Extending the SACOC algorithm through the Nyström method for Dense Manifold Data Analysis

Héctor D. Menéndez*

Department of Computer Science
University College London, United Kingdom
E-mail: h.menendez@ucl.ac.uk
*Corresponding author

Fernando E. B. Otero

School of Computing
University of Kent, United Kingdom
E-mail: F.E.B.Otero@kent.ac.uk

David Camacho

Department of Computer Science
Universidad Autónoma de Madrid, Spain
E-mail: david.camacho@uam.es

Abstract: Data analysis has become an important field over the last decades. The growing amount of data demands new analytical methodologies in order to extract relevant knowledge. Clustering is one of the most competitive techniques in this context. Using a dataset as a starting point, clustering techniques aim to blindly group the data by similarity. Among the different areas, manifold identification is currently gaining importance. Spectral-based methods, which are one of the main used methodologies in this area, are sensitive to metric parameters and noise. In order to solve these problems, new bio-inspired techniques have been combined with different heuristics to perform the cluster selection, in particular for dense datasets. Dense datasets are featured by areas of higher density, where there are significantly more data instances than in the rest of the search space. This paper presents an extension of a previous algorithm named Spectral-based Ant Colony Optimization Clustering (SACOC), a spectral-based clustering methodology used for manifold identification. This work focuses on improving the SACOC algorithm through the Nyström extension in order to deal with dense data problems. We evaluated the performance of the proposed approach, called SACON, comparing it against online clustering algorithms and the Nyström extension of the Spectral Clustering algorithm using several benchmark datasets.

Keywords: Ant Colony Optimization, Clustering, Data Mining, Machine Learning, Spectral, Nyström, SACON, SACOC

Reference to this paper should be made as follows: Menéndez, H.D. and Otero, F.E.B. and Camacho, David. (xxxx) 'Improving the SACOC algorithm through the Nyström extension for Dense Manifold Data Analysis', *International Journal of Bio-inspired Computation*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes:
Héctor Menéndez is Research Associate at University College London (UCL), UK. He received a PhD in Computer Science (2014) from Universidad Autónoma de Madrid. He also holds a BSc in Computer Science and a BSc in Mathematics from Universidad Autónoma de Madrid (2010). He is a member Software Systems Engineering Group at Department of Computer Science at UCL. His main research interests include malware, clustering, graph-based algorithms, genetic algorithms and ant colony optimization.

Fernando E. B. Otero is a Lecturer in the School of Computing at the University of Kent, UK. He received a BSc in Computer Science from Pontifícia Universidade Católica do Paraná, Brazil, in 2002; and a PhD in Computer Science from the University of Kent in 2010. His research interests include biologically inspired algorithms (mainly genetic programming and ant colony optimization), bioinformatics and data mining.

David Camacho is currently working as an Associate Professor in the Computer Science Department at Universidad Autónoma de Madrid, Spain. He is the Head of the Applied Intelligence & Data Analysis group. His research interests include data mining, clustering), evolutionary computation, multi-agent systems and computational intelligence, automated planning and machine learning.

1 Introduction

The importance of analysing large quantities of data is growing as a consequence of the rapid generation of data from social interactions, smart devices, WiFi and networks, among other sources [1]. Several methodologies have been extended in order to deal with the scale of the data. Recently, there has been an increased interest in methodologies based on MapReduce [1] and online analysis [2]. The former is an approach designed to parallelise the execution of an algorithm by using several nodes to distribute computation in two steps: *map* (takes an input pair and produces a set of intermediate values) and *reduce* (merges intermediate values to form a smaller set of values). Online analysis is a methodology where data instances are only processed once and the system keeps no information about old instances, reducing the memory use to allow an algorithm to work with datasets where there are strong density variation.

There are several data mining approaches that deal with dense data. The most relevant ones are those based on supervised and unsupervised learning [3]. Since supervised techniques usually need human assistant in a manual labelling process, which in many cases is costly, unsupervised techniques are gaining importance in this area [2]. One of the most relevant unsupervised techniques is clustering. Clustering is defined as the process of grouping data blindly, using a similarity criterion. Clustering algorithms have been extensively used in several heterogeneous fields [3].

Clustering can be divided into several sub-areas [3], such as centroid, medoid, hierarchical and continuity-based. This work is based on the latter. Continuity-based clustering is a methodology focused on the identification of manifolds within the data. Manifolds are structures defined by the data points inside the search space [4]—i.e., groups of data instances that define a specific form, such as a sphere or a cube. This work aims to extend the Spectral-based ACO Clustering (SACOC) algorithm by incorporating the Nyström extension [5]. SACOC is a bio-inspired clustering algorithm based on Ant Colony Optimisation (ACO) [6].

ACO algorithms are based on the foraging behaviour of the ants: many ants species are able to find the shortest path between their nest and a food source without any visual aid. All communication is performed indirectly by means of pheromone. Ants use pheromones to indicate the path that they have followed, and since ants following shorter paths are faster, more pheromone is deposited—the greater the pheromone concentration, the more attractive a path becomes for other ants. Eventually, all ants will follow the same path, which in most cases corresponds to the shortest path between the nest and a food source. ACO extends this idea to optimization problems and it has been successfully applied in several fields, such as data mining [7, 8].

The SACOC extension proposed in this paper, called SACON, applies a space approximation using a sub-sample of the dataset. This reduces the computational time of the algorithm since it does not need to calculate the similarity metric value for each data pair, allowing the algorithm to deal efficiently with larger datasets. It guarantees accurate solutions even when the reduction has been applied. It is evaluated using several continuity-based dense datasets and the results are compared against online clustering algorithms and the Nyström extension of spectral clustering.

The paper is structured as follows: Section 2 presents the related work, Section 3 introduces the SACOC algorithm, which is extended in Section 4 to create the new SACON algorithm. Section 5 shows the experimental setup used in the experiments presented in Section 6. Finally, Section 7 presents the conclusions.

2 Related Work

Clustering has been widely used in several heterogeneous areas, where the vast amount of data make clustering a promising technique given that it can deal with unlabelled data, while classification usually needs a previous (manual) labelling process. Current challenges are focused on dense and stream data analysis, where online clustering techniques are popular. Another area receiving increasing attention is continuity-based clustering for manifold identification, since it can be used to deal with dense data problems. In this section we review related work on both online and continuity-based clustering, and previous ACO approaches for clustering.

2.1 Online Clustering

The idea behind online clustering algorithm is to analyse this data using real-time techniques. These techniques usually need to deal with large quantities of data, making them suitable for dense data problems. One of the main challenges of real-time analysis is that they need a complete representation of the data space to update the information, limiting their applicability to design new clustering algorithms—e.g., there are clustering algorithms that might not be easily adapted to this kind of analysis [2].

One of the main tools used for online clustering analysis is the Massive On-line Analysis (MOA) tool. This framework provides the following online clustering algorithms:

- Online K-means [2]: This online algorithm updates the centroid position when a new instance arrives. Only one centroid is updated per iteration. It is similar to classical K-means algorithm.

- ClusTree [9]: This online algorithm iteratively updates the information of the clusters. It is able to consider the speed of the data stream generating the concept of the age of an object. It also maintains stream summaries.

- CluStream [10]: This algorithm combines offline clustering and online clustering in order to provide partial clustering solutions, which measures the evolution of the clusters.

2.2 Spectral Clustering

This section presents an overview of the Spectral Clustering (SC) algorithm, which is one of the most relevant continuity-based and manifold identification clustering algorithm. For each data pair, SC starts generating a similarity graph among the data instances. There are three main methodologies that are used [11]:

1. **ϵ-neighbourhood graph**: all the components whose pairwise distance is smaller than ϵ are connected.
2. **k-nearest neighbour graphs**: the vertex v_i is connected with vertex v_j if v_j is among the k-nearest neighbours of v_i.
3. **fully connected graph**: all points with positive similarity are connected with each other.

This work focuses in the fully connected graph. One of the most important metrics used in SC is the RBF kernel [12] defined by:

$$s(x_i, x_j) = e^{-\sigma||x_i - x_j||^2} \ , \tag{1}$$

where the metric calculates the inverse exponent of the Euclidean distance between points x_i and x_j using a normalization factor σ.

The second step is related to the study of the eigenvectors of the Laplacian matrix of the similarity graph. Depending on the Laplacian matrix calculation, there are three different techniques related to SC [11]:

1. **unnormalized Spectral Clustering**: it defines the Laplacian matrix as: $L = D - W$
2. **normalized Spectral Clustering**: it defines the Laplacian matrix as: $L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$
3. **random walks-based normalized Spectral Clustering**: it defines the Laplacian matrix as: $L_{rw} = D^{-1} L = I - D^{-1} W$

In these equations, I is the identity matrix, D represents the diagonal matrix whose (i, i)-element is the sum of the similarity matrix i-th row and W represents the similarity graph. Once the Laplacian is calculated, its eigenvectors are extracted. One of the main problems of SC is related to the consistency of the two classical methods used in the analysis: normalized and unnormalized. A deep analysis about the theoretical effectiveness of normalized clustering over unnormalized can be found in [13].

The last step is the application of a clustering algorithm to the projective space formed by the normalized eigenvectors, considering each row of the matrix as a point. The most frequently applied algorithm is K-means. There are several versions of SC according to the algorithm that is used in this step—e.g., the SACOC [14] algorithm is a Spectral-based algorithm which applies an ACO clustering algorithm instead of K-means.

The main challenge of SC lies in how to compute the eigenvectors and the eigenvalues of the Laplacian matrix of the similarity graph, avoiding a huge memory consumption. For example, when large datasets are analysed, the similarity graph of the SC algorithm requires a high memory storage that makes extremely hard the eigenvalues and eigenvectors computation.

2.3 Ant Colony Optimization in Clustering

ACO algorithms have been extensively applied to supervise learning—e.g., classification rules [15, 16], decision tree induction [7], neural networks [17] and naïve-bayes model [18]. ACO has also been applied to clustering with promising results. Kao and Cheng designed a centroid-based ACO clustering algorithm (ACOC) [19], where ants assign each data instance to one of the available clusters and clusters centroids are adjusted based on this assignment; following a similar direction, we presented the SACOC algorithm [14], which is a spectral extension of ACOC; a similar approach is also used to design a medoid-based ACO clustering (MACOC) algorithm [8], with the difference that ants assign data instances to medoids; Ashok and Messinger focused their work on graph-based clustering of spectral imagery [20], where the data is represented as a graph and an ACO procedure is used to find long paths through the data; several other approaches are discussed in [21].

It is interesting to remark that both MACOC and SACOC are based on ACOC algorithm in terms of the search, where ants visit all data instances to decide the cluster assignation. The main difference between them is the cluster/search space representation used: in ACOC, clusters are represented by centroids in an euclidean space; MACOC represents clusters as medoids (i.e., data instances are used to define the clusters); SACOC uses a spectral transformation to project the original space to a representation where similarities are more apparent.

3 SACOC: Spectral-based ACO Clustering Algorithm

This section presents SACOC [14], as it is the base for the proposed SACON algorithm. SACOC is similar to Spectral Clustering, where the goal of the algorithm is to discriminate the data using the information of a similarity graph and partition it into different clusters.

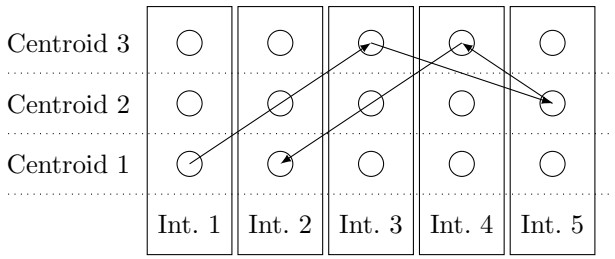


Figure 1 The construction graph of SACOC. The arrows illustrate a potential trail of an ant.

Algorithm 1 High-level pseudocode for the ACO-based clustering procedure.

Require: $X = x_1, \ldots, x_N$ data instances and k number of clusters
Ensure: $c_1, \ldots, c_k$ best k centroids
1: Initialize the pheromone matrix τ_0.
2: **for** iteration $t = 0$ **to** $maxIterations$ **do**
3: Initialize ants: $C^a = \{k$ randomly chosen centroids$\}$, $W_a = 0$ and $tb^a = \emptyset$
4: **for all** ant $a \in A$ (the ants set) **do**
5: **while** $|tb^a| < N$ **do**
6: Select the next instance i
7: Choose exploration or exploitation
8: Select $c_u \in C^a$ as the closest centroid
9: Set $w^a_{i,u} = 1$
10: add instance i to tb^a
11: **end while**
12: Calculate the objective function for each ant: $J^a = \sum_{i=1}^{N} \sum_{j=1}^{k} w^a_{ij} \cdot d(i, j^a)$
13: **end for**
14: Rank the ants according to J^a.
15: Choose the best ant a^* (iteration-best solution).
16: Compare it with the best-so-far solution (a^{**}) and update this value with the maximum between them.
17: Update pheromone values: $\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{h=1}^{r} w^h_{ij} \cdot \Delta\tau^h_{ij}$
18: **end for**
19: Re-centralize instances based on a^{**}.

3.1 Clustering data

SACOC performs the clustering of the data by using a strategy based on the ACOC algorithm [19], as illustrated in Algorithm 1. The search space is defined in terms of instances and centroids, which can be represented as a graph with an associated $N \times k$ matrix (where N is the number of instances and k is the number of centroids or clusters).

The algorithm uses several ants looking for the best path in the graph, illustrated in Figure 1. Each ant a has the following features: a list of visited objects (tb^a), a set of chosen centroids C^a and a weighted matrix W^a (related to the assignment of instances to clusters). To create a solution, an ant uses two different strategies: exploration and exploitation. It chooses the strategy according to:

$$j = \begin{cases} argmax_{u \in N_i}\{[\tau(i,u)][\eta^a(i,u)]^\beta\}\,, & \text{if } q \leq q_0 \\ S & \text{, otherwise} \end{cases} \quad (2)$$

where N_i is the set of nodes associated to instance i, $\tau(i,u)$ is the pheromone value between instance i and centroid u, q_0 is a user-defined exploitation probability, q is a random number for strategy selection, β is an ACO parameter that controls the influence of the heuristic and j is the chosen cluster; $\eta^a(i,u)$ is the heuristic value between i and u for ant a, defined as $1/d(i,u^a)$, where i is a data instance and u^a is the u-th centroid from the ant a centroid list. When q is greater than q_0, an ant uses the probabilistic exploration strategy S, defined by:

$$S = P^a(i,u) = \frac{[\tau(i,u)][\eta^a(i,u)]^\beta}{\sum_{j=1}^{k}[\tau(i,j)][\eta^a(i,j)]^\beta}\ , \quad (3)$$

where $P^a(i,u)$ is the probability of assigning instance i to cluster u and k is the total number of clusters.

The algorithm steps presented in Algorithm 1 can be divided in:

1. Initialize pheromone matrix (line 1).
2. Initialize ants (line 3): (tb^a, C^a, W^a), for each ant a in the colony; then, each ant repeats until tb^a is full:
   (a) Select randomly an instance i satisfying $i \notin tb^a$ (line 6).
   (b) Select a cluster j: first the ant chooses a strategy; then, it calculates the transition probability and visits a node (line 7).
   (c) Update tb^a and W^a (lines 9 and 10).
3. Calculate the objective function for each ant (line 12):
$$J^a = \sum_{i=1}^{N} \sum_{j=1}^{k} w^a_{ij} \cdot d(i, j^a)\ , \quad (4)$$
   where w^a_{ij} is a weight value of the assignment matrix W^a.
4. Select the best solution (lines 14 to 16). First, rank ants solutions, select the iteration-best solution, apply local search (for more details of local search see [22]) to improve the solution and, finally, compare it with the best-so-far solution and update this value with their maximum.
5. Update pheromone trails (line 17): only the best r ants are able to add pheromones. Let ρ be the pheromone evaporation rate ($0 < \rho < 1$), t the iteration number, r is the number of elitism ants and $\Delta\tau^h_{ij} = 1/J^h$ (line 17):
$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{h=1}^{r} w^h_{ij} \cdot \Delta\tau^h_{ij}\ . \quad (5)$$

Algorithm 2 High-level pseudocode of SACOC algorithm [14].

Require: $X = x_1, \ldots, x_N$ data instances and k number of clusters

Ensure: $c_1, \ldots, c_k$ best k clusters

1: Form the similarity graph $W \in R^{N \times N}$ defined by $W_{ij} = e^{-||x_i - x_j||^2/2\sigma^2}$ if $i \neq j$, and $W_{ii} = 0$.
2: Define D to be the diagonal matrix whose (i,i)-element is the sum of the i-th row of W.
3: Construct the matrix $L = D^{-1/2}WD^{-1/2}$.
4: Find $v_1, \ldots, v_k$, the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues) and form the matrix $V = [v_1 v_2 \ldots v_k] \in R^{N \times k}$ by stacking the eigenvectors in columns.
5: Form the matrix Y from V by renormalizing each row of V to have unit length (i.e. $Y_{ij} = V_{ij}/(\sum_j V_{ij}^2)^{1/2}$).
6: ClusterData(Y,k).

6. Check the termination condition: if the number of iterations is greater than the maximum limit, finish; otherwise, go to step 2.

3.2 The spectral hybridisation

The clustering procedure presented in Algorithm 1 was originally designed to use Euclidean space as a search space. However, it can be modified to consider any kernel in a similar way that K-means is modified to generate the SC algorithm—a similar strategy is employed in SACOC. Algorithm 2 presents the high-level pseudocode of SACOC. Consider a graph G and its associated weighted matrix W, which is a pairwise similarity graph among the data. The similarity is calculated using a similarity function defined by a kernel $k(x_i, x_j)$, where x_i and x_j are the i-th and j-th data instances, respectively (line 1). The spectrum of the graph is calculated in a similar fashion as proposed by Ng et al. [23] (lines 2 and 3) to create the SC algorithm. First, we calculate the Laplacian matrix defined by: $L_{sym} = I - D^{-1/2}WD^{-1/2}$, where I is the identity matrix and D represents the diagonal matrix whose (i,i)-element is the sum of the similarity matrix i-th row. After the creation of the Laplacian matrix, we extract the $v_1, \ldots, v_z$, (line 4), which corresponds with the z largest eigenvectors of L—chosen to be orthogonal to each other in the case of repeated eigenvalues—and form the matrix $V = [v_1\ v_2\ \ldots\ v_z] \in \mathbb{R}^{n \times z}$ by stacking the eigenvectors in columns. Finally, we form the matrix Y from V by renormalizing each row of V to have unit length (i.e., $Y_{ij} = V_{ij}/(\sum_j V_{ij}^2)^{1/2}$) (line 5). Then, we can consider Y as a projection of the original space and apply the clustering procedure (line 6) to the representation of each point.

4 SACON: Improving the SACOC algorithm through the Nyström extension

The goal of combining SACOC with the Nyström extension is to reduce the dimensions sampling of the similarity matrix. The reduction is achieved by choosing a subset of points $S = \{s_1, \ldots, s_n\} \in X = \{x_1, \ldots, x_N\}$ (where $n < N$)—the high-level pseudocode of SACON is presented in Algorithm 3. Given a similarity matrix W (related to the similarity graph), we need to extract the eigenvectors of its spectrum in order to project the data. In this work, the spectrum is defined by L_{sym}. In order to improve the computational time through the Nyström method, we need to use the sub-sample S and reformulate the whole process to describe how to extract approximate eigenvectors of the spectrum using less information related to the original similarity matrix.

The approximation is obtained by applying the Nyström extension [5], defined as follows:

Nyström Extension: Let $k(x_i, x_j)$ be a kernel whose Gram matrix K is symmetric and positive semi-definite satisfying $K_{i,j} = k(x_i, x_j)$. We assume that the eigendecomposition is $KU = U\Lambda$ where U is the orthogonal matrix of eigenvectors and Λ is the diagonal matrix composed by the eigenvalues. Then, the Nyström extension for a new instance x is the eigenvector approximation $\bar{u}^k(x)$ to the real $u^k(x)$ given by:

$$\bar{u}^k(x) = \frac{1}{\lambda_k^u} \sum_{j=1}^{N} k(x, x_j) \hat{u}_j^k \ ,$$

where $\bar{u}^k$ is the extension of the eigenvector $\hat{u}^k$ calculated from the sub-sample set of the matrix K, $\hat{u}_j^k$ is the coordinate j of the eigenvector and λ_k^u is the eigenvalue associated to the eigenvector $\hat{u}^k$.

In order to apply the extension to calculate the eigenvectors of L_{sym}, we simplify the process as follows. First, the eigenvectors of $L_{sym} = I - D^{-1/2}WD^{-1/2}$, are equivalent to the eigenvectors of $P = D^{-1/2}WD^{-1/2}$ since the identity matrix can be omitted.

The eigenvectors approximation follows the scheme specified in Algorithm 3. In this case, we use W and take the sub-samples set S to define a sub-matrix of W called A. The matrices satisfy:

$$W = \begin{pmatrix} A & B \\ B^t & C \end{pmatrix} \ ,$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times (N-n)}$ and $C \in R^{(N-n) \times (N-n)}$ (line 3). The Nyström extension will only need the augmented matrix $(A|B)$ formed by A and B. The matrix C is the part that we want to approximate and satisfies that has more elements than A, due to $n < N$. Using matrix C, it will be able to approximate eigenvectors of U calculating the eigenvectors of A, denoted by $\bar{U}$ (line 4). The eigenvectors $\bar{U}$ are extended to form the approximation

$\mathcal{U}$, which is close to the original U. Once the eigenvectors $\bar{U}$ have been calculated, we have the eigendecomposition of A, denoted as $A = \bar{U}\bar{\Lambda}\bar{U}^T$ (where $\bar{U}$ is orthonormal). The approximation $\mathcal{U}$ of the eigenvectors of W is calculated as (line 5):

$$\mathcal{U} = \begin{pmatrix} \bar{U} \\ B^t\bar{U}\bar{\Lambda}^{-1} \end{pmatrix} \ .$$

Now, we need to calculate the Laplacian eigenvectors, which are approximated by the matrix (line 7): $L' = D^{-1/2}W'D^{-1/2}$, where $W' = \mathcal{U}\widehat{\Lambda}\mathcal{U}$, $\widehat{\Lambda} = \frac{N}{n}\bar{\Lambda}$. Setting $D_{i,i} = \sum_{j=1}^{N} w'_{ij}$, we can restructure D as:

$$D = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \ ,$$

where $D_1 \in \mathbb{R}^{n\times n}$, $D_2 \in \mathbb{R}^{(N-n)\times(N-n)}$ and L can be approximated by (lines 7 and 8):

$$L' = \begin{pmatrix} A' & B' \\ (B')^t & (B')^t(A')^{-1}B' \end{pmatrix} \ ,$$

where $A' = D_1A$ and $B' = D_2B$. In order to approximate the eigenvectors of L' we will use the same methodology that we have used to approximate W. Assuming that the eigenvectors of A' and eigenvalues are $\mathcal{V}^o$ and Λ^o, and using A' as the base, the approximation is given by:

$$\Lambda' = \frac{N}{n}\Lambda^o \ , \quad \mathcal{V} = \sqrt{\frac{n}{N}}\begin{bmatrix} A' \\ (B')^t \end{bmatrix}\mathcal{V}^o\Lambda^o \ ,$$

where $\mathcal{V}$ and Λ' are the extended eigenvectors and eigenvalues of L'. Since $\mathcal{V}$ is not orthogonal, which is required by the spectral clustering algorithm, we apply the Fowlkes et al. [5] transformation and define the matrix R as (line 9):

$$R = A' + (A')^{-1/2}B'(B')^t(A')^{-1/2} \ .$$

This matrix can be decomposed as $R = U_R\Lambda_RU_R^t$, since A' is positive define. Then, $\mathcal{V}$ is defined as:

$$\mathcal{V} = \begin{bmatrix} A' \\ (B')^t \end{bmatrix}(A')^{-1/2}U_R\Lambda_R^{-1/2} \ ,$$

and finally, P is defined as:

$$P = D^{-1/2}W'D^{-1/2} = \mathcal{V}\Lambda_R\mathcal{V}^t \ .$$

At the end of this process, $\mathcal{V}$ are the eigenvectors used cluster the data (i.e., the projection of the data based on the Spectrum). This whole process helps to reduce the computational time consumed by the pairwise calculation, which is usually performed to create the whole similarity graph, and also reduces the memory consumption.

Algorithm 3 High-level pseudocode of SACON algorithm.

Require: $X = x_1, \ldots, x_N$ data instances and k number of clusters

Ensure: $c_1, \ldots, c_k$ best k clusters

1: Select a subsample of the data instances: $S = \{s_1, \ldots, s_n\} \in X = \{x_1, \ldots, x_N\}$ where, $n < N$.
2: Form the similarity graph $A \in R^{n\times n}$ defined by $A_{ij} = e^{-\sigma||s_i - s_j||^2}$ if $i \neq j$, and $A_{ii} = 0$.
3: Calculate the matrix B formed by the similarities among the elements of A and the rest of data instances.
4: Calculate the eigenvectors of A, named $\bar{U}$ and the eigenvalues, named $\bar{\Lambda}$.
5: Calculate the approximate eigenvectors $\mathcal{U}$ of W which try to approximate the real eigenvectors, named U, as:
$$\mathcal{U} = \begin{pmatrix} \bar{U} \\ B^t\bar{U}\bar{\Lambda}^{-1} \end{pmatrix}$$
6: Define D to be the diagonal matrix whose (i,i)-element is the sum of the i-th row of W.
7: Construct the matrix $L' = D^{-1/2}W'D^{-1/2}$.
8: Separate $L' = \begin{pmatrix} A' & B' \\ (B')^t & (B')^t(A')^{-1}B' \end{pmatrix}$
9: Set $R = A' + (A')^{-1/2}B'(B')^t(A')^{-1/2}$, and calculate its eigenvector decomposition $R = U_R\Lambda_RU_R^t$.
10: Set $\mathcal{V} = \begin{bmatrix} A' \\ (B')^t \end{bmatrix}(A')^{-1/2}U_R\Lambda_R^{-1/2}$.
11: Find $v_1, \ldots, v_k$, the k largest eigenvectors of L' and form $V = [v_1v_2 \ldots v_k] \in R^{n\times k}$ by stacking the eigenvectors in columns.
12: Form the matrix Y from V by renormalizing each row of V to have unit length (i.e. $Y_{ij} = V_{ij}/(\sum_j V_{ij}^2)^{1/2}$).
13: ClusterData(Y,k).

5 Experimental Setup

The evaluation of a clustering algorithm is a sensitive process, due to clustering being a blind process. There are no universal methods to evaluate it, in particular when the algorithm deals with large datasets. In this work we have focused the evaluation on comparing the proposed SACON algorithm against state-of-the-art algorithms. Experiments have been carried out using the Nyström extension of Spectral Clustering (SC+Nyström) [5], CluStream, Online K-means and ClusTree.

The parameters of SACON are: the sub-sample size fixed to 500 and the sigma value was calculated using the methodology described by Ng et al. [23], the same value used for SC+Nyström); the number of ants is 10, elitism is 1, exploitation probability is 0.0001, initial pheromone values have been set to $1/k$ (where k is the number of clusters), $\beta = 2.0$, $\rho = 0.1$, local search probability is

0.001 and the maximum number of iterations is 1000. We have made no attempt to tune the parameters to individual datasets.

The metrics used are the Euclidean Distance for online algorithms (CluStream, Online K-means and ClusTree) and the RBF for the spectral-based algorithms (SACOC and SC+Nyström). The evaluation is performed comparing the results with the real labels to measure the accuracy, defined by:

$$sim(C_i, C_j) = \frac{\sum_{q=1}^{n} \delta_{C_i}(x_q)\delta_{C_j}(x_q)}{2} \left(\frac{1}{|C_i|} + \frac{1}{|C_j|} \right) \quad ,$$

where $|C_i|$ is the number of elements of cluster C_i and $\delta_{C_i}(x_q)$ is the Kronecker δ defined by:

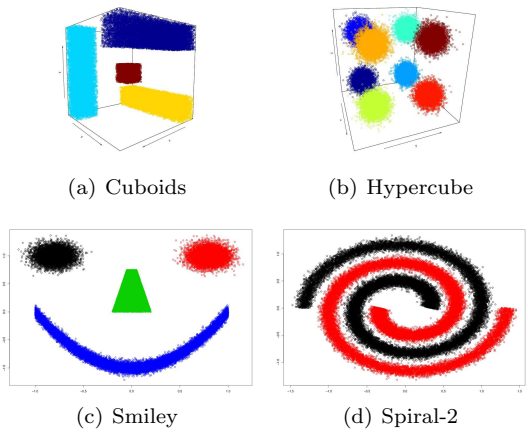
$$\delta_{C_i}(x_q) = \begin{cases} 0 \text{ if } x_q \notin C_i \\ 1 \text{ if } x_q \in C_i \end{cases} \quad .$$

All algorithms have been executed 100 times. The statistical test analysis was performed using the non-parametric Wilcoxon test, which does not require a normal distribution. The aim is to compare SACON against SC+Nyström, since both algorithms use the Nyström extension but differ in the way they cluster the data. We have considered that there is statistically difference when the p-value of the test is lesser than 0.05 (5% significance level).

5.1 Dataset Description

The datasets have been generated using the R package mlbench [24], a collection of standard benchmark problems. The datasets—all composed by 50.000 instances—that have been generated are the following (see Figure 2 for some examples):

- **Cassini**: 2-dimensional problem with 3 clusters (2 external banana-shaped clusters with a circular cluster between them).
- **Cuboids**: 3-dimensional problem with 3 cuboids and a small cube in the middle of them.
- **Hypercube**: 8 spherical Gaussians distributed at the corners of a 8-dimensional cube.
- **Shapes**: a Gaussian, square, triangle and wave in 2 dimensions.
- **Simplex**: 4 spherical Gaussians distributed at the corners of a 4-dimensional simplex.
- **Smiley**: 2 Gaussian eyes, a trapezoid nose and a parabola mouth (with vertical Gaussian noise).
- **Spirals-1**: 2 entangled spirals without noise.
- **Spirals-2**: 2 entangled spirals with noise.



(a) Cuboids (b) Hypercube

(c) Smiley (d) Spiral-2

Figure 2 Illustration of the synthetic datasets used in the experiments.

6 Computational Results

Table 1 presents the results achieved by the algorithms in each dataset. As we can observe, the proposed SACON algorithm obtains very good results overall.

In the case of **Cassini** dataset, SACON obtains the best results—the statistical test shows that SACON is significantly better than SC+Nyström. It is important to remark that the standard deviation (SD) of SACON is 0, which means that these results are stable. SC+Nyström also obtains good results, while the remaining algorithms obtain poor results. The Cassini dataset has two sections that can be easily defined using a Gaussian distribution, but when the number of instances is high the boundaries between these distributions are harder for the discrimination process; SC+Nyström is also sensitive to the noise produced in this situation.

For **Cuboids** dataset, Clustream obtains the best median results (100.0%) followed by SC+Nyström (99.74%), with SACON achieving close results (99.33%). SACON is the most stable of the algorithms in this dataset (0.0066 of SD). The Wilcoxon test shows that there is not statistical difference between the SACON and SC+Nyström.

Hypercube results show that SACON is able to discriminate the clusters perfectly. Clustream also obtains good results during the discrimination process. SC+Nyström, Clustree and Online K-means have problems to discriminate the cluster distribution. This dataset is challenging due to the large quantities of data, which introduces noise during the cluster separation (Figure 2(b)). The algorithms are also less stable than SACON (the SD of SACON is 0). Again, SACON is statistically significantly better than SC+Nyström.

In the case of **Shapes**, the best results according to the median are achieved by Clustream, Clustree and Online K-means. SACON obtains good results (99.98%) and it is also very stable (the SD is 0.0002). SC+Nyström obtains the worst results (68.71%). According to the statistical test, SACON is statistically significantly better than SC+Nyström again.

Table 1 Median and standard deviation accuracy results obtained by each algorithm, measured over 100 executions. Values in bold show the best results. The ▲ symbol indicates the datasets where the results of SACON are statistically significantly better than the results of the benchmark SC+Nyström algorithm.

Dataset	**SACON**	**SC+Nyström**	**Online K-means**	**Clustree**	**Clustream**
Cassini	**99.98%** ± 0.0000 ▲	98.61% ± 0.1681	65.47% ± 0.0003	66.97% ± 0.1301	67.39% ± 0.0478
Cuboids	99.33% ± 0.0066	99.74% ± 0.1736	88.84% ± 0.1219	72.74% ± 0.1108	**100.0%** ± 0.1157
Hypercube	**100.0%** ± 0.0000 ▲	76.71% ± 0.1565	81.36% ± 0.1122	82.29% ± 0.1126	**100.0%** ± 0.0541
Shapes	99.98% ± 0.0002 ▲	68.71% ± 0.4425	**100.0%** ± 0.1389	**100.0%** ± 0.1765	**100.0%** ± 0.0001
Simplex	**100.0%** ± 0.0000	**100.0%** ± 0.0000	**100.0%** ± 0.1508	**100.0%** ± 0.1703	**100.0%** ± 0.0000
Smiley	**99.56%** ± 0.0003 ▲	74.99% ± 0.1627	62.54% ± 0.1729	64.41% ± 0.1608	91.60% ± 0.1451
Spirals-1	**100.0%** ± 0.0000	**100.0%** ± 0.0991	50.00% ± 0.0000	50.01% ± 0.0002	50.01% ± 0.0006
Spirals-2	**63.03%** ± 0.0001 ▲	59.59% ± 0.0392	59.36% ± 0.0000	58.84% ± 0.0295	59.59% ± 0.0392

Simplex is easy for all algorithms. The median values show that they obtain the maximum results. These results are a consequence of the data structure —in this case, they only need to discriminate spheres, which is the simplest clustering problem. There is not statistical difference between SACON and SC+Nyström.

Smiley results show that SACON obtains the best discrimination results (99.59%). The remaining algorithms have problems generating the clusters. This dataset represents a continuity-based problem, therefore only SC+Nyström, Clustream and SACON can discriminate the boundaries (Figure 2(c)). SACON is statistically significantly better than SC+Nyström.

The **Spiral-1** dataset tests show how the algorithms can deal with continuity datasets without noise. In this case, SACON and SC+Nyström obtain the best results (100.0%); Online K-means, Clustree and Clustream have problems discriminating the spirals, which means that these algorithms are not able to deal with pure continuity-based problems. There is not statistical difference between SACON and SC+Nyström.

The **Spiral-2** dataset introduces noise to Spiral-1 (Figure 2(d)). In this case, the results of SACON are worse than in the previous Spiral-1 dataset, as expected, but it obtains the best and more stable results. The results of the remaining algorithms show that there is a good minimal solution and all of them are close to this solution (around the 59%), however, they still find problems discriminating the spirals. SACON is statistically significantly better than SC+Nyström.

6.1 Discussion

SACON shows competitive results identifying manifolds when is applied to continuity-based data—such as Cassini, Shapes, Smiley and Spirals—while the remaining algorithms have problems (e.g., in Spirals, which is a pure continuity-based problem). It also performs better than the rest when we add noise to the dataset. The algorithm also obtains more stable results than the others according to the standard deviation. The spectral transformation in combination with the ACO clustering procedure is probably the main reason for the algorithm performance—the transformation makes the regularities in the data space more apparent and the ACO global search procedure is able to find the (near-)optimal cluster assignment. The performance is correlated to the type of clustering that this algorithm can face, i.e., continuity-based clustering.

In order to compare the memory usage of the proposed algorithm against SACOC (its predecessor), it is important to remark that SACON uses a matrix of $50,000 \times 500$ instances, whereas SACOC uses $50,000 \times 50,000$ instances. The memory usage of the former is around 0.18 GB while the latter consumes around 18.63 GB. The memory consumption of SACON grows linearly, whereas SACOC grows exponentially. The remaining algorithms are not spectral algorithm, with the exception of SC+Nyström—i.e., they do not use a similarity matrix. Regarding the memory consumption, the online algorithms only keep in memory the information about the centroids they are using (similar to K-means), i.e., the memory grows according to the number of centroids.

7 Conclusions

This paper proposed an improvement of the SACOC algorithm [14] using the a Nyström extension [5]. The new algorithm, called SACON, applies the spectral transformations and the Nyström extension to the original search space in order to cluster the data in the projective space. The projection consists on transforming the original data into a graph-based representation (through a similarity graph) and calculate its Laplacian matrix. During this calculation, the Nyström extension guarantees that the Laplacian calculation is accurate enough using less information. Once the Laplacian has been obtained, the eigenvectors are extracted and normalized to generate the projective space—this extraction also applies the Nyström extension in order to reduce the memory usage.

The proposed SACON algorithm shows good results in the studied datasets. It is able to discriminate manifolds and continuity-based clusters with more stable results in dense datasets, when compared to Spectral Clustering using the Nyström extension and modern online clustering algorithms. The future work will be mainly focused on running experiments with real-world

large datasets, in particular large datasets with cluster intersections.

Acknowledgement

This work has been supported by the research projects: TIN2014-56494-C4-4-P, CIBERDINE S2013/ICE-3095, SeMaMatch EP/K032623/1 and Airbus Defence & Space (FUAM-076914 and FUAM-076915).

References

[1] C. Chu, S.K. Kim, Y. Lin, Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. *Advances in Neural Information Processing Systems*, 19:281, 2007.

[2] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proc. 41st Annual Symposium on Foundations of Computer Science*, pages 359–366. IEEE, 2000.

[3] D.T. Larose. *Discovering Knowledge in Data*. John Wiley & Sons, 2005.

[4] H.D. Menéndez, D.F. Barrero, and D. Camacho. A genetic graph-based approach for partitional clustering. *International Journal of Neural Systems*, 24(03), 2014.

[5] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[6] M. Dorigo and M. Birattari. Ant colony optimization. In *Encyclopedia of Machine Learning*, pages 36–39. Springer, 2010.

[7] F.E.B. Otero, A.A. Freitas, and C.G. Johnson. Inducing decision trees with an ant colony optimization algorithm. *Applied Soft Computing*, 12(11):3615–3626, 2012.

[8] H.D. Menéndez, F.E.B. Otero, and D. Camacho. MACOC: a medoid-based ACO clustering algorithm. In *Swarm Intelligence*, pages 122–133. Springer, 2014.

[9] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 29(2):249–272, 2011.

[10] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu. A framework for clustering evolving data streams. In *Proc. of the 29th International Conference on Very Large Data Bases - Volume 29*, pages 81–92, 2003.

[11] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[12] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

[13] U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586, 2008.

[14] H.D. Menéndez, F.E.B. Otero, and D. Camacho. SACOC: A spectral-based ACO clustering algorithm. In *Intelligent Distributed Computing VIII*, pages 185–194. Springer, 2015.

[15] R.S. Parpinelli, H.S. Lopes, and A.A. Freitas. Data mining with an ant colony optimization algorithm. *IEEE Trans. on Evolutionary Computation*, 6(4):321–332, 2002.

[16] F.E.B. Otero, A.A. Freitas, and C.G. Johnson. A New Sequential Covering Strategy for Inducing Classification Rules With Ant Colony Algorithms. *IEEE Trans. on Evolutionary Computation*, 17(1):64–76, 2013.

[17] C. Blum and K. Socha. Training feed-forward neural networks with ant colony optimization: An application to pattern classification. In *Proc. of the 5th International Conference on Hybrid Intelligent Systems*, pages 233–238, 2005.

[18] M. Borrotti and I. Poli. Naïve bayes ant colony optimization for experimental design. In *Synergies of Soft Computing and Statistics for Intelligent Data Analysis*, pages 489–497. Springer, 2013.

[19] Y. Kao and K. Cheng. An ACO-Based Clustering Algorithm. In *Ant Colony Optimization and Swarm Intelligence*, pages 340–347. Springer, 2006.

[20] L. Ashok and D.W. Messinger. A spectral image clustering algorithm based on ant colony optimization. In *Proc. of SPIE (Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVIII)*, volume 8390, 2012.

[21] O.A. Mohamed Jafar and R. Sivakumar. Ant-based clustering algorithms: A brief survey. *International Journal of Computer Theory and Engineering*, 2:787–796, 2010.

[22] P.S. Shelokar, V.K. Jayaraman, and B.D. Kulkarni. An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195, 2004.

[23] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2001.

[24] F. Leisch and E. Dimitriadou. *mlbench: Machine Learning Benchmark Problems*, 2010. R package version 2.1-1.