distribution, as we can see from the related PCA in Fig. 9. The elements of the first class are very similar and they occupy a compact portion of the space, while the others are spread on a less dense region. On this dataset our algorithm returned only one meta-cluster containing patterns belonging exclusively to the first class and thus the resulting average CP is 1. If at a first sight the absence in the output of a meta-cluster representing the second class may look as a failure, this behavior is perfectly aligned with the design of LD-ABCD, which tries to identify only the most compact and separated clusters in the dataset. From the point of view of clustering, in this dataset there is only one well-defined cluster (the one represented with yellow triangles). In fact, every time an agent tries to evaluate a cluster over the elements of the class represented with red dots, it systematically rejects those clusters because the related CQ is too low (they are highly conductive).

**Iris** For this dataset we have performed two different runs. In the first one, we have kept the threshold $\tau_{CQ}$ to the standard value (0.9) used in all other experiments, while in the second run we have lowered it to 0.5. In this way we allowed the algorithm to return more clusters, since the ones with a lower CQ are accepted. In the first test, only one meta-cluster is returned that contains points from the most isolated region

| Datasets | CP results on the considered UCI datasets |
|----------|------------------------------------------|
| Name (40) | Size | Dims | Classes | $N$ | APR | MARW $a = 3$ | MARW $a = 4$ | LD-ABCD |
| Wine (40) | 178 | 13 | 3 | 82.31 | 86.80 | 88.02 | 91.76 | 100.0 ± 0.000 |
| Breast cancer (160) | 683 | 9 | 2 | 93.26 | 94.66 | 94.37 | 96.02 | 100.0 ± 0.000 |
| Iris (40) | 150 | 4 | 3 | – | – | – | – | 76.00 ± 0.120 |
| E-Coli (50) | 336 | 8 | 8 | – | – | – | – | 91.00 ± 0.231 |

In parentheses we show the number of distinct vertices required to stop the RWs.
Fig. 9 First two components of the PCA of the considered UCI datasets. We use different colors and shapes for each class in the dataset (color figure online).

(see the PCA in Fig. 9). The CP obtained in this first run is equal to 1.

In the second run, instead, three different meta-clusters are returned. The first one contains again elements of the most isolated class and its CP is equal to 1, while the other two meta-clusters represent the two remaining classes and their CP is lower. In fact they are not well separated and the agent during a random walk switch between elements belonging to these two different classes, decreasing the CP of the resulting clusters. Also the CQ of those two clusters is significantly low since the agent moves freely on a larger portion of the graph, returning then a subgraph characterized by a higher conductance. The CP associated to the PC with the highest CQ of those two clusters is, respectively, 0.6 and 0.67, making the total CP obtained on the dataset equal to 0.76.

E-Coli Notwithstanding the dataset contains 8 different classes, the number of the resulting meta-clusters is 3 and they are mainly populated by patterns belonging to the largest classes of the dataset. In fact, the number of elements in the 5 remaining classes is remarkably lower, and they have been partially aggregated in the clusters representing the 3 principal classes. For this reason, the CP obtained on this dataset is not 1, even if it still maintains a good score; the best PC of the 3 clusters have the following CP: 0.96, 0.93 and 0.84, making the average CP of the whole dataset 0.91.

4.2.2 Classification using subspaces

When data are represented with sectioned vectors (Komorowski and Zytkow 1997) and the parameters of the dissimilarity measure represent the importance of each single feature, the task performed by LD-ABCD in tuning the PC corresponds to the selection of relevant features for each cluster and can be straightforwardly associated to the problem of subspace clustering. Among all the subspace clustering algorithms, we selected PROCLUS (Aggarwal et al. 1999), which was one of the first algorithms implementing the top-down approach (Parsons et al. 2004).

PROCLUS consists in three different steps called initialization, iteration, and cluster refinement, during which the clustering is iteratively improved. In the initialization, PROCLUS selects with a greedy strategy a set \( \mathcal{M} \) of candidates, called medoids, for the cluster representatives in the original
An agent-based algorithm exploiting multiple local dissimilarities... 

input space. The iteration phase selects a random set of $k$ medoids from $\mathcal{M}$ and for each medoid the algorithm considers a neighborhood and the set of dimensions along which the distances of the elements in the neighborhood are small. The total number of dimensions associated to medoids must be proportional to $l$, an input parameter selected by the user. Successively, all the elements of the dataset are assigned to their closest medoid, using a dissimilarity measure which considers for each medoid only the associated dimensions. At the end of each iteration bad medoids are replaced with new ones. Successively, the clustering quality is checked to assess improvements. The refinement phase computes new dimensions for each medoid based on the clusters formed (rather than on the neighborhood) and then reassigns points to medoids, removing outliers. One of the main drawbacks of the algorithm is its strong dependence on the parameters $k$ and $l$ which, in many cases, can be hard to be set in advance, since it requires an adequate knowledge of the problem and the dataset. Additionally, since the average number of dimensions is given, the subspaces for each cluster will be composed of similar dimensions. The output of PROCLUS is then a set of $k$ disjoint clusters with the associated relevant dimensions and an additional set of outliers:

$$\{\langle c_1, d_1 \rangle, \ldots, \langle c_k, d_k \rangle, \text{outliers} \}.$$ 

If the parameters of the dissimilarity measure are binary weights, $\{0, 1\}$, which are in charge of considering or not the relative feature in the dissimilarity computation, the dimensions returned by PROCLUS coincide with the PC of LD-ABCD.

To test our algorithm we have designed the following experiment. We run both LD-ABCD and PROCLUS on a dataset for classification with $K$ classes to obtain a collection of clusters. In particular, the result of PROCLUS is a partition of the dataset and a set of outliers, while the output returned by LD-ABCD is the set of meta-clusters. For each class $\gamma_i$ in the dataset, we initially find the meta-cluster $c_j$ that better represents such a class, according to the score function $s(c_j, \gamma_i)$ defined as follows:

$$s(c_j, \gamma_i) = \frac{|\{v(j) \mid l(v(j)) = \gamma_i\}|}{|\{v(j) \mid l(v(j)) \neq \gamma_i\}|} \cdot |c_j|, \quad (8)$$

where $v(j)$ is an element of cluster $c_j$ and its class is given by the value of the function $l(\cdot)$. Once the two clusters $c_{\text{best}}$ and $c_{\text{best}}^i$ have been identified as representatives for the class $\gamma_i$ among the clusters, respectively, returned by PROCLUS and LD-ABCD, we retrieve their associated dissimilarity configurations $m_{\text{best}}^p$ and $m_{\text{best}}^i$. For what concerns PROCLUS, $m_{\text{best}}^p$ is the set of the relevant dimensions of the cluster $c_{\text{best}}^p$, while in LD-ABCD $m_{\text{best}}^i$ is the PC with the highest CQ among the ones associated to $c_{\text{best}}^i$. Then, for each class $\gamma_i$ we train a specific classifier using the dissimilarity measure configured with the parameters retrieved by the two algorithms. Such classifier is designed to discern all the elements of class $\gamma_i$ from the remaining elements of the dataset. The classification is performed using a k-NN classifier (with $k = 7$) adopting the leave-one-out policy: $n - 1$ patterns are considered for training and the remaining one is classified simultaneously by the K classifiers. The procedure is repeated $n$ times (to effectively test all patterns in the original dataset) and $K$ confusion matrices are returned as output, each one relative to a specific classification problem. Due to the stochastic nature of the results of both PROCLUS and LD-ABCD, such a procedure is repeated 10 different times. In Table 5 we report the classification results, described by five different indicators, which are the True-Positive Rate (TPR), False-Positive Rate (FPR), Accuracy (ACC), F-score (F1), and Matthews Correlation Coefficient (MCC). Those indicators are extracted from the confusion matrix and we reported their average value and their variance computed on 10 different runs.

As we can observe from the results, the performances obtained using the dissimilarity configured using LD-ABCD are very similar and in many cases better than the ones shown by PROCLUS. This confirms the effectiveness of our method when applied in a subspace-clustering context. The only case in which LD-ABCD performs poorly is on the Breast Cancer dataset. Like in the previous test in Sect. 4.2.1, LD-ABCD identifies correctly only one cluster in this dataset. In fact, the two classes are characterized by a very different distribution and consequently patterns belonging to the class characterized by a low-density cluster are rarely visited by the agents performing RWs. However, this behavior is expected, given the design of LD-ABCD.

However, our algorithm, w.r.t. PROCLUS, offers an important advantage. In fact, as stated previously, PROCLUS depends on the two critical parameters, $k$ and $l$, which are in general unknown. For setting them, in our experiments we tried all the possible combinations of $l$ in $2, \ldots, D$, where $D$ is the maximum number of dimensions, and $k \in \{K - 1, K, K + 1\}$, being $K$ the correct number of classes. Then we executed the algorithm 10 times with each instance of $\langle k, l \rangle$ and we evaluated the average purity of the clusters obtained and the number of outliers. We then select the configuration $\langle k^*, l^* \rangle$ which obtained a good average purity and a low number of outliers. We can then claim that PROCLUS requires a priori knowledge for tuning such parameters, which in this case were the class labels, while LD-ABCD can rely exclusively on the CQ that operates in an unsupervised way.

To provide an additional evidence of the reliability of the CQ as an effective measure to identify the discriminating fea-
that is, the representation of the data in the dimensions of the principal component space. In this representation, which we call WinePCA, the dimensions of the data are ordered according to the data covariance, implying that the first two components of the pattern vectors are the most discriminat-

For each class, we configured the dissimilarity measure according to the PC associated to the cluster which better represents each class. We reported the results obtained using the most representative PC found by LD-ABCD and the one found by PROCLUS.

| Class | TPR | FPR | ACC | F1 | MCC |
|-------|-----|-----|-----|----|-----|
| Wine  |     |     |     |    |     |
| Proclus | 1  | 91.6 ± 1.7 | 0.3 ± 0.8 | 94.3 ± 1.1 | 95.5 ± 0.9 | 88.2 ± 2.2 |
| 2     | 99.6 ± 0.8 | 6.4 ± 2.7 | 97.2 ± 0.8 | 97.7 ± 0.6 | 94.2 ± 1.6 |
| 3     | 96.7 ± 0.8 | 0.0 ± 0.0 | 97.6 ± 0.6 | 98.3 ± 0.4 | 94.3 ± 1.4 |
| LD-ABCD | 1  | 98.7 ± 0.7 | 1.3 ± 0.7 | 98.9 ± 0.4 | 99.2 ± 0.3 | 97.7 ± 1.0 |
| 2     | 1.0 ± 0.0 | 9.2 ± 4.6 | 96.2 ± 1.8 | 97.0 ± 1.4 | 92.4 ± 3.6 |
| 3     | 97.8 ± 1.0 | 0.0 ± 0.0 | 98.4 ± 0.7 | 98.9 ± 0.5 | 96.2 ± 1.7 |
| Iris  |     |     |     |    |     |
| Proclus | 1  | 1.0 ± 0.0 | 0.0 ± 0.0 | 1.0 ± 0.0 | 1.0 ± 0.0 | 1.0 ± 0.0 |
| 2     | 95.6 ± 0.5 | 5.6 ± 0.9 | 95.2 ± 0.3 | 96.3 ± 0.2 | 89.3 ± 0.6 |
| 3     | 97.4 ± 0.5 | 9.2 ± 1.0 | 95.2 ± 0.3 | 96.4 ± 0.2 | 89.1 ± 0.6 |
| LD-ABCD | 1  | 1.0 ± 0.0 | 0.0 ± 0.0 | 1.0 ± 0.0 | 1.0 ± 0.0 | 1.0 ± 0.0 |
| 2     | 96.6 ± 0.8 | 4.0 ± 0.0 | 96.4 ± 0.5 | 97.2 ± 0.4 | 92.0 ± 1.2 |
| 3     | 96.4 ± 1.3 | 9.6 ± 0.8 | 94.4 ± 0.5 | 95.8 ± 0.4 | 87.3 ± 1.2 |
| Spectf |     |     |     |    |     |
| Proclus | 1  | 92.4 ± 1.5 | 90.4 ± 4.6 | 75.4 ± 0.8 | 85.6 ± 0.5 | 2.8 ± 5.7 |
| 2     | 9.8 ± 4.6 | 7.6 ± 1.5 | 75.4 ± 0.8 | 13.8 ± 6.2 | 2.8 ± 5.7 |
| LD-ABCD | 1  | 84.6 ± 2.3 | 60.7 ± 5.2 | 75.3 ± 2.4 | 84.5 ± 1.6 | 24.1 ± 6.4 |
| 2     | 37.8 ± 4.5 | 14.0 ± 1.4 | 76.1 ± 1.0 | 39.4 ± 3.3 | 24.6 ± 3.6 |
| Breast |     |     |     |    |     |
| Proclus | 1  | 94.1 ± 1.0 | 2.8 ± 0.2 | 96.1 ± 0.4 | 94.4 ± 0.6 | 91.4 ± 0.9 |
| 2     | 97.0 ± 0.1 | 0.0 ± 1.4 | 96.4 ± 0.5 | 97.3 ± 0.4 | 92.1 ± 1.2 |
| LD-ABCD | 1  | 95.0 ± 1.0 | 3.3 ± 0.6 | 95.9 ± 0.6 | 94.1 ± 0.8 | 91.0 ± 1.2 |
| 2     | 93.0 ± 0.7 | 7.3 ± 2.2 | 92.8 ± 4.8 | 94.3 ± 4.0 | 84.8 ± 9.3 |
| E-coli |     |     |     |    |     |
| Proclus | 1  | 94.9 ± 0.8 | 3.6 ± 0.7 | 95.5 ± 0.8 | 96.0 ± 0.7 | 90.9 ± 1.6 |
| 2     | 90.2 ± 2.4 | 28.8 ± 2.3 | 85.9 ± 2.4 | 90.8 ± 1.6 | 60.7 ± 6.0 |
| 3     | 99.0 ± 0.8 | 66.0 ± 46.5 | 95.1 ± 3.6 | 97.5 ± 1.8 | 34.8 ± 51.9 |
| 4     | 95.0 ± 1.5 | 19.6 ± 8.4 | 92.7 ± 2.6 | 95.6 ± 1.5 | 73.1 ± 9.6 |
| 5     | 1.0 ± 0.0 | 1.0 ± 0.0 | 99.8 ± 0.2 | 99.9 ± 0.1 | 98.7 ± 0.3 |
| 6     | 1.0 ± 0.0 | 1.0 ± 0.0 | 99.4 ± 0.1 | 99.7 ± 0.0 | 99.1 ± 0.2 |
| 7     | 95.1 ± 0.3 | 73.7 ± 20.4 | 87.9 ± 2.3 | 93.4 ± 1.2 | 23.9 ± 19.3 |
| 8     | 99.5 ± 0.6 | 19.3 ± 31.7 | 99.1 ± 0.8 | 99.5 ± 0.4 | 62.3 ± 19.3 |
| LD-ABCD | 1  | 95.8 ± 0.3 | 2.6 ± 1.3 | 95.5 ± 0.8 | 96.0 ± 0.7 | 90.9 ± 1.6 |
| 2     | 89.6 ± 1.5 | 27.5 ± 2.3 | 85.6 ± 1.7 | 90.5 ± 1.1 | 60.5 ± 4.3 |
| 3     | 99.2 ± 0.1 | 42.2 ± 2.7 | 96.8 ± 0.1 | 98.3 ± 0.0 | 67.8 ± 1.4 |
| 4     | 95.9 ± 1.2 | 26.1 ± 1.7 | 92.5 ± 0.7 | 95.5 ± 0.5 | 71.0 ± 1.9 |
| 5     | 1.0 ± 0.0 | 1.0 ± 0.0 | 99.4 ± 0.0 | 99.7 ± 0.0 | 99.3 ± 0.2 |
| 6     | 1.0 ± 0.0 | 1.0 ± 0.0 | 99.4 ± 0.0 | 99.7 ± 0.0 | 99.4 ± 0.1 |
| 7     | 95.6 ± 0.5 | 47.4 ± 4.3 | 91.1 ± 0.9 | 95.0 ± 0.5 | 90.4 ± 5.0 |
| 8     | 98.8 ± 0.3 | 12.0 ± 26.8 | 98.7 ± 0.5 | 99.3 ± 0.2 | 67.6 ± 18.7 |

In the dataset, we monitored how the weights associated to such features in the PCs are selected. To this end, we applied a principal component analysis (PCA) on Wine and we considered our dataset to be the principal component scores, that is, the representation of the data in the dimensions of...
The PCs found and we order them according to their CQ.

Successively, we studied how each value of the \( i \)th component PC\((i) \) changes with decreasing CQ. Since high CQ values correspond to PCs that better distinguish the relevant dimensions in the data, we expect that PCs with high CQ, select the first dimensions as relevant more often than the PCs that come after in the ordering. In other words, the first components’ PC\((i) \), which in Wine_{PCA} are associated to the most discriminative dimensions, should be set equal to 1 more often in the first PCs rather than in the latter.

For verifying this trend with a visual representation, in Fig. 10 we reported a moving average of the value of each component PC\((i) \) as a function of the decreasing CQ value. Notice that while each PC\((i) \in \{0, 1\} \), each value on the Y-axis in Fig. 10 is the average value assumed by PC\((i) \) in a set of “neighbouring” PCs with similar CQ rank value. As we can see the first two components PC\((1) \) and PC\((2) \) (depicted in red and blue in the image) are selected very often in the PCs with high CQ values and their selection frequency drastically decreases as the CQ decreases. We note also that the remaining components (depicted in the image with black lines) of the PC are always selected with the same frequency and they do not influence the results, since they are referring to features which are much less important for discriminating the patterns.

This proves the effectiveness of the CQ as a measurement for evaluating PC which select features which are discriminative in the problem at hand.

5 Conclusions

In this paper, we have proposed a dissimilarity-based multi-agent system, LD-ABCD, capable of discovering relevant clusters in a dataset, whose elements are grouped according to different but possibly equivalent configurations (instances of parameter values) of the dissimilarity measure. Agents in LD-ABCD perform multiple and independent random walks over the dataset represented as a weighted graph. Accordingly, each agent discovers and takes decisions independently over one cluster at a time. The multiple parameter configurations highlight the characteristics of patterns within a cluster that are considered to be discriminative, and represent the key for interpreting and characterizing semantically the regularities found in the dataset. The identified clusters are subgraphs whose quality is evaluated as a function of their conductance normalized by considering the bounds of the conductance of the entire graph. Guiding the evolution of our system with a cluster quality measure based on the conductance, allowed us to define a powerful tool for evaluating the effectiveness of a given configuration of the parameters and to identify well-formed clusters, as outlined also by the tests performed on the UCI datasets for classification. Our work highly relied on the celebrated Cheeger’s inequality as reference to define suitable bounds for the definition of the cluster quality. In this paper, we employed a very fast approach for computing an approximation of the minimum conductance of a graph, which is based on the numerical approximation of eigenvalues using the power method. This solution proved to be very useful and handy in our practical implementation.

We presented two different approaches for searching the parameters characterizing the dissimilarity measure: (i) the first one consisted in extracting configurations of the dissimilarity function parameters by means of a uniform distribution and (ii) an improved search strategy in which the initial solutions are successively refined by searching in their neighborhood (the exploitation search strategy). In this second strategy, agents are divided into two main families: the explorers and the exploiters. Performance comparison has been performed w.r.t. both graph-based and subspace clustering algorithms taken from the literature. The discussed experiments showed how LD-ABCD is capable of identifying the characterizing parameters of the dissimilarity measure, locally tailored for each single discovered cluster. Furthermore, when applied on the UCI datasets with a known class structure, the clusters returned by our algorithm contain elements belonging mostly to the same class.

Our future work will be focused on applying our system for clusters and knowledge discovery to larger datasets. Accordingly, we will focus on the aspects related to scalability and parallelization, showing how our algorithm can work by distributing the computations over different cores and/or distinct workstations, each of which would access a suitable
fraction of the entire dataset. In fact, since LD-ABCD does not produce a partition of the data, it could also operate only on a suitable subset of the entire dataset.

Acknowledgements The work presented in this paper has been partially funded by Telecom Italia S.p.a. The authors wish to thank Corrado Moiso, Software System Architect at Telecom Italia—Future Centre, for the benefits that he provided to the present project through the valuable comments, ideas and assistance to the writing and the undertaking of the research summarized here.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

6 Appendix: Graph conductance and related approximation

Given a graph $G = (V, E)$, with $n = |V|$, the conductance of a cut induced by the subset $S \subset V$ is defined as:

$$\phi(S) = \frac{\sum_{u \in S, v \in \bar{S}} A(u, v)}{\min(A(S), A(\bar{S}))},$$

(9)

where $\bar{S} = V \setminus S$ and $A(S) = \sum_{u, v \in S} A(u, v)$ are the number of edges in $S$. If the graph is weighted, then $A(u, v)$ contains the weight (i.e., the strength) of the edge among $u$ and $v$; if it is not weighted then $A(u, v)$ is equal to one if and only if there is an edge among $u$ and $v$. While computing the conductance (9) of any subset $S \subset V$ is simple, computing the conductance of the graph $\Phi(G)$ consists in solving the following NP-hard problem (Chung 1994):

$$\Phi(G) = \min_{S \subset V} \phi(S).$$

(10)

Finding the global optimum is infeasible even for small graphs. As a consequence, many approximation techniques have been proposed so far (Leighton and Rao 1999; Arora et al. 2009; Madry 2010; Gkantsidis et al. 2003; Sarma et al. 2011).

Among the many techniques, spectral techniques (Chung 1994) provide a very powerful approach. Let $A$ be the (weighted) adjacency matrix of $G$, and let $D$ be diagonal matrix containing the vertex degrees:

$$D = \text{diag}(d_1, \ldots, d_n), \text{ where } d_i = \sum_{j=1}^{n} A(i, j).$$

(11)

Let us define the transition matrix $M$ as:

$$M = D^{-1}A.$$  

(12)

The matrix $M$ is not always symmetric. Therefore, it does not always admit a spectral representation of the form $M = \Lambda U U^T$, where $\Lambda$ is a diagonal matrix containing the $n$ eigenvalues and $U$ is a matrix containing the corresponding eigenvectors. Notwithstanding, $M$ is conjugate to a symmetric matrix, $N$, which is defined as follows:

$$N = D^{-1/2}AD^{-1/2} = D^{1/2}MD^{-1/2}.$$  

(13)

$M$ and $N$ have the same eigenvalues and the eigenvectors are linearly correlated (Lovász 1996; Chung 1994). The eigenvalues of $N$ satisfy the following relation:

$$1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq -1.$$  

(14)

The celebrated Cheeger inequality (Lovász 1996) establishes an important relation among the conductance of $G$ (10) with $\lambda_2$:

$$\frac{\Phi(G)^2}{8} \leq 1 - \lambda_2 \leq \Phi(G),$$  

(15)

which can be rewritten as:

$$1 - \lambda_2 \leq \Phi(G) \leq \sqrt{8(1 - \lambda_2)}.$$  

(16)

Since $\phi(S) \geq \Phi(G)$ for any $S \subset V$, Eq. (16) can be used as a local reference for a specific graph. According to Eq. (16), it is possible to define the lower and the upper bound of the graph conductance as

$$\text{lb}(\Phi(G)) = 1 - \lambda_2,$$

$$\text{ub}(\Phi(G)) = \sqrt{8(1 - \lambda_2)},$$

(17)

which can be used for evaluating how much the conductance of a cut $\phi(S)$ is close to the conductance of the whole graph, $\Phi(G)$.

To make use of the bounds of Eq. (17), we need to compute the $\lambda_2$ eigenvalue. The QR decomposition (Trefethen and Bau 1997) is the most straightforward numerical technique for this purpose, which is, however, characterized by a cubic computational complexity. To overcome this drawback, we can use the power method described in Trefethen and Bau (1997), a fast algorithm that is able to compute in pseudo-linear time the largest eigenvalue and related eigenvector of a positive semi-definite (PSD) matrix. Notably, the computational complexity of the power method is $O((|V| + |E|)^{1/2} \log \frac{|V|}{\epsilon})$, where $\epsilon \geq 0$ is the approximation used in computing $\lambda_2$. Algorithm 1 describes the pseudo-code of the power method. The algorithm starts by randomly initializing a vector, $x_0 \in [-1, 1]^n$; it returns the vector $x_f = \tilde{M}x_0$, where $\tilde{M}$ is the PSD under analysis. The following theorem is an important result for the convergence of the power method (Arora et al. 2008; Hoory et al. 2006).
Theorem 1 For every PSD matrix $\tilde{M}$, positive integer $t$, a parameter $\epsilon > 0$ and a vector $x_0$ randomly picked with uniform probability $p$ in $[-1, 1]^n$, with $p > \frac{1}{10}$ over the choice of $x_0$, the power method outputs a vector $x_t$ such that

$$\frac{x_t^T M x_t}{x_t^T x_t} \geq \lambda_1 (1 - \epsilon) \frac{1}{1 + 4n (1 - \epsilon) 2^t}, \quad (18)$$

where $\lambda_1$ is the largest eigenvalue.

The eigenvector $v_1$ related to $\lambda_1$ would be approximated by $\frac{x_0}{\|x_0\|}$. Given a PSD matrix $M$ and the (unitary) eigenvector $v_1$ related to $\lambda_1$, we can compute $\lambda_2$ by means of Algorithm 2, which is a variation of Algorithm 1. The algorithm (2) returns a vector $x_t \perp v_1$, such that

$$\frac{x_t^T M x_t}{x_t^T x_t} \geq \lambda_2 (1 - \epsilon) \frac{1}{1 + 4n (1 - \epsilon) 2^t}. \quad (19)$$

The power method can only be applied to a PSD matrix, which is not the case of $N$, whose eigenvalues are the ones in Eq. (14). Consider now the matrix $\tilde{N} = N + I$. Every eigenvector of $N$ with eigenvalue $\lambda$ is clearly also an eigenvector of $\tilde{N}$ with eigenvalue $1 + \lambda$ and vice versa, thus $\tilde{N}$ has eigenvalues $2 = 1 + \lambda_1 > 1 + \lambda_2 \geq \cdots \geq 1 + \lambda_n \geq 0$ and thus it is PSD.

Using $v_1$ (an eigenvector of $\lambda_1$ computed with Algorithm 1), and setting $t = O(\epsilon^{-1} \log \frac{2}{\epsilon})$, Algorithm 2 will find with probability at least $3/16$ a vector $x_t \perp v_1$, such that

$$\frac{x_t^T M x_t}{x_t^T x_t} \geq \lambda_2 - 4\epsilon. \quad (20)$$

From Eq. (20), it is possible to derive the approximation of $\lambda_2$ that in turn can be used in Eq. (17).

Algorithm 1 Power method algorithm.

**Input:** PSD matrix $M$, tolerance $\epsilon$

**Output:** Approximation of eigenvector $v_1$ and related eigenvalue $\lambda_1$

1: Pick random vector $x_0 \in [-1, 1]^n$ with uniform probability;
2: $t = \epsilon^{-1} \log \frac{2}{\epsilon}$
3: for $i = 1$ to $t$
4: $x_i = M \cdot x_{i-1}$
5: $x_i = \frac{x_i}{\|x_i\|}$
6: end
7: $v_1 = x_t$
8: $\lambda_1 = \frac{x_0^T M x_0}{x_0^T x_0}$
9: return $v_1, \lambda_1$

Algorithm 2 Computation of the second eigenvalue.

**Input:** PSD matrix $M$, eigenvector $v_1$, and tolerance $\epsilon$

**Output:** Approximation of $\lambda_2$

1: Pick random vector $x_0 \in [-1, 1]^n$ with uniform probability;
2: $x_0 = x_0 - (v_1 \cdot x_0) v_1$
3: $t = \epsilon^{-1} \log \frac{2}{\epsilon}$
4: for $i = 1$ to $t$
5: $x_i = M \cdot x_{i-1}$
6: $x_i = \frac{x_i}{\|x_i\|}$
7: $x_i = x_i - (v_1 \cdot x_i) v_1$
8: end
9: return $\lambda_2 = \frac{x_0^T M x_0}{x_0^T x_0} - 1$.

References

Aggarwal CC, Wolf JL, Yu PS, Procopiuc C, Park JS (1999) Fast algorithms for projected clustering. SIGMOD Rec 28(2):61–72. doi:10.1145/304181.304188

Agogino A, Tumer K (2006) Efficient agent-based cluster ensembles. In: Proceedings of the fifth international joint conference on autonomous agents and multiagent systems, ACM, pp 1079–1086

Alamgir M, von Luxburg U (2010) Multi-agent random walks for local clustering on graphs. In: Proceedings of the IEEE 10th international conference on data mining, pp 18–27. doi:10.1109/ICDM.2010.87

Andersen R, Chung F, Lang K (2006) Local graph partitioning using pagerank vectors. In: Proceedings of the 47th annual IEEE symposium on foundations of computer science. IEEE Computer Society, Washington, DC, USA, pp 475–486. doi:10.1109/FOCS.2006.44

Arora S, Rao S, Vazirani U (2008) Geometry, flows, and graph-partitioning algorithms. Commun ACM 51(10):96–105

Arora S, Rao S, Vazirani U (2009) Expander flows, geometric embeddings and graph partitioning. J ACM 56(2):5

Azran A, Ghahramani Z (2006) Spectral methods for automatic multiscale data clustering. In: Proceedings of the 2006 IEEE computer society conference on computer vision and pattern recognition. IEEE Computer Society, Washington, DC, USA, pp 190–197. doi:10.1109/CVPR.2006.289

Bache K, Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 18 Feb 2015

Bereta M, Pedrycz W, Reformat M (2013) Local descriptors and similarities measures for frontal face recognition: a comparative analysis. J Vis Commun Image Represent 24(8):1213–1231. doi:10.1016/j.jvci.2013.08.004

Bianchi FM, Livi L, Rizzi A (2015) Two density-based k-means initialization algorithms for non-metric data clustering. Pattern Anal Appl. doi:10.1007/s10044-014-0440-4

Bulić SR, Pelillo M (2013) A game-theoretic approach to hypergraph clustering. IEEE Trans Pattern Anal Mach Intell 35(6):1312–1327

Cao L (2009) Data mining and multi-agent integration. Springer, Berlin

Cao J, Wu Z, Wu J, Liu W (2013) Towards information-theoretic K-means clustering on graphs. In: Proceedings of the IEEE 10th international conference on data mining, pp 18–27. doi:10.1109/ICDM.2010.87

Chaumont S, Atkinson K, Coenen F (2012) A framework for multi-agent based clustering. Auton Agents Multi-Agent Syst 25(3):425–446. doi:10.1007/s10458-011-9187-0

Chandrasekhar U, Naga P (2011) Recent trends in ant colony optimization and data clustering: a brief survey. In: 2nd international conference on intelligent agent and multi-agent systems (IAMA), pp 32–36. doi:10.1007/IAMA.2011.6048999
Chang CC (2012) A boosting approach for supervised mahalanobis distance metric learning. Pattern Recogn 45(2):844–862
Chung F (1994) Spectral graph theory. AMS, Providence
De Smet F, Aeyels D (2009) Cluster transitions in a multi-agent clustering model. In: Proceedings of the 48th IEEE conference on decision and control, 2009 held jointly with the 2009 28th Chinese control conference. CDC/CCC 2009, pp 4778–4784. doi:10.1109/CDC.2009.5400314
Delvenne JC, Yaliraki SN, Barahona M (2010) Stability of graph communities across time scales. Proc Natl Acad Sci 107(29):12,755–12,760. doi:10.1073/pnas.0903215107
Ditterich TG (1997) Machine learning research: four current directions. Artif Intell Magz 4:97–136
Duin RPW, Pekalska E (2012) The dissimilarity space: bridging structural and statistical pattern recognition. Pattern Recogn Lett 33(7):826–832. doi:10.1016/j.patrec.2011.04.019
Ferrer M, Valveny E, Serratosa F, Bardaji I, Bunke H (2009) Graph-based k-means clustering: a comparison of the set median versus the generalized median graph. In: Proceedings of the 13th international conference on computer analysis of images and patterns, CAIP ’09. Springer, Berlin, pp 342–350. doi:10.1007/978-3-642-03767-2_42
Foretiger G, Gančárskis P, Wemmert C (2010) Collaborative clustering with background knowledge. Data Knowl Eng 69(2):211–228
Gallesco C, Mueller S, Popov S (2011) A note on spider walks. ESAIM: Probab Stat 15:390–401
Gallicchio L, Michel O, Compon M, Hero AO III (2012) Graph-based k-means clustering. Signal Process 92(9):1970–1984. doi:10.1016/j.sigpro.2011.12.009
Gallicchio L, Michel O, Compon M, Kliger M, Hero AO III (2013) Clustering with a new distance measure based on a dual-rooted tree. Inf Sci 251:96–113. doi:10.1016/j.ins.2013.05.040
Giannella C, Bhargava R, Kargupta H (2004) Multi-agent systems and data mining. In: Cooperative information agents VIII. Springer, Berlin, pp 1–15
Gisbrecht A, Schleif FM (2015) Metric and non-metric proximity transformations at linear costs. Neurocomputing 167:643–657. doi:10.1016/j.neucom.2015.04.017
Gkantsidis C, Mihail M, Saberi A (2003) Conductance and congestion in power law graphs. ACM SIGMETRICS Perform Eval Rev ACM 31:148–159
Gönen M, Alpaydin E (2011) Multiple kernel learning algorithms. J Mach Learn Res 12:2211–2268
Gorodetsky V, Karsaeyv O, Samoilov V (2003) Multi-agent technology, IEEE, pp 438–441
Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques. Elsevier, New York
Hofroy S, Linial N, Wigderson A (2006) Expander graphs and their applications. Bull Am Math Soc 43(4):439–561
Iakian H, Pedrycz W, Jamal I (2013) Clustering spatio-temporal data: an augmented fuzzy C-means. IEEE Trans Fuzzy Syst 21(5):855–868. doi:10.1109/TFUZZ.2012.2233479
Kannan R, Vempala S, Vetta A (2004) On clusterings: good, bad and spectral. J ACM 51(3):497–515
Kim SW, Duin RPW (2009) A combine-correct-combine scheme for optimizing dissimilarity-based classifiers. In: Bayro-Corrochano E, Eklundh JO (eds) Progress in pattern recognition, image analysis, computer vision, and applications, LNCS, vol 5856. Springer, Berlin, pp 425–432. doi:10.1007/978-3-642-10268-4_49
Klusch M, Lodi S, Moro G (2003) Agent-based distributed data mining: the KDEC scheme. In: Klusch M, Bergamaschi S, Edwards P, Petta P (eds) Intelligent information agents, vol 2586. Springer, Berlin, pp 104–122. doi:10.1007/3-540-36561-3_5
Komorowski J, Zytkowski J (1997) Principles of data mining and knowledge discovery. Springer, Berlin
Kriegel HP, Kröger P, Zimek A (2009) Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans Knowl Discov Data 3(1):1–1:58. doi:10.1145/1497577.1497578
Leighton T, Rao S (1999) Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. J ACM 46:787–832. doi:10.1145/313524.313526
Livi L, Rizzi A, Sadeghian A (2014) Optimized dissimilarity space embedding for labeled graphs. Inf Sci 266:47–64. doi:10.1016/j.ins.2014.01.005
Livi L, Rizzi A, Sadeghian A (2015) Granular modeling and computing approaches for intelligent analysis of non-geometric data. Appl Soft Comput 27:567–574. doi:10.1016/j.asoc.2014.08.072
Lovász L (1996) Random walks on graphs: a survey. In: Mikhó D, Sós VT, Székely T (eds) Combinatorics, Paul Erdös is eighty, vol 2. János Bolyai Mathematical Society, Budapest, pp 353–398
Madry A (2010) Fast approximation algorithms for cut-based problems in undirected graphs. In: Proceedings of the 51st annual IEEE symposium on foundations of computer science, pp 245–254. doi:10.1109/FOCS.2010.30
Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21(6):1087–1092. doi:10.1063/1.1699114
Mitra S, Banka H, Pedrycz W (2006) Rough-fuzzy collaborative clustering. IEEE Trans Syst Man Cybern Part B: Cybern 36(4):795–805
Mu Y, Ding W, Tao D (2013) Local discriminative distance metrics ensemble learning. Pattern Recogn 46(8):2337–2349. doi:10.1016/j.patcog.2013.01.010
Negenborn RR, Hug-Glanzmann G, de Schutter B, Andersson G (2010) A novel coordination strategy for multi-agent control using overlapping subnetworks with application to power systems. In: Mohammadpour J, Grigoriadis KM (eds) Efficient modeling and control of large-scale systems. Springer, Norwell, pp 251–278
Nguyen TM, Wu QMJ (2013) Dynamic fuzzy clustering and its application in motion segmentation. IEEE Trans Fuzzy Syst 21(6):1019–1031. doi:10.1109/TFUZZ.2013.2240682
North MJ (2014) A theoretical formalism for analyzing agent-based models. Complex Adapt Syst Model 2(1):3
Parsons L, Haque E, Liu H (2004) Subspace clustering for high dimensional data: a review. ACM SIGKDD Explor Newsletter 6(1):90–105
Pedrycz W (2002) Collaborative fuzzy clustering. Pattern Recog Lett 23(14):1675–1686
Pedrycz W (2005) Knowledge-based clustering: from data to information granules. Wiley, New York
Pedrycz W (2013) Proximity-based clustering: a search for structural consistency in data with semantic blocks of features. IEEE Trans Fuzzy Syst 21(5):978–982. doi:10.1109/TFUZZ.2012.2236842
Predromidis A, Chan P, Stolfo S (2000) Meta-learning in distributed data mining systems: issues and approaches. Adv Distrib Parallel Knowl Discov 3:81–114
Provost FJ, Hennessy DN (1996) Scaling up: distributed machine learning with cooperation. In: Proceedings of the thirteenth national conference on artificial intelligence, vol 1, pp 74–79
Queiroz S, de Carvalho FDAT, Lechevalier Y (2013) Nonlinear multicriteria clustering based on multiple dissimilarity matrices. Pattern Recogn 46(12):3383–3394. doi:10.1016/j.patcog.2013.06.008
Sarma AD, Gollapudi S, Panigrahy R (2011) Estimating pagerank on graph streams. J ACM 58(3):13
Shen C, Kim J, Liu F, Wang L, van den Hengel A (2014) Efficient dual approach to distance metric learning. IEEE Trans Neural Netw Learn Syst 25(2):394–406. doi:10.1109/TNNLS.2013.2275170
Spielman DA, Teng SH (2013) A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. SIAM J Comput 42(1):1–26
An agent-based algorithm exploiting multiple local dissimilarities.

Tabrizi SA, Shakery A, Asadpour M, Abbasi M, Tavallaie MA (2013) Personalized pagerank clustering: a graph clustering algorithm based on random walks. Phys A: Stat Mech Appl 392(22):5772–5785. doi:10.1016/j.physa.2013.07.021

Trefethen LN, Bau D III (1997) Numerical linear algebra, vol 50. SIAM, Philadelphia

Vidal R (2010) A tutorial on subspace clustering. IEEE Signal Process Maga 28(2):52–68

Yang L, Jin R, Mummett L, Sukthankar R, Goode A, Zheng B, Hoi SCH, Satyanarayanan M (2010) A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval. IEEE Trans Pattern Anal Mach Intell 32(1):30–44. doi:10.1109/TPAMI.2008.273

Yin X, Shu T, Huang Q (2012) Semi-supervised fuzzy clustering with metric learning and entropy regularization. Knowl-Based Syst 35:304–311

Zhang H, Yu J, Wang M, Liu Y (2012) Semi-supervised distance metric learning based on local linear regression for data clustering. Neurocomputing 93:100–105

Zhang L, Pedrycz W, Lu W, Liu X, Zhang L (2014) An interval weighted fuzzy c-means clustering by genetically guided alternating optimization. Expert Syst Appl 41(13):5960–5971