Optimization of the Performance of Modal Association Clustering: Application on Gaia Astrometric Data

Enrique Utrilla Molina

September 2017

Universidad Nacional de Educación a Distancia

Master en Inteligencia Artificial Avanzada, curso 2016/2017

Trabajo de Fin de Master

Contents

List of Figures

List of Tables

1 Introduction

One of the most common tasks in unsupervised machine learning, clustering is the problem of discerning multiple categories in a collection of objects while the category labels are not given [2]. An intuitive approach is that data points within a cluster are more similar to each other (or at least to some of them) than to data points in other clusters. In general nothing is known a priori about these categories, neither their number nor its characteristics, not even what makes one of them different to the others. Sometimes it is not even known if different categories do actually exist. Nevertheless, in some cases some assumptions have to be done over some or all of these unknowns. In others, even if some boundaries between classes are known beforehand, this information might be deliberately ignored so as to confirm that a difference does exist in the data rather than being just an arbitrary limit.

While this might be a task that any human can do if the datasets are two dimensional and presented in a graphical format, it is much harder to achieve in case that three or more variables are involved. While multiple charts representing two-on-two relationships and selective filtering might help a human analyst in some cases, this can be a daunting task in large datasets, and regular statistical techniques might prove ineffective.

Unsupervised clustering can thus be used to discover relationships (and differences) between examples on a dataset that might not be obvious or known, and that might be difficult to pinpoint among large amounts of multidimensional data. The larger, more multidimensional and more complex a dataset is, the more valuable clustering analysis can be to extract novel structural information.

The most relevant obstacles are the computational cost and the dependency on input parameters. Clustering is typically a costly operation in terms of computation time, and scalability can be a major concern, in particular since as we have just mentioned it would be desirable to apply these methods to very large datasets.

In turn, the performance of the clustering at discovering relevant information is in many cases both controlled and conditioned by some required input parameters. Whether it is the number of clusters to be found, some initial value for prototypes or some kind of limit for distance, these parameters can lead to very different results in both the distribution of points to clusters and the computational cost to obtain it. Some small clusters cannot be detected unless a “fine grain” setting is imposed on the clustering algorithm, but in excess this leads to a proliferation of clusters that do not represent any meaningful relation, but only noise that hides the useful information. This means that sometimes applying a clustering algorithm can be a tedious trial-and-error task until the desired results are found, requiring several runs with different combinations of values for the parameters.

1.1 Types of clustering algorithms

Unsupervised clustering techniques based on unlabelled examples (as opposed to, for instance, nodes in a connected graph, which represent a separate field of work) can be classified in four main families.

The first one is based on pairwise distance between the examples, like linkage clustering [3], or other types of hierarchical clustering. Due to this dependence on pairwise distances, the complexity of these algorithms is typically $O(N^2 \log N)$ for aggregate (bottom-up) methods [8] or even worse, $O(2^N)$ for divisive (top-down) methods. Advanced techniques using Nearest Neighbour graphs have worst-case computation times of $O(N \log N)$ [9]. In some cases, these methods tend to create long chains of points, making hard the definition of any practical separation criteria [10].

The second family of clustering algorithms tries to maximize the similarity to a set of prototypes using some kind of distance or merit function, as in K-means [4]. The goal of this method is to partition the available observations into a finite number of clusters so that each observation belongs to the cluster with the closest mean, which serves as a prototype for the cluster. This problem is NP-hard, so heuristic algorithms are used to find quickly local optimum solutions. These solutions, nevertheless, tend to be of comparable size, rather than mixing clusters big and small or with different shapes. Also, the number of clusters must be known in advance.

The third ones uses statistical modelling, fitting a mixture of “simple” statistical distributions, e.g. a multivariate Gaussian for each cluster, so that it matches the distribution that could generate a population like the one observed [11].

In most cases this translates into finding a set of parameters e.g. the weight, means and covariance of each cluster. Depending on the approach, this can be achieved by using the classification maximum likelihood procedure [12] or the iterative Expectation-Maximization (EM) algorithm [13]. Nevertheless the former has been found to perform better if the population is small with well separated and similarly sized subpopulations, while providing inconsistent results in other cases. For this reason EM is generally preferred.

Another possibility to set these parameters is to assume that they are random variables extracted from some prior distribution and use Bayesian inference to find the most probable values. Since this inference usually cannot be performed analytically, MCMC algorithms are used for estimation.

Except in the case of Bayesian analysis with MCMC, in which the number of components of the mixture can be one more of the parameters to infer, it is generally required some kind of analysis to estimate how much the fit of a model to the data improves by adding an additional cluster.

This parametric approach is nevertheless restrictive in some cases depending on the shape of the clusters in the data. The fit might not be accurate if the subpopulations don’t really follow a simple distribution, which is the case in many real-world applications. Statistical non-parametric methods will be discussed in further detail in section 2.

The last type of clustering methods is based on the connection of points with a similar local density of neighbours. The best known of these algorithms are DBSCAN, created in 1996 [23], and a more general version called OPTICS [24]. The basis of DBSCAN is that, for each point, the number of points within an N-sphere of radius ε centred on it is counted. If this number exceeds a given threshold, the point is considered part of a cluster and all the neighbouring points that meet the same density are connected to it. Otherwise, it is considered “noise”. OPTICS doesn’t actually generate clusters explicitly, but sorts the points in a dataset in such a way that all points that would be grouped together by DBSCAN for any value of ε are contiguous. Thus it removes the dependence on this parameter, which only needs to be selected to generate explicitly the clusters. Several variants of these methods have later been developed to remove dependencies of the parameters and/or improve the performance [25].

These methods typically have a complexity $O(N^2)$ without indexing, $O(N \log N)$ if indexing is available to retrieve the neighbourhood (e.g. using an R-Tree). Another clear advantage is that these methods are non-parametric and can adapt to clusters of any arbitrary shape.

Nevertheless, they rely on sharp changes in the density to detect cluster borders, and are typically outperformed by the statistical methods in case of overlapping Gaussians or smooth density changes.

1.2 Modal Association Clustering

Modal Association Clustering, initially defined by Li, Ray and Lindsay in 2007 [1] represents an interesting and powerful technique for unsupervised clustering. Although it is a mixture-model method, it is non-parametric, so it does not require making any previous assumptions on the number, shape or size of the clusters to be found.

Modal clustering solves this by assuming that the mixture is not a composition of distributions whose parameters must be identified, but a sum of fixed, identical distributions centred on each data point. Through a modified version of the Expectation-Maximization algorithm (MEM), a path from each data point is followed while ascending towards a local maximum in the Probability Density Function (PDF) defined by the mixture of Gaussians. Points whose associated local maximum, or mode, are the same are assigned to the same cluster. This algorithm was called Modal Association Clustering or MAC.

MAC still has some input parameters, namely the standard deviation or “sigma” of each of the Gaussians or kernels. This might be a single value, a diagonal matrix if the standard deviation is different in each dimension, of a full matrix if there are known correlations between them.

A further refinement involves the usage of several levels of clustering, each with different values for the sigma of the kernels. The initial clustering classifies all the input points into a potentially high number of clusters using the smallest values for sigma. In the next level, only the modes of the clusters found in the previous level are grouped using a different mixture, with kernels centred in all the points of the initial dataset, but a larger value for their sigmas. The procedure can thus continue until only one cluster with all the points is left, or no more values for sigma are provided. This algorithm was called Hierarchical MAC or HMAC.

Nevertheless, HMAC is computationally expensive and its complexity on it first level is roughly quadratic. It implies a challenge for bigger datasets that nowadays are becoming more frequent.

1.3 The Gaia mission

One of the large datasets that can benefit from clustering analysis is the catalogue being produced by the European Space Agency mission Gaia [5]. The Gaia spacecraft was launched in December 2013 and is currently orbiting the Lagrange L2 point some 1.5 million kilometres away from Earth, spinning in a scanning law that allows it to survey the whole sky in all directions. Gaia carries two telescopes that allow it to gather information about stars as faint as 21st magnitude, and sends back to Earth an average of some 30-40 GB of compressed data each day.

The goal of this mission is to generate a catalogue of over one billion stars, including 3D position and proper motions with unprecedented accuracy, as well as photometry and spectrometry when possible. For the number of individual observations in which this catalogue will be based, it can be on the order of 10^{11} at the end of the nominal 5 year mission. The daily processing pipeline generates around 300 GB of calibrated, cross-matched observations per day which is distributed to affiliated Data Processing Centres around Europe for further scientific processing [15]. As a result of this process, additional data products are generated, such as photometric analysis, variability analysis and classification, inference of astrophysical properties and characterisation of the orbits and physical parameters of observed Solar System objects such as asteroids and comets.

The accuracy of the individual observations depend on a number of factors that are unknown, such as the true exact orientation of the spacecraft in the moment of the measure with an accuracy of microarcseconds, the correct calibration of the instruments, relativistic effects, etc. Thus, all the available data are processed cyclically to achieve an astrometric solution corrected from all known systematic effects and that minimizes the error while estimating these unknown variables.

The last step in the generation of the catalogue is the integration of all the data available for each source and its scientific validation, filtering out those entries that do not match the required quality criteria.

During this chain of processes, clustering could prove to be beneficial in several points. In particular, a variant of HMAC is being integrated into the astrophysical parameters inference system. The variability analysis system, features a supervised classification algorithm to label different types of causes for changes on the parameters of stars with the time, such as being eclipsed by a binary companion, a planet, of having a natural cycle of varying brightness due to its physical characteristics. This algorithm was considered to be complement with an unsupervised clustering algorithm so that new, yet unknown causes of variability could be singled out for proper classification.

Of course, clustering can also be performed not only on intermediate products based on individual observations, but also on the consolidated records for each star available in the published catalogue. In any case, this involves datasets made up of tens of millions to hundreds of billions (10^7-10^{11}) entries, with typically tens of dimensions. These sizes imply a challenge for the performance and scalability of clustering algorithms.

For instance, the first version of the Gaia catalogue (Gaia-DR1) was released on September 19th, 2016 [6], including 2D positions for 1,142,679,769 objects and additional data, statistics and flags for a total of 57 different columns. Additional tables contain classification and light curve characteristics for a few thousand variable stars of Cepheid or RR-Lyrae types, plus cross-matches between Gaia sources and the contents of several popular star catalogues. The Gaia-DR1 catalogue is publicly available in several sites, and can be queried in the ESA-maintained official Gaia Archive:

https://gea.esac.esa.int/archive/

The Gaia catalogue is being released in an incremental manner as more data becomes available. The first releases of the catalogue include only basic data which can be obtained from just a few observations of each object, such as the measured position. Other parameters require many observations to be calculated with any accuracy, or a large enough time baseline to record slow changes.

For instance, the distance to a star up to 300 or 400 light years away from Earth can be calculated from the parallax, the apparent movement of the object in the sky as it is observed from opposite extremes in the Earth orbit around the Sun. The greater this change in measured position, the closer the star is. Nevertheless there is another factor that produces exactly the same effects, which is the proper motion of the star with respect to the Sun. Over the course of a single year both effects cannot be disentangled. With a longer baseline, the periodic effect of the parallax can be separated from the cumulative effect of the proper motion. The accuracy of this attribution improves with the number of observations and the accumulated displacement due to the proper motion, and thus, with the amount of time between the first and last observations. Gaia-DR1 was based on only one and a half years of observations, and thus it was unable to provide reliable parallaxes for most of the stars contained in it.

For a subset of these objects, though, relatively accurate observations of their positions are available from the Tycho-2 catalogue based on observations from the Hipparcos mission launched in 1989. For most of those stars, Tycho and Gaia observations could be combined to generate a roughly 25 year baseline that allowed calculation of their parallaxes [7].

This subset of the Gaia-DR1 catalogue is called Tycho-Gaia Astrometric Solution (TGAS), and includes over two million stars which can be queried in a dedicated table in the Gaia Archive. At the time of this writing the complete TGAS dataset in CSV format can also be downloaded from the following URL:

http://cdn.gea.esac.esa.int/Gaia/tgas_source/csv/

The TGAS dataset contains the same 57 columns of the regular Gaia sources, plus two extra columns for the associated Hipparcos or Tycho-2 identifier of the star.

1.4 Application of Modal Clustering to Gaia data

The basic premises for this work was the application of HMAC or a similar derived method to a large astronomic dataset, in the range of billions ($\approx 10^9$) of records with tens of different fields in each one. The implementation language should be Java, and it should run in a medium-sized cluster of computers (e.g. around 10-20 commercial blade-type servers) in a maximum time of around one week to be practical.

An initial analysis concluded that modifications to the basic MAC algorithm were required in order to reduce the complexity and allow parallelization in a distributed cluster of computation nodes. Otherwise the computation would take so much as to result unfeasible for any practical purposes.

In particular, a single-threaded implementation of the MAC algorithm (e.g. a single level of the HMAC) in Java mimicking the Matlab implementation provided by the authors of the original HMAC paper (see section 4), ran over a small test dataset of 105 2-dimensional points took 60 milliseconds. Assuming a quadratic computational complexity for 10^7 more points, a simplistic estimation would result in the order of 2 million years to complete the computation. Later, more detailed analyses were consistent with this estimation.

Additionally, 10^9 records of 50 floats each would take up 200 GB of memory, and therefore would not fit in any but the bigger and most expensive computers available nowadays, so algorithm changes and time-consuming memory swapping would be most probably required.

It seems clear that in order to make a practical use of the algorithm under these constraints, some simplification is required to reduce the complexity level from $O(N^2)$, ideally making it closer to $O(N)$, so that the computation time is several orders of magnitude smaller.

After considering different alternatives, a modified algorithm with improved performance was implemented and tested. The result, named SG-MAC, will be presented in the following sections.

Due to the lack of availability of appropriate computing resources it has not been possible to perform any clustering over the complete Gaia-DR1 catalogue or a similar sized dataset. Partly for that same reason, but also in order to compare properly the performance of different alternatives, no parallelization has been used, although it has been taken into account in the design so that it is easily doable. Nevertheless, the design constraints for a billion-item dataset are still considered, and the results of runs of different variants of MAC and SG-MAC on the smaller TGAS dataset or subsets of it will be presented to assess their reliability, performance and scalability.

2 State of the Art

For a wide number of applications, older approaches such as K-means and parametrical Gaussian Mixture Models (GMM) seem to perform adequately. Even if determining the optimum number of clusters is an issue, this seems to be solved by performing the clustering algorithm several times with different values and choosing the one that performs best according to some metric like the likelihood ratio or some information theory criterion.

Nevertheless, these methods run into problems with oddly shaped or overlapped clusters. For those particular cases other, non-parametric methods to estimate the probability density function are better suited. In the family of statistical clustering methods, there are two main methods for this estimation, Dirichlet Process Gaussian Mixture Models (DPGMM) [19], and kernel-based estimation of the probability density function.

The first method models the PDF as a sum of (potentially) infinite Gaussian kernels, each of them with different random parameters (mean and covariance matrix) drawn from a base random distribution that acts as prior knowledge of the system. Another scalar parameter defines the probability that a new sample is extracted from one of the existing distributions, or requires the creation of a new distribution with random parameters, thus allowing for a varying number of clusters. Fitting a set of observations over this model can be performed by obtaining the posterior probability that the parameters of the distribution are the ones proposed given those observations. This can be achieved through the use of a modified Gibbs samples, a type of Markov Chain Montecarlo (MCMC) method [20].

On the other hand, kernel-based estimation of the probability density function, also known as Parzen windows [17] represents a more direct approach. In this case the probability density function is approximated by a sum of N independent, identical kernels centred on each of the observed data points. In this case no fitting of any model is required, but on the other hand every time the PDF needs to be estimated at any point the contribution of the N Gaussians has to be added.

Once the PDF has been estimated, individual data points must be assigned to each cluster. A popular mechanism seems to be Bayesian inference, such as in [18], although it does require defining some prior probabilities that, once again, might impact the performance of the clustering.

An alternative technique is described in [16], where the clusters are derived from maximizing the Cauchy-Schwarz distance measure between them. This maximization is performed iteratively, in a procedure that can be considered a constrained gradient descent algorithm with a different step size in each dimension. This method has a complexity of order $O(N^2)$, but by estimating the gradient only from a sample of the data points rather than from all of them, this complexity can be reduced to $O(N\cdot M)$, obtaining good results with values of M as low as 15% of N. This is the only reference found in the literature of an statistical clustering algorithm with complexity lower than $O(N^2)$.

It can be proven that for certain types of kernels, such as Gaussians, the mode of the mixture of kernels tends to the mode of the (hypothetical) distribution generating the samples [17]. This is an important fact since it is the basis of the HMAC algorithm, since it assumes that the mode of the estimated PDF is the same as the mode of a hypothetical, unknown distribution modelling a specific subpopulation.

Regarding the density-based methods, recent developments include variants that try to remove the dependency from parameters and improve the performance. Comparing the density-based algorithms with MAC/HMAC, it is easy to notice parallelisms between them. The calculation of the density at each point Is equivalent to calculating the value of a mixture of functions that have value 1 at a distance up to dc, which makes the function of sigma, and zero elsewhere. Of course the difference is that MAC is iterative through a gradient, while the density methods use a hard threshold over those values.

In particular, the algorithm described in [25] is able to select automatically the centres of the clusters based on finding points that have both a high density and a high distance to another point that has a greater density than itself.

As previously seen, one of the main problems of the clustering algorithms is their computational complexity, roughly quadratic at least with the number of considered information points. This implies that the required computing power grows quite fast with the size of the dataset. For big datasets containing billions of records, this may imply processing times so long that the application of this algorithm becomes unfeasible for practical purposes.

The optimization of MAC/HMAC itself has not been a subject of debate up to now. The only references available discussing its performance are the applicable sections in the papers of the own authors of the algorithm [1].

The obvious solution to perform the computations in a reasonable time, adding more hardware resources, is a limited option. Vertical scaling e.g. using bigger and faster machines is limited by the availability of such devices and their price. Horizontal scaling, meaning using several processor cores or whole computers in parallel, is more promising, although still expensive and requiring modifications to the algorithm.

The authors of the original HMAC paper suggest a natural way to perform this parallelization: Dividing the dataset in several subsets, run HMAC in each of them separately and then combine the clusters appeared in each subset, grouping together the records of two clusters in different subsets if their mode was the same within a given tolerance.

It must be understood that this implies dividing the data points for which the associated modes will be calculated, but considering in each split the whole set of kernels. For very large datasets this may be problematic, since they might not fit into the available memory. Also, while it allows parallelization, it doesn’t reduce the computational complexity. It could imply trading a quadratic execution time by a quadratic increase in computing hardware requirements.

Other practical factor to consider is the memory requirement. In general, in order to calculate the path from a data point to its mode, it is required to use the information of all the points in the dataset. If the dataset is large enough, it may not fit in the available RAM of the computer running the algorithm, which would require storing parts of the dataset in disk and exchanging continuously data between the file system and the memory. This can be an extremely slow task compared to regular processing, and should be avoided if possible.

Another option for speeding up the computation, also proposed in [1], is to apply an initial clustering to the input data. The exact clustering mechanism to be used is undefined, but its purpose would be to reduce the problem from calculating the paths for N points to calculating the paths for M points, with M<<N. The density estimation would then be weighted mixture of Gaussian kernels centred in the clusters modes or prototypes, and where the weight of each kernel is the number of points that have been replaced by that cluster.

Based in this idea, one of the subsystems of the Gaia astrophysical characteristics processing system performs a variant of HMAC clustering using an iterative procedure. A detailed description of this procedure has not yet been published, but in general it involves performing the clustering operation in parallel in each small area of the sky separately. These areas are defined in terms of healpix indexes, a system to divide a sphere in regular, triangular-shaped pixels with a hierarchy of different levels of resolution [22]. Even if the clustering is not perfect in the borders due to the lack of information of points in other areas (see discussion in section 3.1.2), this reduces the number of points that are considered in a single unit. Then the clustering can be applied on a higher level, with a lower level healpix index and therefore covering a larger area of the sky, using the clusters in the previous iteration to represent the data points.

The drawback of this method is that is tailored for spatial parameters in polar coordinates, or at least data that contain this type of information, even if it is not really used in the clustering. This is a situation which is common in astronomy, but not in all cases.

One problem that has not been addressed in this work is the problem commonly known as “Curse of Dimensionality”. As the number of dimensions grows, clusters can be dissolved by distributing the data points through additional dimensions that do not provide any extra information, e.g. with enough points it would be discovered that they are similarly distributed in that dimension so that it does not help discriminating between different clusters.

Much of the recent work in the field of clustering algorithms is devoted to subspace clustering or other techniques that deal with this potential problem [26]. Nevertheless, for the present work, the application domain is somehow restricted to values with some physical sense, so it is not expected that finding automatically subsets of the available dimensions that improve the results of the clustering is a priority.

3 Analysis of the problem

3.1 Considered alternatives for optimization

When analysing the HMAC algorithm, it is clear that the most time-consuming task is the first iteration on MAC, which requires calculation of the modes for all the points in the dataset. Later stages repeat the process only on the modes of the previous stages and therefore are typically much faster. It seems evident that the optimization of HMAC is really the optimization of MAC, in particular in its first level where the set of data points and the set of kernels is the same.

Thus, the MAC algorithm can be characterised as a double nested loop over the list of points in the dataset, hence its quadratic computational complexity. There is an additional loop for the average number of steps in the path until it converges to a mode, but in principle it doesn't depend directly on the number of points in the dataset and can be seen as a constant factor in the complexity. The computational cost of additional tasks, such as comparing the modes found to select the only the distinct values, is negligible in comparison.

Depending on the purpose of the clustering, it might be considered to use only a subset of the data points as starting points for the paths while using all the points as kernels. That would, in theory, allow discovering most of the clusters, if that is what was required. That would certainly reduce the complexity, at the price that not all points in the dataset would be classified.

If classifying all the points is required, as in this case, the only other way to significantly reduce the complexity is reducing the number of points used as kernel centres while calculating the paths to the modes. This might lead to a different list of clusters with different shapes and associated points. It remains to be seen what is an acceptable similarity; in which circumstances it is achieved; or whether it is important or not as long as the clustering obtained is sensible.

Three mechanisms have been identified that could achieve this goal:

1. Use only a random subset of the points
2. Use some intelligent parametric selection of the points
3. Use a sampling of the PDF

3.1.1 Option 1: Random subset of the kernels

A simple possibility is to sample the number of points used as kernel centres to reduce the number of calculations. This trivial solution assumes that randomly sampling the kernels does not change the overall shape of the PDF. Nevertheless, there are two effects that might prevent this:

a) The distribution of the data points is random but the distribution of the points of each cluster is not homogeneous. That might lead points of an underrepresented cluster to be assigned to a different close, overrepresented cluster.

b) The distribution of the data points is homogeneous enough, but the distribution is so sparse in the sampled kernel distribution that the points do not get to be associated to the same cluster as they would in the complete dataset. This can be particularly important in datasets with a high number of dimensions.

Both these effects should be more noticeable the lower the sampling fraction is, and so it seems to be limited in terms of reducing the computational complexity.

3.1.2 Option 2: Parametric selection of the kernels

Instead of using random sampling, the second alternative is to select some subset of the kernels that are applicable for the path being computed. This approach is based on the hypothesis that the influence of kernels that are far away should be negligible for the calculation of the PDF at any given point.

A possible way to make use of this hypothesis is to select from the whole dataset those points that are in the surroundings of the point of the data space where the value of the PDF is being calculated. Nevertheless, that would involve either iterating over the whole dataset or using some kind of data structure with spatial indexing. In the first case the number of expensive mathematical operations to be performed is reduced e.g. the exponential of the Gaussian is calculated for fewer points, but the quadratic complexity is maintained, as well as the need to either hold the whole dataset in memory or to swap information between memory and disk. In the second, for very large datasets a separate, specialized product (e.g. a database) may be required and probably it would imply the exchange of a large body of information between processes. For datasets that do fit into memory it might be a viable option.

Another option is to split the whole data space into smaller sub-regions and treat them as units of information. The points inside of each of them would be “close” to the path connecting a contained data point to its mode, and those in other areas would be “far” and could be discarded. Depending on the size of these sub-regions this may be far from optimal from the point of view of the path calculation efficiency, since they may still include many kernels with close to zero influence in any given point, but on the other hand it removes the dependency on other data points and reduces significantly the need to load into memory different chunks of the dataset frequently.

This approach is similar to the divide-and-conquer strategy used in the astrophysical characteristics estimation module of the Gaia ground segment. Nevertheless, it should be generalised to be usable with any type of variable, rather than just polar coordinates. Also, the clustering of points close to the borders between regions should be addressed.

The distribution of the data points into subsets following some parametric limits can be a common case in large datasets, e.g. points with $x_i<k$ are placed in one file and points with $x_i>=k$ are placed in another so that the size of the files is kept under a maximum size to keep it manageable. In these cases it might also be impractical and/or time consuming to rearrange the data points in a different distribution. So this approach might be seen as a natural fit for the problem at hand.

As mentioned, the main drawback is that if the limit between two subsets crosses one of the clusters of the overall distribution, the removal of part of the kernels would distort severely the shape of the PDF. Points inside the sub-region would pull the path and not be compensated by the influence of points outside it that are no longer considered. This would result on a larger number of different modes, at least one per sub-region, for each cluster in the overall distribution.

In a hierarchical schema this might not be a major issue since both clusters might be associated in a single unit in one of the higher levels. Nevertheless this is for now an untested hypothesis. If this doesn't happen it is possible that the final distribution of clusters follows the boundaries between sub-regions in an obvious systematic error

In principle it is impossible to figure out in advance where the limits between sub-regions could be placed safely without distorting the overall PDF shape. Otherwise that distribution of limits would be equivalent to defining a clustering solution, and further processing would not be required. Therefore a simple, regular distribution in n-dimensional rectangular sub-regions ("hypercubes") would be used, since that would simplify the distribution of data points and figuring out which sub-regions were adjacent to others.

Since the kernels have a Gaussian shape, it might be assumed that the contribution to the PDF of kernels whose centres were at a distance longer than $k \cdot \sigma$, e.g. three times the standard deviation, was negligible. This is not entirely true, since for instance 13 points at 3σ could have a higher influence on the PDF than a single point at 2σ, but it could be a reasonable work hypothesis. This is an assumption that had to be tested, and it turned out that the area on influence was much wider than expected.

Taking into account the previous point, the shape of the PDF would not be substantially affected if for the data points contained in a hypercube of width w_i the PDF in each step of the path was calculated using the points in the extended hypercube of width $w_i+2 \cdot k \cdot \sigma_i$ and the same centre, with i the index of each of the dimensions of the data space.

This would imply that while there would be no overlap between the data points in different sub-regions, there would be some overlap between the data points considered as centres of the kernels. This would be a slight increment in memory requirements and might have implications on the way the data is stored.

The next problem is that, if the PDF shape is not altered in each sub-region, when executing the MEM algorithm to find the mode associated to a data point the path can lead outside the hypercube of the sub-region. This means that the mode of a point lies in a different sub-region. For the sake of this discussion, it is logically equivalent whether the data of two sub-regions are processed in different computers or computer cores at the same time or in the same hardware resources at different times, and so we will call this a "computation unit". In this case, this means that a path started in a computation unit must continue its processing in another one, and so some kind of communication and coordination mechanism must be set up, complicating the algorithm.

3.1.3 Option 3: Sampling of the PDF

The third alternative considered represents a wider conceptual jump from the base MAC algorithm.

As discussed in section 1, MAC uses a variant of the Expectations Maximization (EM) algorithm to find the mode associated to a point for any given PDF. This is conceptually and, for Gaussian kernels, mathematically equivalent to using the gradient of the PDF in a gradient ascending algorithm. And it must be noted that while sampling the points used to calculate the PDF might change the PDF shape severely, sampling the PDF itself, or its gradient, might not since all points of the gradient would capture some information from all the points in the original dataset.

Although developed independently, this approach is conceptually similar to the stochastic heuristic described in [16] and that reported an $O(N \cdot M)$ complexity.

Therefore a potentially viable approach is to replace the Modal EM algorithm by a gradient ascent algorithm using a sample of the PDF function. Nevertheless, for this mechanism to work, it is required to make a number of approximations and some design decisions.

3.1.3.1 Distance measurement

Several different distance measures are available: Euclidean, Manhattan, Mahalanobis, etc. and one initial decision is which ones of it to use.

Considering that the components of the data points to be clustered in general can have very different variabilities and magnitudes, Euclidean distance doesn't seem to be adequate. Otherwise, small relative changes in values in the ranges of thousands of units would dominate over big relative changes in values measured in thousands of a unit.

For this reason, in general Mahalanobis distances scaled by the standard deviation of the kernels in each dimension (sigma) will be used.

In some cases, for the sake of simplicity and computational efficiency, filtering will be performed on an upper limit for this distance, rather than calculating the distance in each point. In particular, points would be included if the absolute value of the difference in multiples of sigma of each coordinate between two points is lower than a given threshold.

3.1.3.2 Sampling

One of the first decisions would be where to take the samples of the PDF over the whole data space.

An obvious choice would be to use a regular sampling. Nevertheless, even with as few as ten samples in each dimension, with a high number of dimensions there would be more samples than points in the dataset, and the problem would be more complex rather than simpler. It would be trivial to access the value of any point, but memory requirements would be prohibitive.

Therefore a non-uniform sampling must be considered. An option would be to randomly selecting points in the data space, but we opt for just selecting points from the original dataset. If the sample is fully random, it should contain proportional densities of points to the original distribution in each unit of volume, meaning that in areas where the complete dataset contains more kernels and the PDF can change faster the sampling points would tend to be closer together, providing more resolution to follow accurately the PDF.

It must be noted the difference between using a random sample of the data points in the dataset to calculate the PDF in any other point, and using all the data points in the dataset to calculate the PDF in a sample of points of the data space. In the first case some information is been discarded, whereas in the second we are just assuming that the PDF does not change too much between samples, while using all of them to calculate the value at the sample.

The calculation of the PDF value and gradient requires iterating over the whole dataset. Nevertheless the contribution of each kernel is additive, which means that this can be parallelized easily without further complications. The same set of sampled points must be passed to each computation unit, plus any subset of kernel points. The selection of the points in each subset is irrelevant. The results of each computation can be combined by simple addition. For each sample point we will calculate both the value of the PDF and its gradient.

3.1.3.3 Number of samples

In principle, the sample can be taken over any number of points in the data space. The number of samples might be the same as the number of points in the dataset, or it might be just a fraction of them.

Therefore, if for a dataset of N points the gradient of the pdf is sampled at M points, the computation of the gradient at any given point for the calculation of the next step in the path to the mode is reduced to finding the closest of the M sample rather than considering the N samples. If M<<N the computational complexity is reduced accordingly and gets closer to O(N), actually O(N · M) rather than O(N^2).

Reducing the number of samples too much might be a risk, though. With fewer samples, resolution is lost, the local maxima of the PDF* might be displaced and the resulting clustering might be quite different.

Actually, as will be discussed later, this decimation of the samples can take place in two different moments. The first one reduces the number of points in which the PDF and its gradient is calculated, whereas the second selects which of those samples are used to calculate the paths from each point to its mode.

3.1.3.4 Interpolation

With a regular grid of samples the typical approach to estimate the PDF at any given point of the data space would be to interpolate the values of the PDF sampled at the vertices of the hypercube containing the point, weighted by the distance between the point and each vertex. Nevertheless, in a non-regular, random sampling this becomes more complex.

A proposal to simplify the calculation is just to use the value of the gradient in the closest sample of the PDF. In this way, the whole data space is decomposed using a Voronoi tessellation and approximating the PDF by a function in which the gradient is constant in each tile.

Other approaches, such as considering the weighted or unweighted average of the k closest points would be also possible, but their performance has not been assessed in this work.

A key aspect of this approach is nonetheless that the tessellated PDF, PDF*, is no longer continuous. This has an impact in that the basic gradient ascent algorithm can no longer be used.

3.1.3.5 Gradient Ascent and convergence

The gradient ascent algorithm relies on the magnitude of the gradient being zero at any local maxima to reach convergence. In this case, except in the unlikely case of the samples of the PDF being right on top of one of these maxima, this is not generally true. Therefore a new convergence criterion must be defined.

Please notice that this is just an approximation to the full MAC algorithm that is expected to not be too inaccurate. In particular, paths are no longer guaranteed to end up at the same point within a tolerance defined by the numerical precision of the host computer. Some might never converge to any given point, but end up “orbiting” around a point that they never reach due to the error committed by assuming the gradient value is the same as in the closest point. This should not be a problem in the initial steps of the path, but it would be in the last ones close to the final convergence where the increments are small.

In order to prevent infinite loops, it will be defined a maximum number of iterations for a path to reach convergence. The points whose path does not reach convergence would be assigned to a global cluster named “NOT_CONVERGED”, without a defined mode.

There are several possible convergence criteria that could be used. One of them is to keep in a list of the last k positions, and calculate an average of the distance in each step as:

$$\overline{\Delta x} = \frac{1}{k}\sqrt{\sum_{i=0}^{d}\frac{(x_i(t) - x_i(t-k))^2}{{\sigma_i}^2}}$$

Where d is the number of dimensions of the data space, $x_i(t)$ is the coordinate I of the point in the path at the iteration t, k is the number of steps considered, and σ_i is the standard deviation of the kernel Gaussians in the coordinate i.

That is, the Mahalanobis distance between the current position of the path and the position it had k iterations ago, divided by the number of iterations in between. If the algorithm is progressing in more or less the same direction, this value should be similar to the magnitude of the increment in each individual step. If, on the other hand, the path is stuck orbiting a point, the difference between the current point of the path and the one k iterations ago should be moderate, and divided by a large k should become small to mark convergence.

Nevertheless this mechanism didn't achieve good results in the experiments. The size of the "orbit" around a maximum point might have different radius, so in order to make the mean increment small, the number of steps is the path considered (k) should be large. For the sake of simplicity and performance, a different criterion has been used.

The proposed solution is to calculate in advance which of the sampled PDF points is a local maximum compared to the rest of nearby sampled PDF points. For each point, it is possible to identify it as one such maximum by inspecting all the samples that are closer than a threshold distance, typically a multiple of the standard deviation of the kernels. If the value of the PDF in each of the samples that pass the filter is lower than or equal than the value of the PDF in the point, it is considered a local maximum. This is similar to the criterion used to identify the centre of clusters in a density-based method described in [25].

As a work hypothesis we assume that the exact value of the mode is not important as long as the same value is assigned to all the points of the cluster, and to no others. Therefore it doesn't matter that the sampled point is not the actual maximum of the function, as long as it is close enough to the real maximum.

This criterion is subject to further improvements if, for instance, the number of points at the specified distance is small, or the gradient is small, leading to local maxima being defined by what amounts to noise levels. Using a large threshold distance reduces this effect, but also fuses local maxima that are close to each other, and thus reduces the number of possible clusters to be obtained.

It must be noted that with this definition, it is possible to classify as a local maximum a point some distance away from the actual maximum of the estimated PDF depending on the disposition of the rest of points. Figure 3-1 presents such a case. The chart on the left presents the disposition of the sample points. The size of the blob is related to the value of the PDF at that point. The chart on the right displays the sampled values of the gradient.

The gradients hint at two maxima very close together, one at (74, 2.5) and another at (71, 3). Our algorithm will identify one local maximum, in blue in the left figure, which has an only slightly higher value of the PDF than its neighbours. Nevertheless, the sampled gradients will still push a path coming from the right of the figure to the rightmost maxima. This must be accounted for by the convergence criteria, or it would result in non-converged data.

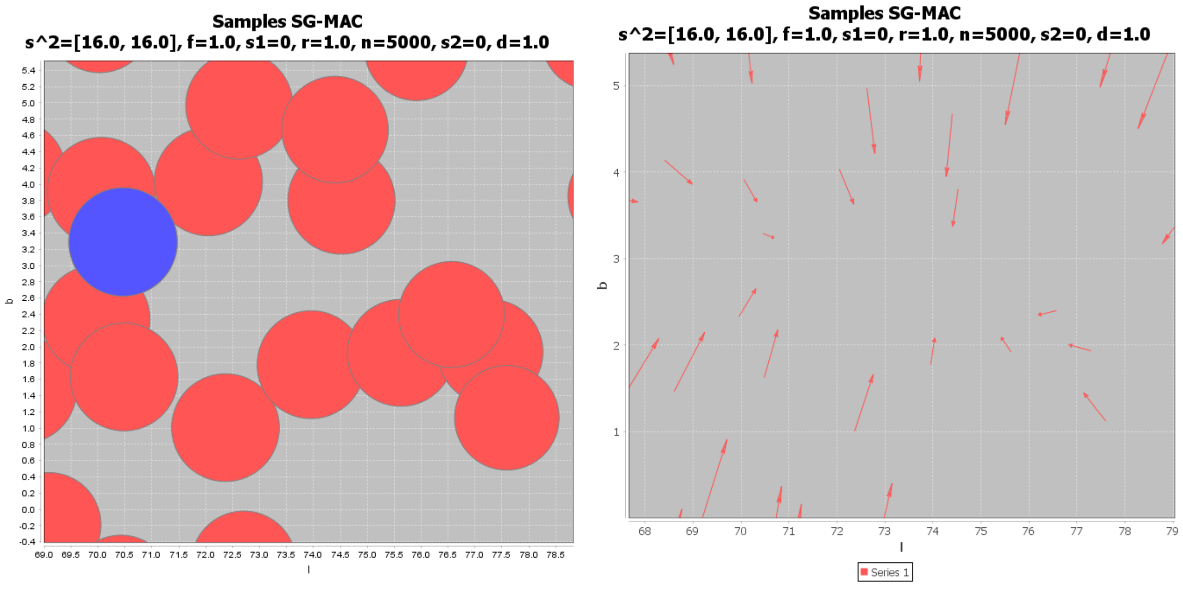


Figure 3-1: Sample position and gradients

Once the local maxima are identified, they can be used as convergence criteria: once the path from a data point to its mode gets close enough to a local maximum, it can be assumed that it is the mode that was being looked for.

In this case, “close enough” equals the threshold distance used to calculate whether a sample is a local maximum. Otherwise, the path might end up orbiting the real PDF maximum, at a position that contains no sample and that might be at some distance of the sample with the higher PDF value.

Please notice that these local maxima will receive at the very least one mode (the one from the point itself) and therefore the number of local maxima will equal the number of clusters of the distribution, with the possible exception of the extra cluster created if any of the points does not converge. This is a useful piece of information that, with this algorithm, can be obtained far in advance of the clustering of all the data points, and that indeed might be everything that is required for some applications.

3.2 Criteria to assess the alternatives

The criteria to accept any of the previous alternatives are the following:

a) Results similar enough to the basic MAC algorithm
b) Linear or close to linear computational complexity
c) Limited memory requirements
d) Ease of implementation and parallelization

The criterion a) is somewhat relaxed and does not require that all the results are identical to the base case, for two reasons:

- A reduction of the complexity involves using approximations and simplifications, so it is to be expected that some results can be different. Nevertheless, good simplifications should keep these differences to a minimum.
- In some cases it might be just impossible to know the distribution that MAC would yield, since it would take too long to compute it.

3.3 Assessment of the alternatives considered

The first alternative is discarded almost immediately since it is not possible to guarantee that the output would be even similar to the results of the basic MAC algorithm in all cases, since it is possible to obtain a very dissimilar PDF.

The second alternative should minimize the distortion of the PDF, and therefore provide close enough results to the original. The complexity should be notably reduced if the sub-regions are small enough, which is also good for parallelization. But the management of the data to distribute the overlapping points in the extended hypercubes and to ensure that memory constraints are not violated, as well as the coordination between computation units when paths cross from one hypercube to another would be complex to implement and maintain in a distributed environment.

The third alternative has a higher risk of distorting the PDF than the second, but it is considered likely to have good enough results. The memory requirements are easily controlled by choosing the size of the PDF sample, and no constraints are put on the distribution of data points into subsets. It is easy to implement and parallelize, and the computational complexity with respect to MAC should be reduced. Also it is possible to use it both in small datasets in a single computer and in large datasets in a distributed cluster.

Considering this assessment, the option selected to be implemented and tested for this work was the third alternative. If needed, e.g. if reducing the sample size cannot make the description of the PDF estimation fit into the memory of the computing nodes, the distribution into hypercubes described in option 2 could also be later combined into its clustering stage.

4 Implementation of the proposed algorithm

The modified algorithm, SG-MAC (Sampled Gradient – Modal Association Clustering) has been implemented in Java. The reference implementation can be downloaded from:

http://github.io/enriqueutrilla/sgmac

For the sake of comparison the original MAC algorithm has also been re-implemented after the original reference implementation in Matlab that can be found in:

http://personal.psu.edu/jol2/hmac/

4.1 High level description of the algorithm

Even though it is critical to reduce the computational complexity for the algorithm to be usable at all, parallelization can decrease the total computation time by several orders of magnitude and it is important that this algorithm is designed so that it is easy to do so. The final design is composed of several sequential stages, which can be parallelized although as previously mentioned it has not been done as part of this work.

In the following description, N denotes the number of data points in the complete dataset and M_1 and M_2 the number of samples of the PDF taken after each of the decimations.

Similarly, *sigma* or σ is the standard deviation of the kernels. It can be either a single value, a list of values for each dimension, or a matrix if there is cross-correlation between them.

4.1.1 Stage 1: Pre-selection of points for PDF sampling

As discussed in section 3.1.3.2, if the number of data points is high enough it is possible to select only a subset of them as points where the PDF estimated from the Gaussian mixture will be sampled.

This could be achieved by random selection without replacement. Nevertheless, in many cases it is simpler to just decimate the data points, that is, take only 1 out of each K points. In the same way as pseudorandom processes can be initialized with a seed either to guarantee different values to be returned in each run or, on the contrary, a repeatable behaviour, a decimator can be initialised with the first index from which the samples start being counted.

The decimation method is simple, deterministic and fast, but has the disadvantage of not being too flexible when specifying the sampling rate. If the number of points is reduced at all, the minimum rate implies taking one out of each two, so a factor of 0.5. Increasing intervals between selected samples equates to ratios of 0.33, 0.25, 0.2, 0.16, etc. The lower the sampling factor, the more accurately the decimation is able to achieve it.

In practise this is usually not a problem, since reducing the number by a high ratio e.g. 0.9, does not improve much the computation time. Indeed, although using only a fraction of the points as samples of the PDF gradient is usually possible without degrading much the clustering solution, during this stage there is a real risk of removing some of the local maxima, and this can have a larger impact. Therefore, for this stage it is usually safer to avoid any decimation and simply take all the data points as points where the PDF will be sampled.

The complexity of the decimation, if it is run at all, is O(N), with N the total number of samples. This stage can be parallelized by splitting the whole dataset in k subsets, drawing M/k values from each one, and concatenating the resulting lists. Nevertheless, due to its simplicity, this should only rarely be required.

4.1.2 Stage 2: Calculation of the PDF and its gradient at each sampling point

Once a list of points is selected for the sampling in the stage 1, it is needed to calculate the estimation of the PDF and its gradient at each of those points.

To estimate the value of the PDF at any point, the original MAC algorithm would iterate over all the data points to calculate the contribution at that point of a multivariate Gaussian kernel with the coordinates of that data point as mean and the common set of values for their standard deviation. In this case, it is possible to optimize this step by iterating only over those points that are in the close vicinity of the sampling point, assuming the contribution of all other points is zero. This requires some way to spatially index the data. One of such mechanisms is discussed in section 4.2.3.

In order to calculate the PDF and the gradient the MEM algorithm as defined in [1] is used, but just a single time in each point. The result of this algorithm is a sum of all the kernel contributions, which is either the PDF or proportional to it; and a new position for the next step in the path towards the mode. The difference between the new position and the current would be used as gradient. It can be proven that this is mathematically equivalent to the gradient as partial derivatives in each dimension of the sum of Gaussian functions.

In particular, for each sample point s,

1. All the data points in the neighbourhood of the sample point are selected.
   This could be based on the Mahalanobis distance, but for performance reasons an slightly bigger selection is performed with the shape of an hypercube. See section 4.2.3 for details.

2. For each data point x_i a contribution p_i is calculated as an unnormalized Gaussian PDF:

$$p_i = e^{-0.5\,\sum\frac{(s_k - x_{ik})^2}{{\sigma_k}^2}}$$

3. All the values p_i are accumulated into a variable c:

$$c = \sum_i p_i$$

4. For each coordinate k, the updated value in the gradient ascent path would be:

$$s'_k = \sum_i (\frac{p_i}{c} x_{ik})$$

5. Therefore, the value of the gradient for each coordinate k is:

$$g_k = s'_k - s_k$$

The unnormalized exponential of the Gaussian function can be used because for the calculation of the update the unnormalized probability is divided by the unnormalized sum. Since the normalization factor for all of them is the same in all cases because all the kernels use the same covariance matrix, it would be cancelled out. That saves a multiplication per data point which is not required.

As for the use of the PDF value in later stages, only the relative value between points is required, thus rendering the lack of normalization factor irrelevant.

This stage is computationally expensive. In a way, it means moving away calculations from the clustering stage and caching them in the initialization. Its complexity is $O(N^2)$, or $O(N \cdot M_1)$ if there was a decimation in the stage 1.

It can be parallelized in two different levels: In the first place, the list of points where the samples are calculated can be split into K subsets, which can be processed independently. When all the subsets are done, the results of each one can be simply concatenated.

Another possible parallelization involves splitting not the sampling points, but the data points. In this way, each parallel unit would compute only part of the total contribution for each sample. To combine the results of each unit, the PDFs are just added, while the gradient is the weighted mean of each partial gradient, where the value of the PDF contribution is the weight of each unit.

Both parallelization schemes can be combined, with a different computation unit for each combination of data points split and sample points split.

4.1.3 Stage 3: Identification of local maxima

As discussed in section 3.1.3.5 the selected convergence criteria involves identifying the local maxima of the PDF function.

Considering that the estimation of the PDF might be noisy, and it is sampled, there are several possible definitions for what can be considered a local maximum. In this implementation, a point is considered a local maximum if no other sample point has a higher value of the PDF at a Mahalanobis distance lower than a parameter D. In case there are isolated points with no more sample points in that distance, it is expanded until at least one more point is available for comparison.

Other criteria might replace or complement the one described. For instance, it might be required that the gradient fell below some threshold, since at the maximum it is supposed to be 0, or that the gradient in the closest k neighbours points in the general direction of the candidate rather than away, but in the multiple tests performed the simple absence of greater PDF values provided the better results.

This stage as defined also has a complexity of $O(N^2)$ (or $O(M_1^2)$ in case of first-stage decimation). Nevertheless it is possible to use again spatial indexing to reduce the complexity to $O(N \cdot k)$, k being the mean number of points returned for the neighbourhood of the point

This stage is parallelizable by splitting the list of sampled points into different computation units. Nevertheless, each of them potentially requires access to all the other sampled points. If this was not possible due to memory constraints, for instance, the split should be done based on compact sub-regions of the data space so that the rest of points can be selected as an expanded hypercube as discussed in section 3.1.2.

4.1.4 Stage 4: Selection of sample points to use during the clustering calculation

At this point it is possible to perform a second random sampling/ decimation of the calculated PDF samples. For the same reasons already discussed in section 1, a simple decimation has been selected. An important difference in this case is that identified local maxima are excluded from the decimation and added always. Otherwise, bumps in the shape of the PDF might lead the path of some points towards them or “orbiting” around them without never reaching the vicinity of any identified maximum, slowing down the processing without ever reaching convergence.

Apart of that, tests show that decimating the samples at this point (but not before) reduces computation time and memory requirements with a low or moderate impact in the clusters calculated.

As with the stage 1, it is generally possible to be parallelized, although it should seldom be required.

These four first stages are considered the initialization of the algorithm.

4.1.5 Stage 5: Path calculation and classification of each data point into a cluster

The actual clustering is performed in a similar way to the regular MAC clustering. For each data point:

1. A Path point is initialized on the position of the data point.

2. If any of the points in the list of local maxima is closer than a given threshold, the position of the point is updated to that of the local maxima directly.

3. Otherwise, the closest sample point to the path point is selected.
   To do this efficiently, a narrow neighborhood is selected to calculate the pairwise Mahalanobis distances and select the smallest one. If no points are found the size neighborhood is increased until a closest sample is found.

4. The gradient at the path point is approximated by the gradient sampled at the closest point found

5. An increment is calculated as the value of the gradient multiplied by an optional stepFactor.

6. If the norm of the increment is smaller than a threshold, the path is also considered as converged, much in the same way as the original MAC

7. The position of the path point is updated with the increment

8. The process is repeated from step 2 until convergence is achieved, or a maximum number of iterations are reached.

9. For each point, the number of iterations until convergence is achieved is stored to generate some basic statistics at the end. This can be used to detect cases where the maximum number of iterations is too low and is causing an excess of not convergences.

This basic algorithm can be modified with different alternative approaches e.g. different convergence criteria as discussed in section 3.1.3.5.

4.2. Description of the implemented software

4.2.1 Main classes of the implementation

The core of both MAC and SGMAC is implemented by the same class, `ModalClustering`. This class implements the iteration over a number of input sources to find the mode associated to it, and the assignment to either an existing cluster with the same mode, a new cluster if no previous data point was assigned to that mode, or the special cluster "NOT_CONVERGED" if the path has not converged to any local maximum in the specified number of iterations.

The path calculation is performed in each case by a different subclass of the abstract class `PathUpdater`. In the case of the basic MAC algorithm, the class `ModalEmUpdater` is used. This class has as initialization parameter in its constructor the list to all the points in the dataset that will represent the centers of the Gaussian kernels, plus the standard deviations in each dimension.

On the other hand, for SG-MAC a different implementation is used, called `SampledGradientUpdater`. This class implements the gradient ascent algorithm using the gradient of the closest sample, as well as the convergence criterion. This class takes as initialization parameter not the centers of the Gaussian kernels, but the samples of the PDF to be used, which have to be previously calculated.

Clustering itself is only the last of the five stages defined in section 4.1. This means than initialization is rather more complex in SG-MAC.

Both potential sample decimations (stages 1 and 4) are implemented in the class `SampleSelection`. The calculation of the PDF value and gradient samples (stage 2) is implemented in the class `PdfGradientSampler`. The identification of the local maxima (stage 3) is implemented in the class `PdfLocalMaximaSearch`. All the initialization is performed in a public method that can be called independently of the clustering itself, and both the PDF samples and the subset of them that are local maxima are stored for reference, so that they can be stored or presented in a chart. Please note that if the mode of the clusters was the only information required, no actual clustering would be required to take place with SG-MAC.

4.2.2 Data model

In order to hold the data, an inheritance hierarchy of classes has been defined:

- `Point`: Basic class that contains an array of `doubles` representing the coordinates of a point in the data space. It is immutable, so the coordinates cannot be changed.
- `DatasetPoint`: Subclass of `Point` which contains also a `long` unique numeric identification.

- `SampledPdfPoint`: Subclass of `DatasetPoint` which contains also variables for the value of the PDF, its gradient, whether it is a local maximum and whether its path converged to a mode.
- `PathPoint`: Subclass of `Point`, used to store the position at each step of the path traversed looking for the mode. It is the only type of `Point` that can modify its coordinates once it is created.
- `Cluster`: Represents a full cluster, and contains an array of `double`s representing the mode plus a `List` of the `DatasetPoints` associated to it.

4.2.3 IndexedDataset

A key aspect of the optimization is the capability to operate on a smaller number of objects based on its distance to a given point in standard deviations. The class `IndexedDataset` implements this feature.

There are a number of different algorithms that could have been used, such as R-Trees [27]. Nevertheless, for the sake of simplicity of implementation, a similar data structure is used. This structure creates a regular grid that is filled in dynamically, thus supporting sparse structures. The query performance is probably lower than that of other, more complex solutions, but adequate for the purposes of this work. Any other implementation able to retrieve a set of the points at a maximum distance d of a central point could be used instead.

Theoretically, the use of the indexing and filtering should reduce the complexity from $O(N^2)$ to only $O(N \cdot \log N)$

This class takes a List of `Point` or any of its subclasses, and populates with its contents a multilevel tree. There might be a layer of tree nodes for each coordinate of the dataset. It is possible to define a limit so that only the first k coordinates are indexed.

Each node of the first level defines a slice of the dataspace with a width of $w \cdot \sigma_1$, being w a parameter of the class and σ_1 the standard deviation in the first coordinate. The child nodes under it represent slices in the second coordinate of that same slice of width $w \cdot \sigma_2$, etc. In each tree node, the list of children nodes is stored in a `HashMap` with an `Integer` index.

When indexing a point, each coordinate is discretized by taking the integer part of the coordinate divided by the width of the cell in that coordinate. That is the index to be used for the node, so all the points whose i coordinate is in the range $[n \cdot w \cdot \sigma i, (n+1) \cdot w \cdot \sigma i)$ will end up in the same subtree defined by the node with index n. If a slice does not have any points, no node is created for it.

Nodes that are leaves of the tree (e.g. they contain no child nodes) contain instead a list of `Points`. Those are the contents of a hypercube of widths $w \cdot \sigma_i$, i=0…d.

It is possible to limit how many dimensions are used in the indexation, so that a node in level k will become a leaf node even if there are more dimensions. Otherwise, with high dimensionality datasets, the lower levels would tend to contain a node for each single value, which takes both a computational and a memory cost. The tree keeps some statistics on how many nodes are created in each level, so that it is possible to detect the cases in which the number of tree nodes in a level is comparable to that of the number of data points and it is advisable to use less dimensions. In this work this limit was set to 2, so only the first three dimensions (if available) were used in the indexation.

While retrieving the neighbourhood of a point, the process is reversed, but at each level instead of returning the contents of the node associated to the index n, the results of the nodes n-1, n-2,…, n-k, and n+1, n+2, …, n+k are also included, k being the ceiling of the division between w and the desired number of standard deviations around the points. For instance, if w is 2 and the 8-sigma neighbourhood is requested, at each level all Points in nodes from n-4 to n+4 will be included. Thus, this mechanism implements a crude spatial indexation mechanism to retrieve points at a given maximum sigma-corrected Manhattan distance, e.g. with w=2 and neighbourhood=8, a 18-sigma wide in each dimension hypercube roughly centered in the reference point specified.

One of the working hypotheses was that considering only a 3-sigma neighbourhood would not affect significantly the calculation of the path. Experiments revealed that this hypothesis was not correct, and the clustering provided different results in the basic MAC algorithm with and without indexing. Some tests suggested that the effect disappeared by using an 8-sigmas neighbourhood, in some datasets, while others required up to 14. In the case of the tests with SG-MAC, 8 sigmas seemed to achieve a good balance between accuracy and performance, and so it was used as default value in all the tests unless specifically noted.

5 Results of the tests

In general, it is not easy to assess how well a clustering algorithm works. A possibility is to compare the solution with the output of another clustering algorithm or some external distribution to generate an aggregated statistical measure of similarity, if the categories of both datasets can be easily matched. Nevertheless, human inspection is usually required, even though it can be subjective.

In order to assess the results generated by SG-MAC, a number of aspects needs to be covered:

- In order to check the "correctness" of the solutions, a first step would be to compare the results of SG-MAC to those of the basic MAC algorithm and check that they are at least similar or, in case they are not, assess under which conditions the approximations done in SG-MAC are not valid.

  In order to compare these solutions we will use human inspection of the similarity of the generated clusters. To facilitate this task, we will restrict to simple, 2D clustering with unrealistic values of sigma that produce few large clusters on relatively small- or medium-sized datasets.

  Indeed, an even more basic step is to compare the results of our MAC implementation while replicating the experiments with the results reported in [1] to ensure that MAC produces the expected results in the first place.

- In order to assess the performance improvement, the computation times in the previous step will be measured and compared.

- To evaluate the scalability of the solution, even larger datasets will be processed with the faster versions of the algorithm so that trends can be extracted and extrapolated.

- As a second and final evidence of functional performance, the algorithm will be run on a more realistic multidimensional example trying to detect a previously known cluster in the data.

Therefore these experiments are divided in the following sections:

1. Tests using a small synthetic dataset used in the original HMAC paper (test6) to compare the results reported there with the solutions obtained by the Java MAC and SG-MAC implementation as a minimum baseline.

2. Tests using different decimated versions of the TGAS dataset with unrealistic values of sigma to evaluate similarity and computation times.

3. Tests using non-decimated TGAS data or subsets of it with realistic values of sigma to perform multidimensional clustering.

5.1 “Test6” clustering

The test6 dataset is a synthetic dataset generated by the authors of HMAC and available for download as part of the Matlab sources in the URL specified in section 4. It is composed by 200 2/dimensional points. The usefulness of this dataset is that an example of the first level of clustering with HMAC is available in [1], and so it can be used to validate the MAC implementation in Java.

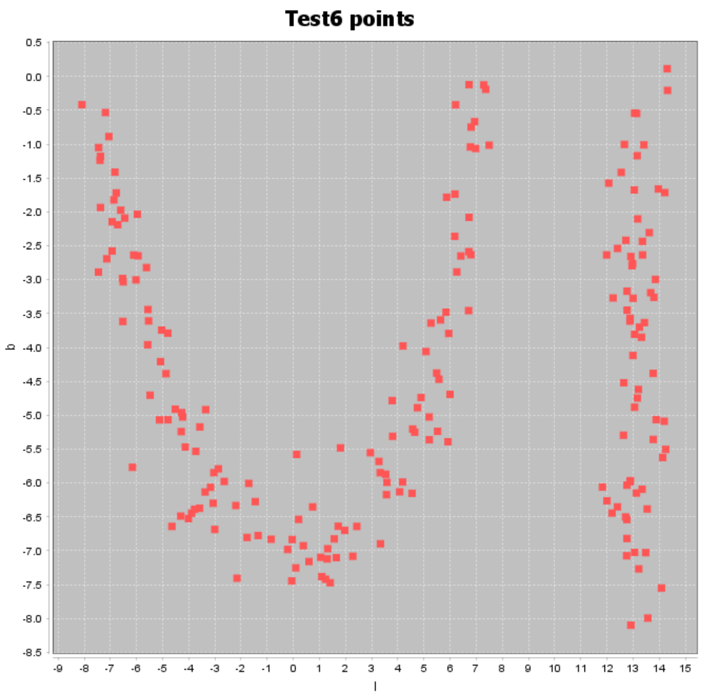


Figure 5-1: Points of the test6 dataset

The result of the clustering with MAC, using a value of sigma of 0.2 for both dimensions matches exactly the clustering solution in [1]:

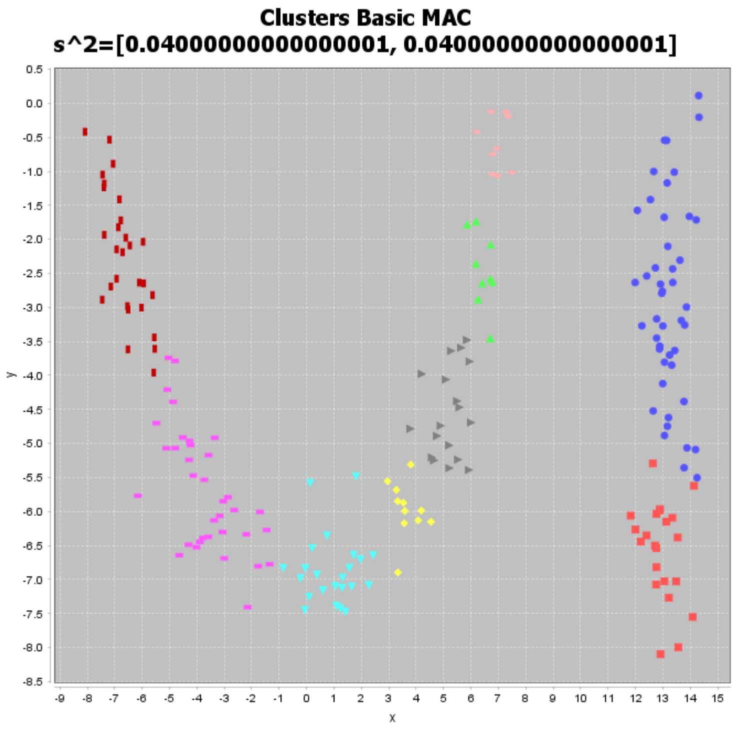


Figure 5-2: Clusters generated by MAC

The first validation step is to check whether MAC with the indexing feature generates the same clustering solution. It turns out that with a neighbourhood of 3 sigmas the results are quite different.

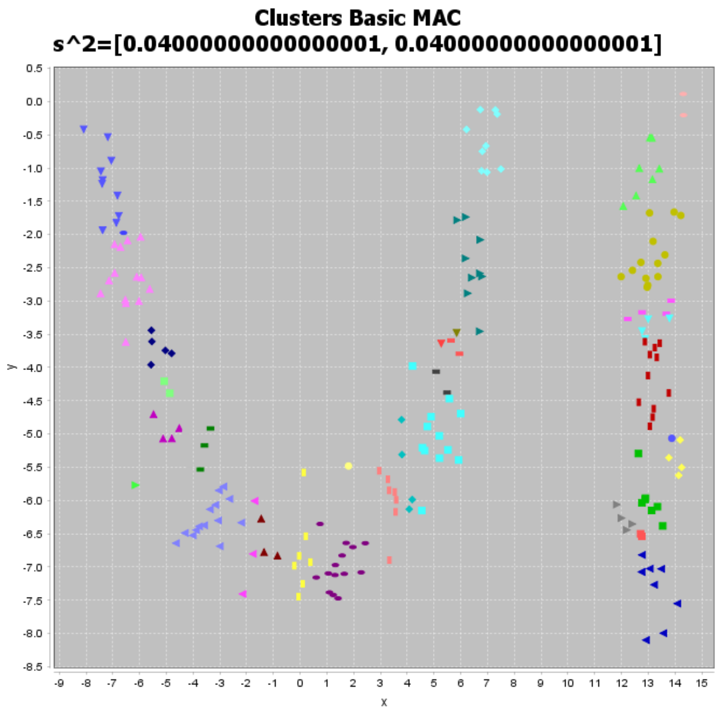


Figure 5-3: Clusters generated by MAC with indexing and a 3-sigma neighbourhood

Increasing the neighbourhood size to 8 sigma renders better results although not yet perfect (Figure 5-4). It is not until a neighbourhood of 14 sigmas is used that the clustering result is identical to the case without indexing. In this case, that corresponds to a square 6-units-wide centred on each data point (Figure 5-5).

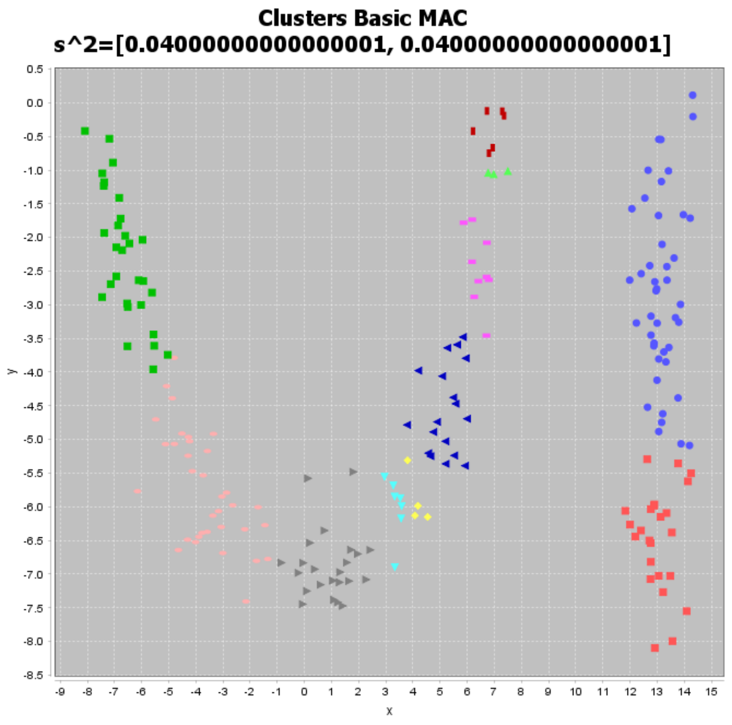


Figure 5-4: Clusters generated by MAC with indexing and an 8-sigma neighbourhood

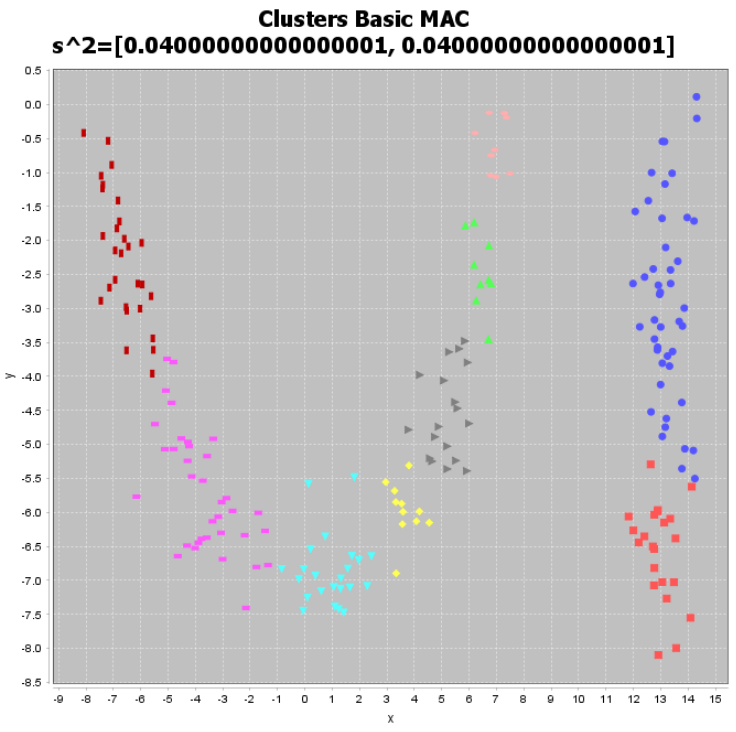


Figure 5-5: Clusters generated by MAC with indexing and a 14-sigma neighbourhood

The next step is using SG-MAC instead of the basic MAC algorithm. For this run we use a minimum separation between local maxima of 7 sigmas, and no decimation since the number of points is so small. The PDF calculation is done with a neighbourhood of 8 sigmas, since a larger neighbourhood does not introduce any changes in this case.

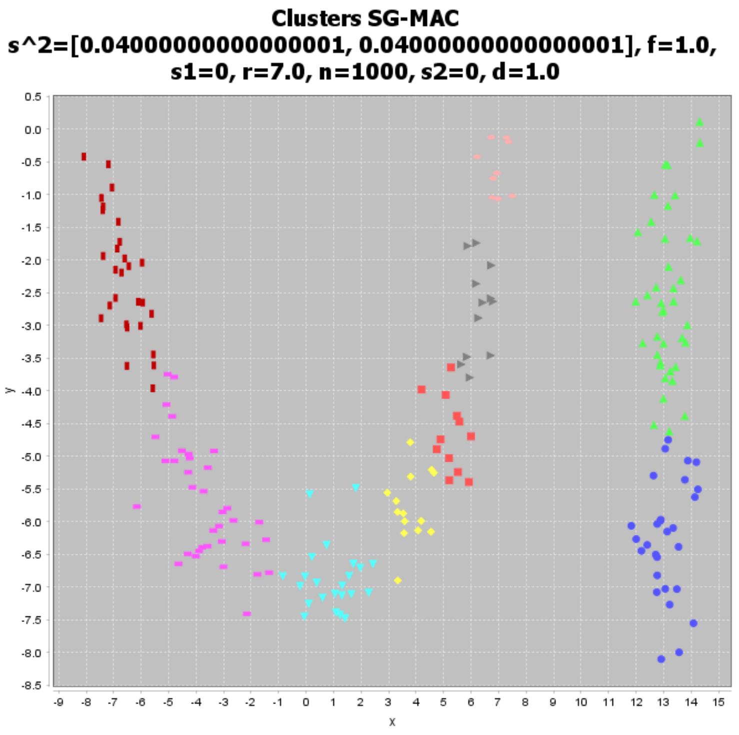


Figure 5-6: Test6 clusters generated by SG-MAC

The first conclusion that can be derived is that SG-MAC does generate successfully clusters that are quite similar to those from MAC for this test case, although not exactly the same. Points that are grouped in a different way are located in the boundaries between clusters identified by MAC, so the result makes sense and is usable.

5.2 Unrealistic TGAS clustering

For this first batch of tests several different versions of the TGAS dataset will be used. The complete dataset contains 2,057,050 entries with 59 columns each in 15 files. From that complete dataset we select 11 columns, although only two of them are used for now tests, the latitude and longitude of the star in galactic coordinates. Several datasets of varying length are generated by taking one out of each 20, 40, 100 and 200 points in each file to obtain datasets of roughly 100,000, 50,000, 20,000 and 10,000 points.

As can be observed in Figure 5-7, the full TGAS dataset represents a distribution of stars concentrated along the galactic latitude 0 (the galactic plane), with less and less stars closer to the North and South poles. Nevertheless, almost all 1-degree-wide cells have at least one star, as can be seen in Figure 5-8, where the density of stars is presented in logarithmic format. Therefore, when displaying the points of the distribution, rather than the density a solid wall of colour is obtained (Figure 5-9).

In the density map, some structures and threads can be seen outside the galactic plane. These structures are not real, but an artefact caused by the fact that Gaia has not scanned those regions as frequently as others yet, and therefore the density of stars observed a large enough number of times is lower.

The list of TGAS-based datasets generated is presented in Table 5-1.

Dataset name	Decimation	Size
TGAS 10000	200:1	10,288
TGAS 20000	100:1	20,574
TGAS 50000	40:1	51,432
TGAS 100000	20:1	102,865
TGAS full	1:1	2,057,050

Table 5-1: List of TGAS-based datasets

The goal of the tests with a dataset as complex as TGAS is to compare the performance and the high-level similarity of the results. Therefore the parameters of the algorithm have been selected so that the number of clusters and the similarities and differences of the shapes can be identified easily. In particular, in the tests of this section only the galactic latitude (“b”) and longitude (“l”) are used, despite additional data being available, to prevent additional dimensions from interfering and interleaving the examples of different clusters, making it harder to spot any differences. The values of sigma (σ=2° for both b and l) have also been selected in such a way that a large part of the galaxy is typically selected in each cluster. This renders these first results scientifically useless, since they don’t identify any structure that can lead to any new knowledge, but adequate to evaluate the performance of the SG-MAC algorithm compared to the basic MAC.

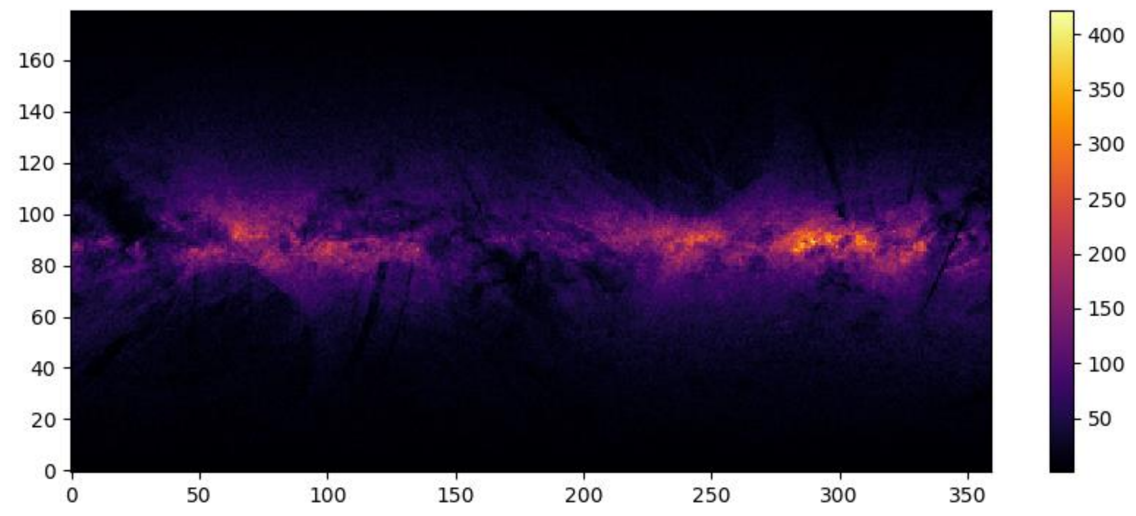


Figure 5-7: Density of the full TGAS dataset (linear scale)

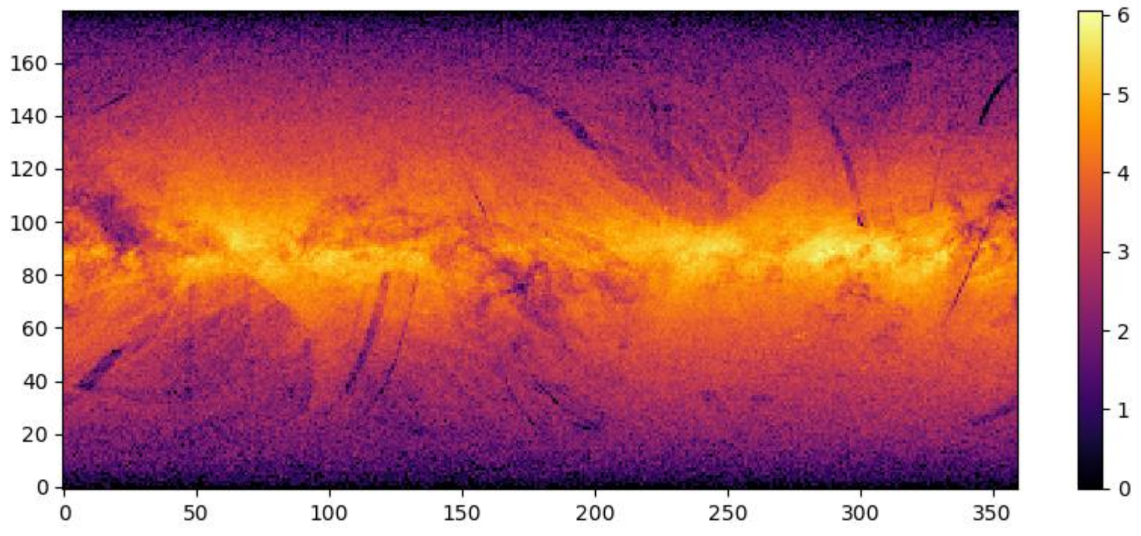


Figure 5-8: Density of the full TGAS dataset (log scale)

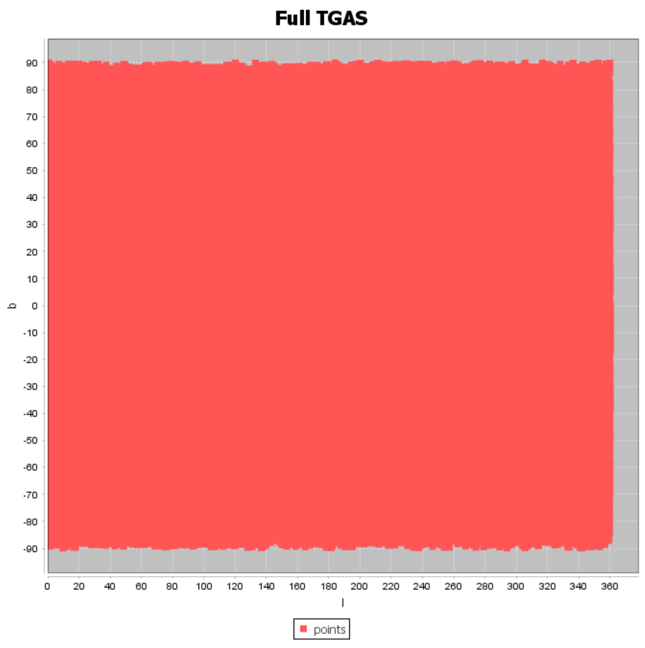


Figure 5-9: Points of the full TGAS dataset

In general, this “solid wall” look will be true for all the presentations except for the smaller datasets generated by higher levels of sample decimation. For instance, the distribution of individual instances with 10,288 uniformly selected points is presented in Figure 5-10.

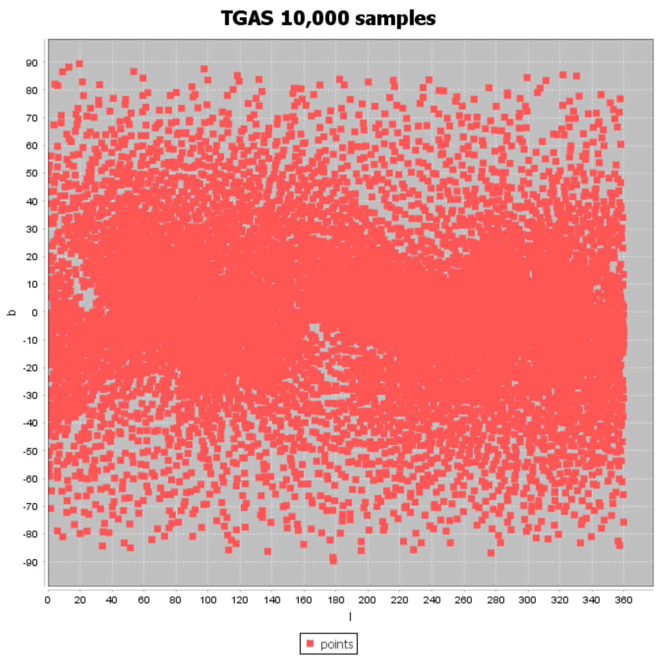


Figure 5-10: Points of the TGAS 10,000 dataset

5.2.1 Tests performed with the basic MAC algorithm

The baseline for the benchmarking of the clustering of the TGAS dataset is the result of the basic MAC algorithm over the TGAS 10000 and TGAS 20000 datasets with a value of sigma of 2 degrees in each dimension, that is displayed in Figure 5-11. MAC was also applied to the TGAS 50000 and TGAS 100000 datasets, but the run was aborted since it would have taken too long, as it will be seen in section 5.2.3.

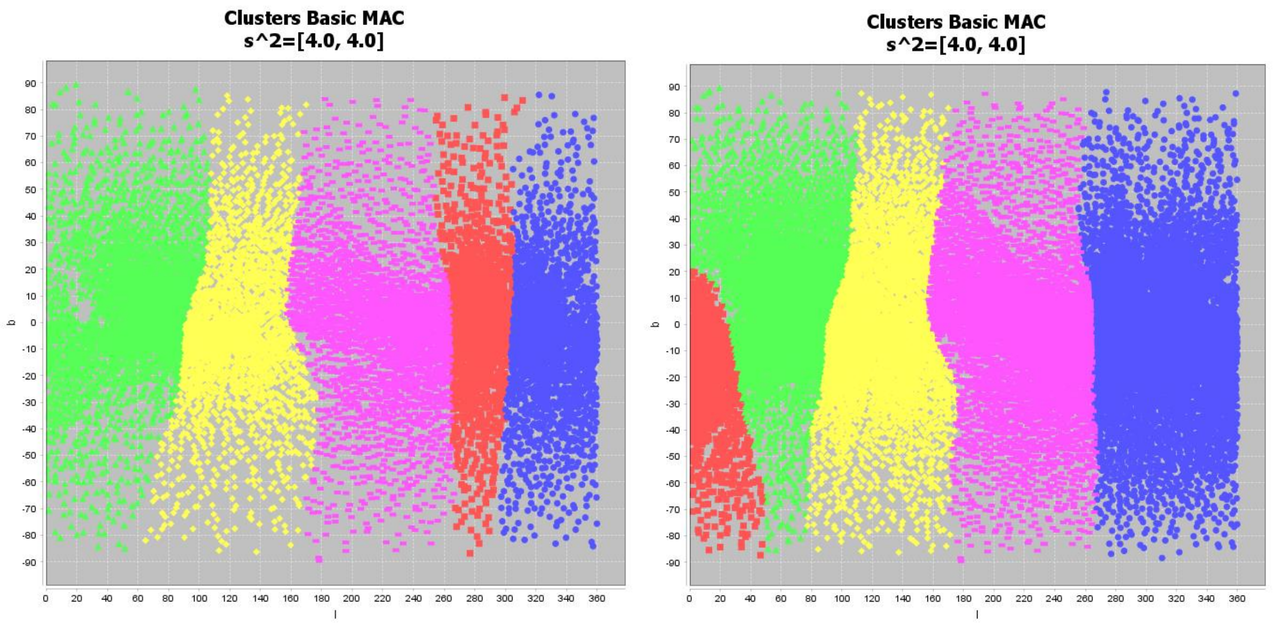


Figure 5-11:MAC clustering of TGAS 10000 (left) and TGAS 20000 (right)

The result of the clustering is a series of bands that propagate vertically. This is to be expected, since all the clusters with a high density of stars are located in the galactic plane, where they compete with each other. Away from this plane, there are no such concentrations that can create they own clusters, and so the points get assigned to the closest local maxima in the galactic plane. As mentioned, this particular clustering is not very useful, but provides a good challenge for the algorithms.

It is worth noting that although providing additional samples changes the clustering result, as it could be expected some features remain the same between them. Compared to the results with TGAS 10000, TGAS 20000 splits the leftmost cluster into two, and joins the two rightmost clusters into one. This indicates that the separation between these clusters is not as significant as the rest of the boundaries. It also serves as evidence that even with these big values of sigma, simply taking a subset of the dataset can change the clustering solution.

The next step is to perform the clustering using the dataset indexing feature. For the sake of performance, and for proper comparison with the SG-MAC results, a neighbourhood of 8 sigmas was used in the tests. This allowed performing the clustering over all four of the decimated datasets in a reasonable time. The results are presented in the Figure 5-12.

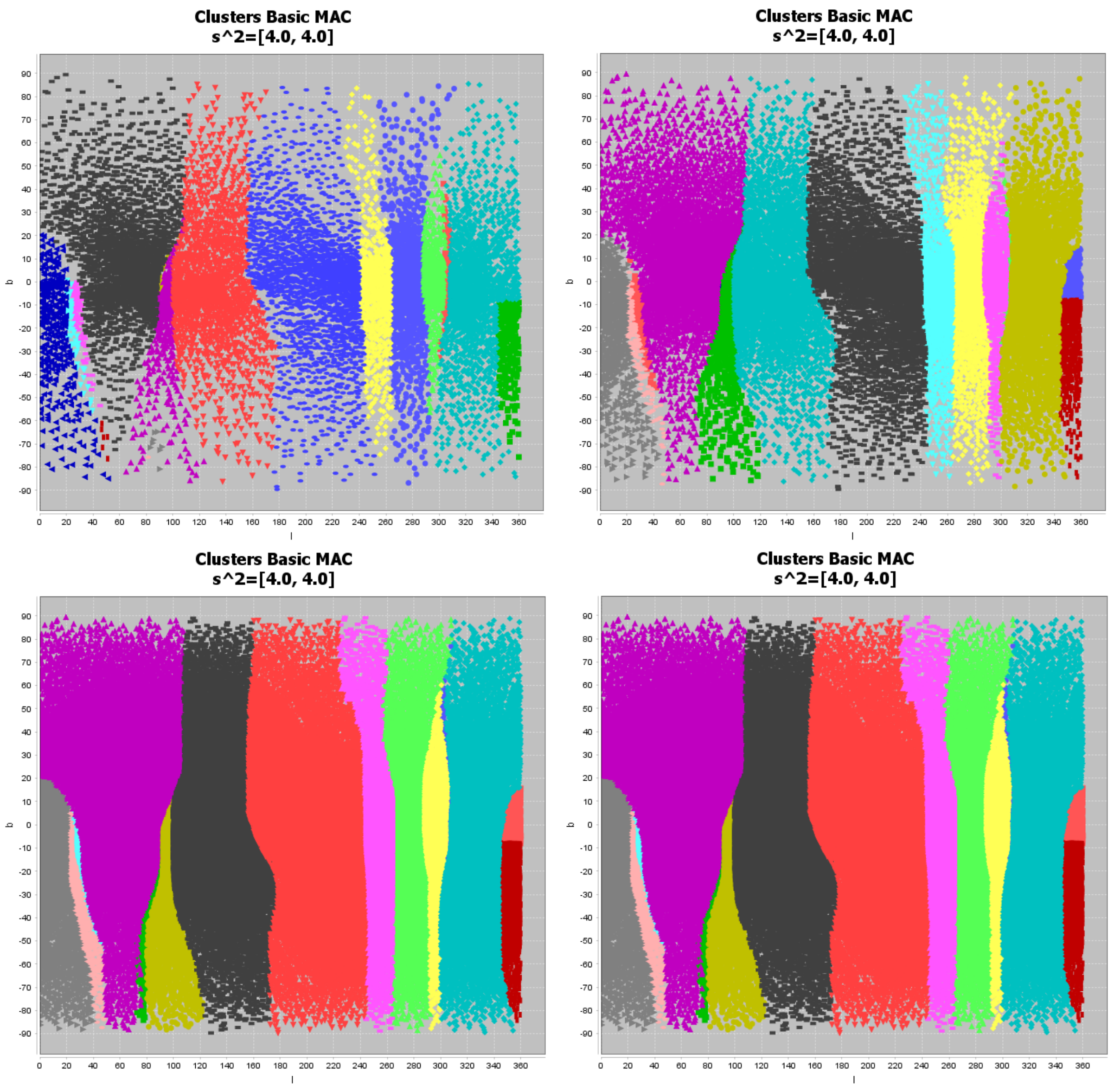


Figure 5-12: MAC clustering with indexing

TGAS 10000 (top-left), TGAS 20000 (top-right)
TGAS 50000 (bottom left) and TGAS 100000 (bottom-right)

In this case the clustering reveals additional clusters, artefacts of the reduced size of the neighbourhood (a square of 36 degrees). Nevertheless, the general shape and prominent features of the clusters generated without indexing can still be easily identified.

5.2.2 Tests with SG-MAC

After some trial an error, it is selected a minimum distance between local minima of 4 sigmas. Initially, no decimation is used at all, so all points of the dataset become samples of the PDF to be used in the clustering process itself.

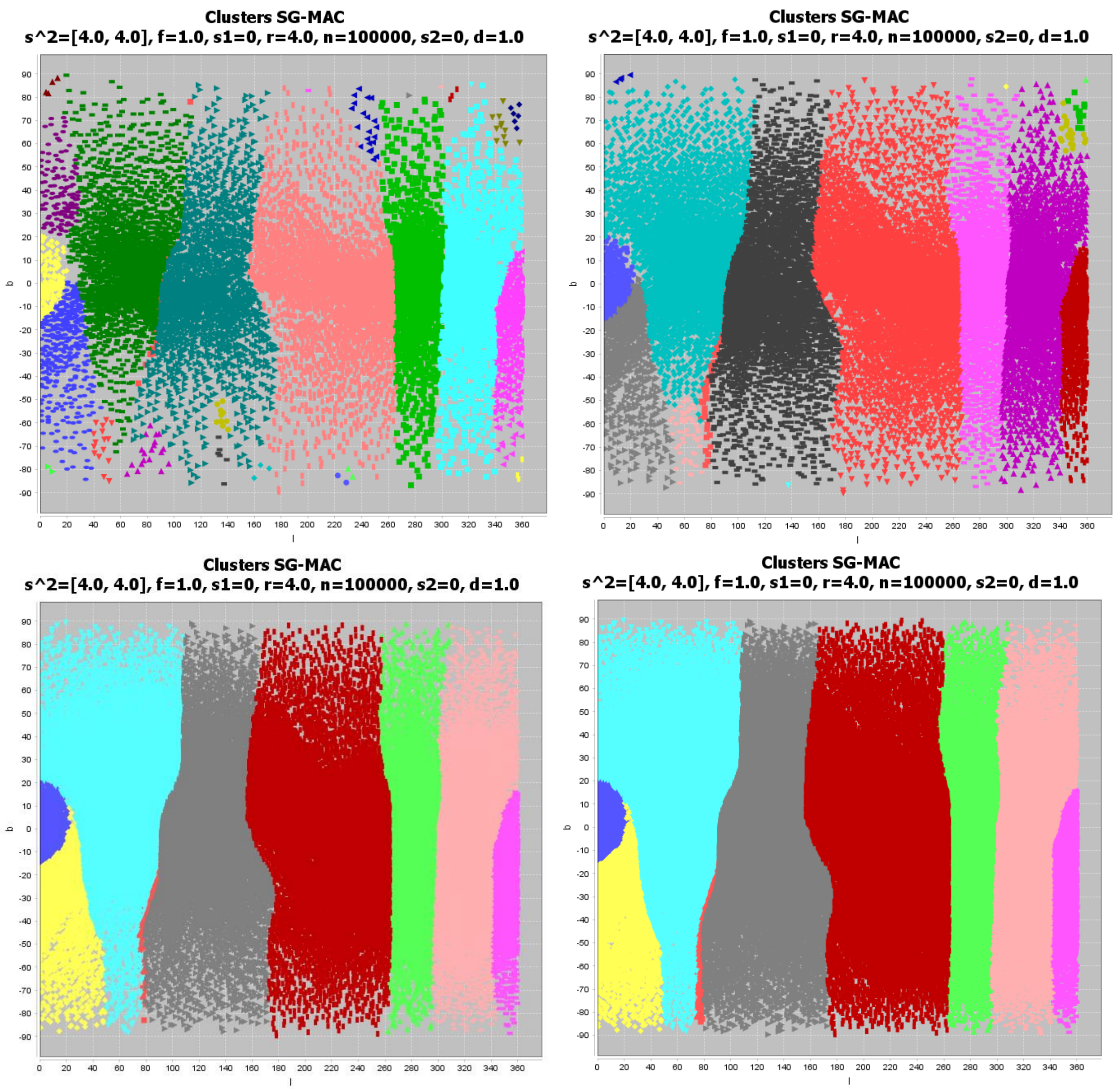


Figure 5-13: SG-MAC clustering without decimation

TGAS 10000 (top-left), TGAS 20000 (top-right)
TGAS 50000 (bottom left) and TGAS 100000 (bottom-right)

The results obtained are rather similar to those produced by the basic MAC algorithm. Actually, the extra clusters generated by the limitation of the indexing neighbourhood disappear, so that the end result is closer to the non-indexed case. With small number of points some extra clusters appear at high (positive and negative) latitudes, where the attractive effect of the galactic cluster is lower, but they do disappear as the number of available data points increases. In general, although the differences are evident, the main features of the MAC clustering are recognizable.

If decimation in stage 5 is added, so that for the clustering stage only around 1000 PDF samples are used, the results are remarkably similar to those without decimation. The difference in this case, though, is that the clustering time is quite lower, as we will see in section 5.2.3. An added benefit is that this decimation reduces the amount of information that is required to perform the clustering. This can be a considerable advantage if it has to be transferred to worker nodes in a distributed cluster, or if the dataset is big and the amount of memory is limited.

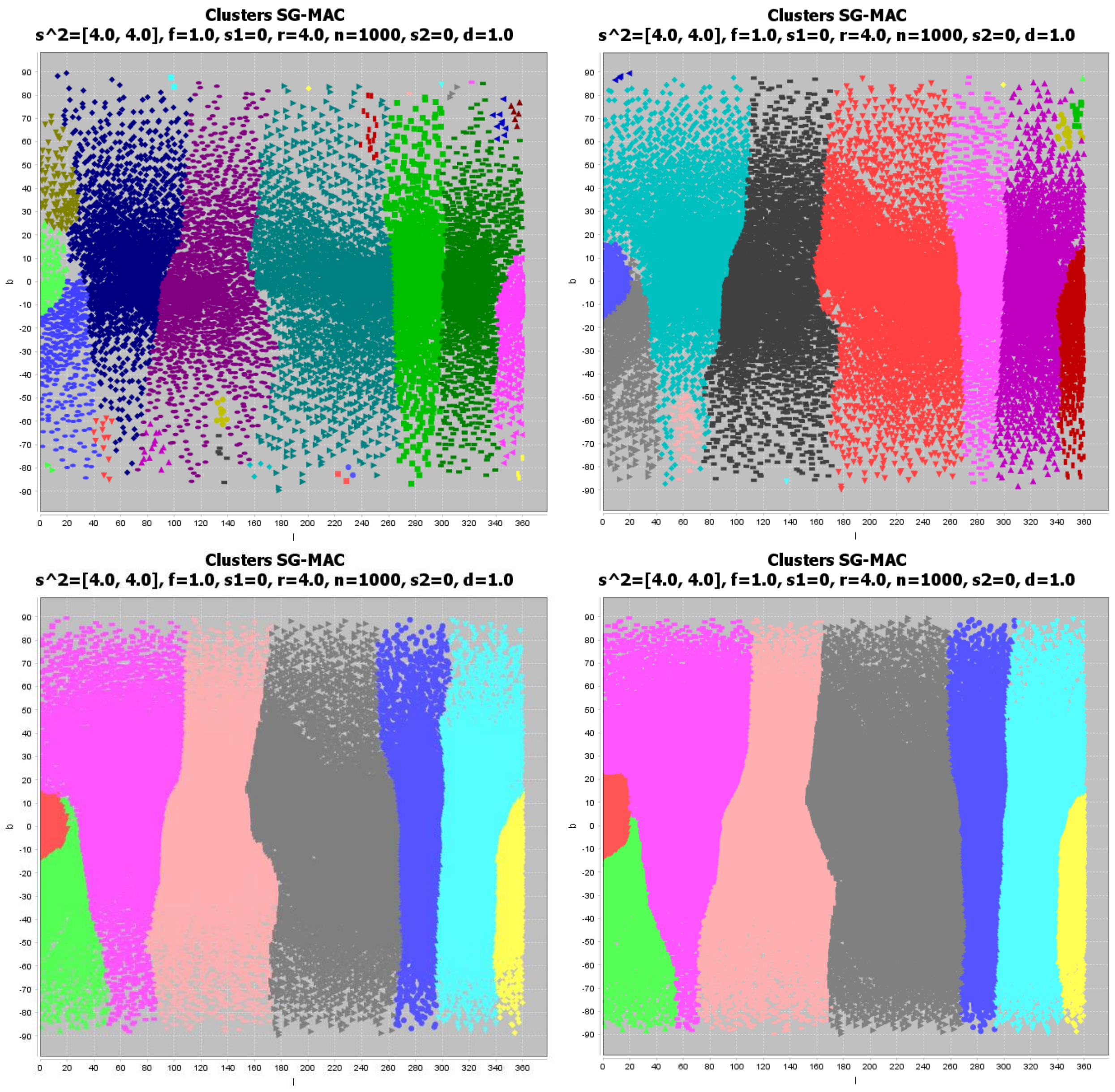


Figure 5-14: SG-MAC clustering with stage 5 decimation

TGAS 10000 (top-left), TGAS 20000 (top-right)
TGAS 50000 (bottom left) and TGAS 100000 (bottom-right)

The final feature of SG-MAC to test is the decimation of the points for which the PDF will be calculated. This pre-selection has an impact on the time of the SG-MAC initialization, but not on the clustering time itself.

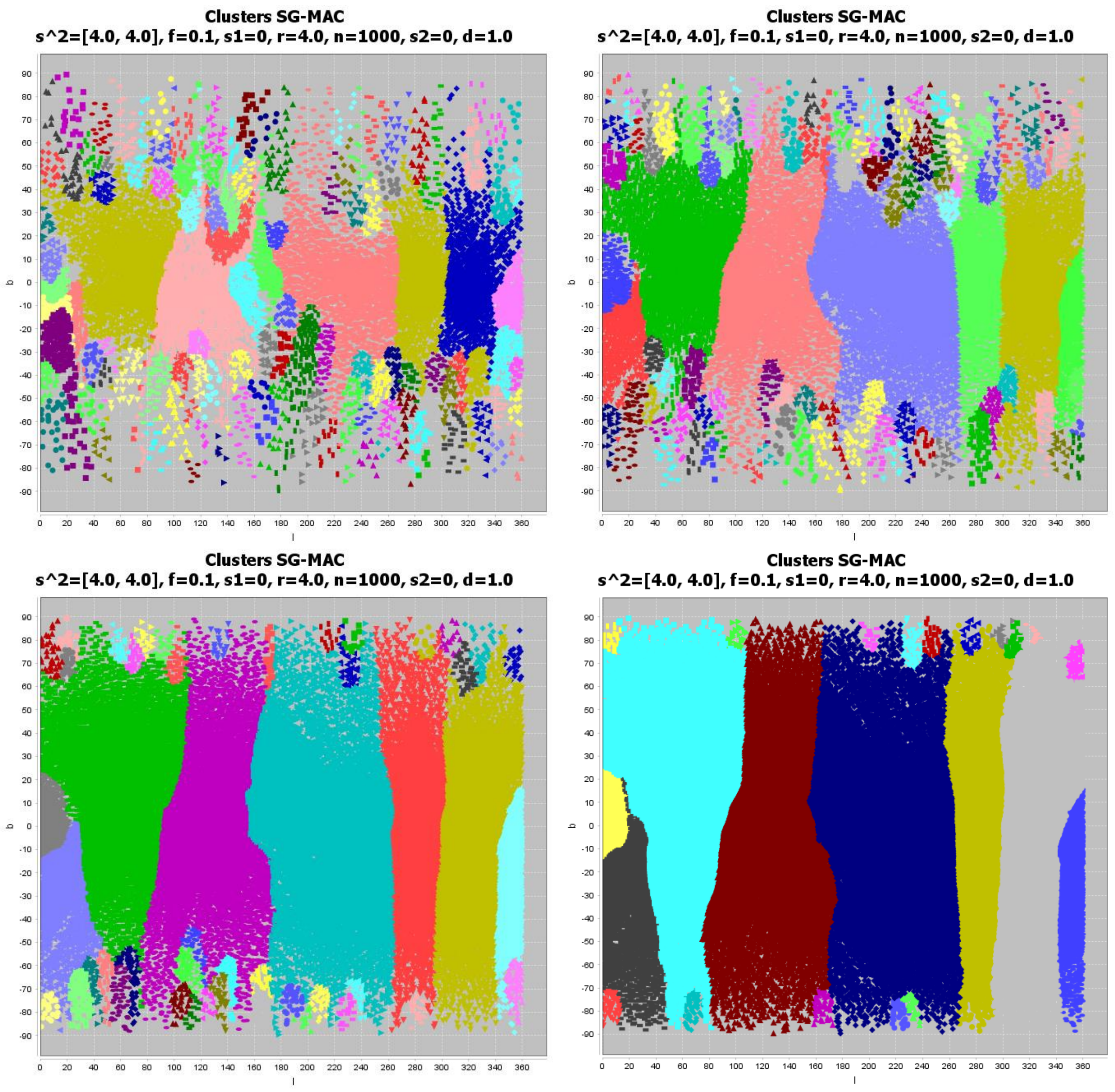


Figure 5-15: SG-MAC clustering with pre-selection of 10% of the points for sampling and stage 5 decimation

TGAS 10000 (top-left), TGAS 20000 (top-right)
TGAS 50000 (bottom left) and TGAS 100000 (bottom-right)

The reduction of the points on which the PDF sample is calculated has a quite notable and negative effect in general. Fewer sample points imply a larger distance between them, and more possibilities for a point to become a local maximum, and therefore a cluster mode on small differences of the PDF value, in particular at high latitudes where the density of points is much lower. Nevertheless the computation time is reduced, and with enough samples to start with the impact is limited.

Actually, the computation time improves so much that in this case it is possible to run SG-MAC over the full TGAS dataset, as presented in Figure 5-16.

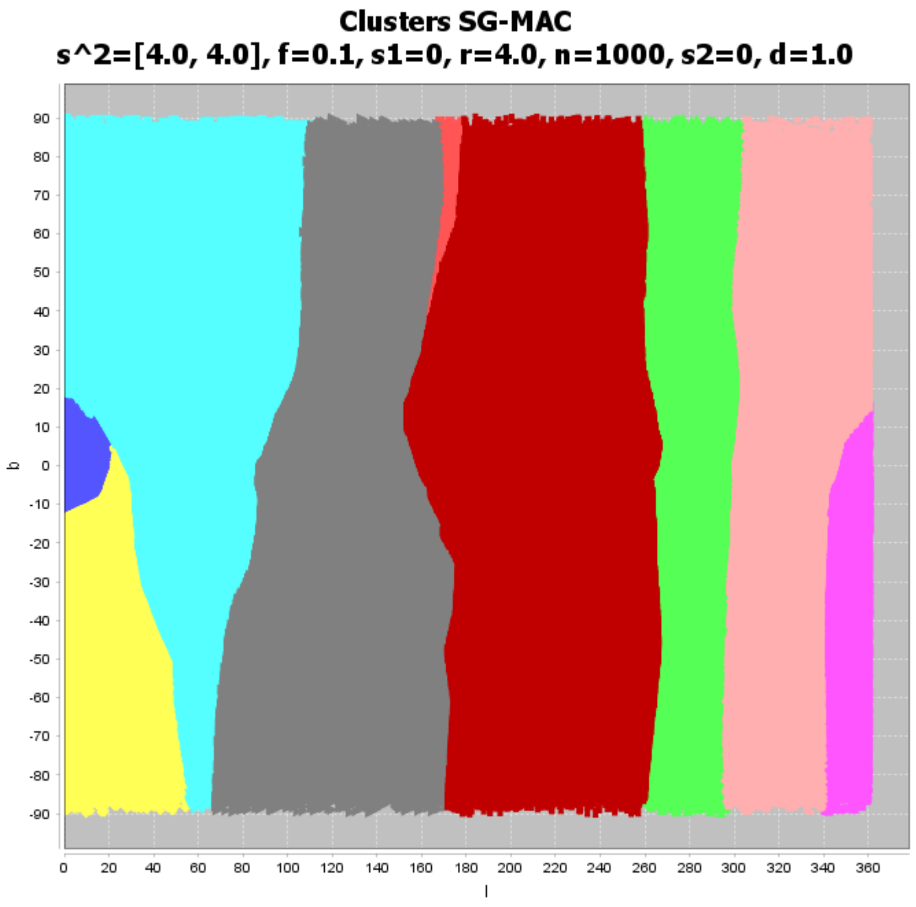


Figure 5-16: SG-MAC clustering over the full TGAS dataset
with pre-selection of 10% of the points for sampling and stage 5 decimation

In this case a small fraction of the points does not converge to any cluster, but the shape of the clusters is perfectly recognizable and does not present any negative impact from the pre-selection decimation.

5.2.3 Assessment of the processing times

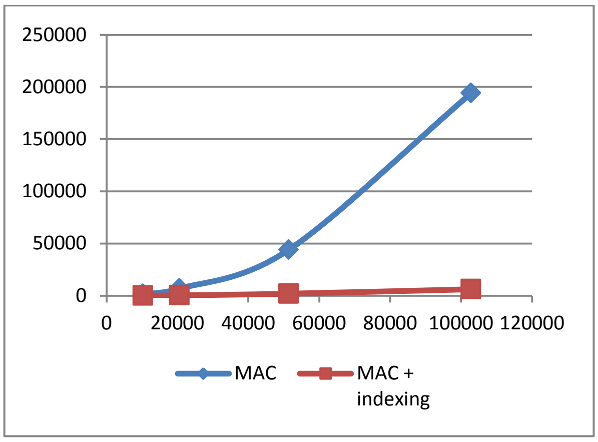
All the tests were run in a laptop with an Intel core i5 processor at 2.3 GHz with 8 GB of Ram and running Java 1.8.0_45 over Windows 7 64-bit.

The summary of the execution times is presented in the Table 5-2 and the Figure 5-17.

# points in dataset	MAC	MAC + Indexing	SG-MAC			SG-MAC + decimation			SG-MAC + pre-selection + decimation		
	total	total	Init.	clustering	total	Init.	clustering	total	Init.	clustering	total
10,288	1466	80.1	0.9	2.8	3.7	0.8	0.8	1.6	0.3	0.67	1.0
20,574	6742	299.0	2.2	9.8	12.0	1.9	1.7	3.6	0.4	1.3	1.7
51,432	44043*	1894.0	9.5	48.8	58.3	10.3	4.1	14.4	1.2	4.1	5.3
102,865	194125*	6130.5	44.7	123.4	168.1	37.7	8.2	45.9	3.7	7.4	11.1
2,057,050	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1249.0	127.2	1376.2

Table 5-2: Summary of computation times

* Time estimated from linear interpolation of the calculation time of the first 5000 points



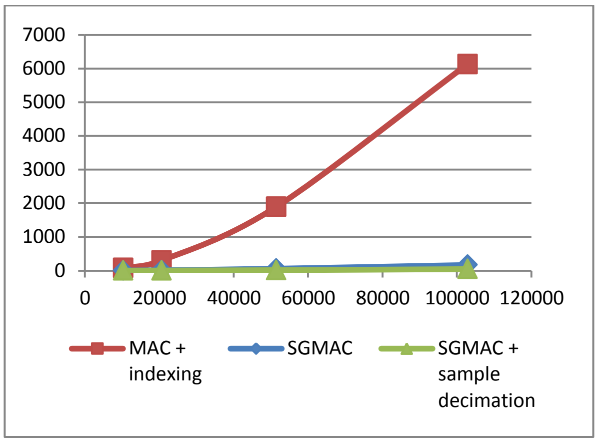


Figure 5-17: Comparison of computation times

It is evident that the use of SG-MAC reduces notably the regular MAC computation time, up to 4 full orders of magnitude for a 100,000 points dataset.

5.2.4 Assessment of the equivalent computational complexity

Let's assume that the computation time for a given program has polynomial complexity ($O(n^k)$). If so, the logarithm of the total computation time would be:

$$log_{10}(t_T) = log_{10}(c\, n^k) = k\, log_{10}(n) + \, log_{10}(c)$$

That means that if the logarithm of the total computation time is plotted against the logarithm of the number of inputs, for a polynomial-complexity program the result should be a straight line with a slope that corresponds to the equivalent order of the polynomial.

Doing such calculations for the obtained computation times of MAC and SG-MAC, the curves summarized in Figure 5-18 and Table 5-3 are obtained.

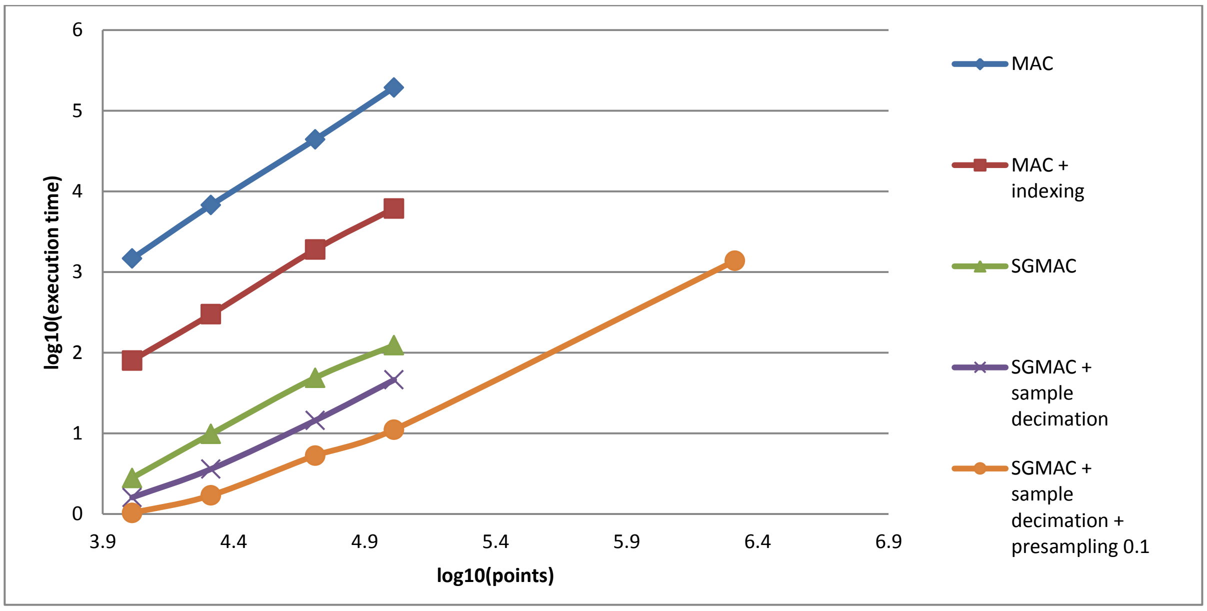


Figure 5-18: log10(time) vs log10(points)

Algorithm	Estimated slope
MAC	2.122
MAC + indexing	1.884
SG-MAC	1.657
SG-MAC + sample decimation	1.457
SG-MAC + pre-selection + sample decimation	1.454

Table 5-3: Estimated equivalent polynomial complexity of each algorithm

If the tendency holds for even bigger datasets, the prediction for the clustering of a dataset of 1 billion elements for SG-MAC with pre/selection and decimation is 6,717,762 seconds, which corresponds to roughly 78 days. Considering that this assumes a single-thread environment, if the work can be distributed between different machines in a cluster, the total computation time can be reduced to a few days, which can be acceptable in many applications.

5.3 Realistic multidimensional TGAS clustering

For these tests two different datasets will be used. The first one, TGAS_pleiades, is composed of the first 50,000 stars available in the first TGAS file. By pure chance that subset of TGAS contains the area of the sky where the well known Pleiades open cluster is located.

One of the first goals of the test would be to try to isolate the members of this star cluster in one of the groups generated by SG-MAC, first by simple inspection, and then comparing the list of sources with a listing from Gaia data generated using a different method and available in [21]. The parameters of the clustering will be selected manually to get a reasonable match.

Star clusters are groups of stars that are physically close to each other and tend to move in the same direction. These characteristics can help isolate them from other stars that are close in the line of sight but either closer of farther away, or that are moving in a different direction.

For this reason, the parameters to be used for this clustering are:

1. `source_id`: Identification column, not used in the clustering itself, but useful in the comparison with external listings.
2. `ra`: Right ascension of the star, one of the polar coordinates that specify its position in the sky.
3. `dec`: Declination, the second polar coordinate of the star.
4. `parallax`: Apparent movement of the star when observed from opposite extremes of the Earth orbit. It is roughly the inverse of the distance from the Sun to the star in parsecs.
5. `pmra`: Proper motion of the star in the direction of the right ascension
6. `pmdec`: Proper motion of the star in the direction of the declination.

It would be desirable to have also the radial velocity, the equivalent of `pmra` and `pmrec` in the radial direction, but that information is not available in TGAS. Also, it should be noted that the parallax is not a measure of a physical position, but of its inverse. The difference of parallaxes between two stars that are close to each other and to the Earth can be quite higher than that of two stars far away between them but also from Earth. As such, clustering using this item can lead to counterintuitive results. Nevertheless, transforming the parallax into distance this causes its own problems and uncertainties due to the different distribution of the errors in both cases, and so it has been chosen to use the data without further processing.

Then the same parameters will be applied to the full TGAS dataset. The assumption to be tested is whether these settings would allow selecting into the same group the objects that belong to the same star cluster, provided the star cluster has characteristics similar enough to those of the Pleiades open cluster.

In order to check this assumption, stars corresponding to the Coma Berenices cluster will be checked in a similar way as the Pleiades stars, since a list of members is also provided in [21].

5.3.1 Tests with TGAS_pleiades

The values of sigma handpicked to be used for each variable are those listed in Table 5-4:

Variable	Sigma
`ra`	0.3
`dec`	0.3
`parallax`	0.2
`pmra`	5
`pmdec`	5

Table 5-4: Values of sigma for each variable

With these values, a minimum distance between local maxima of 7 and without any pre-selection nor decimation of the samples, the clustering with SG-MAC generates 3212 clusters, of which only 54 have more than 10 members. 48,729 points out of 50,000 converged to a solution, which represents an 97.46% convergence rate. The initialization was completed in 9.4 seconds and the clustering stage in 91 seconds.

Figure 5-19 contains a representation of the samples of the PDF in all the points of the dataset. The size of the blobs corresponds to the value of the PDF at that point. Identified local maxima are presented in blue. Since this is a 2D projection of a 5-dimensional dataset over `ra` and `dec` , being close to a local maximum in the figure does not imply that a point would be assigned to it, since the distance can be big in some of the non-visible dimensions.

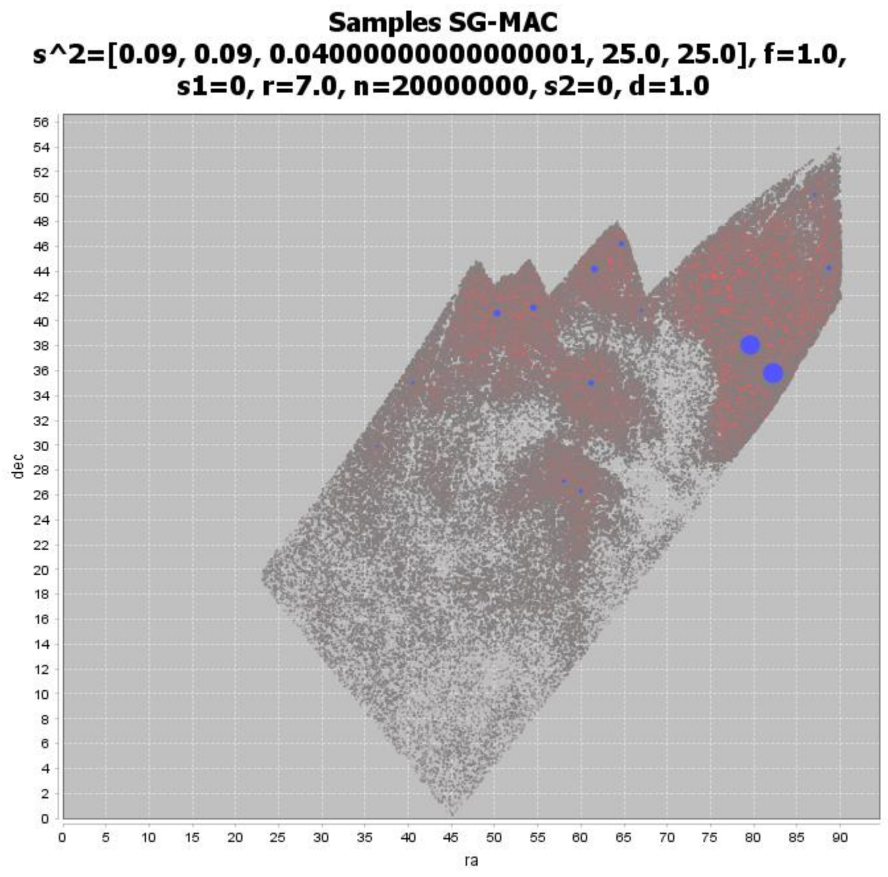


Figure 5-19: PDF value samples

Despite the large values of the PDF in the upper part of the diagram, our area of interest is in the centre of the region covered by this dataset. A detail of this area is presented in Figure 5-20:

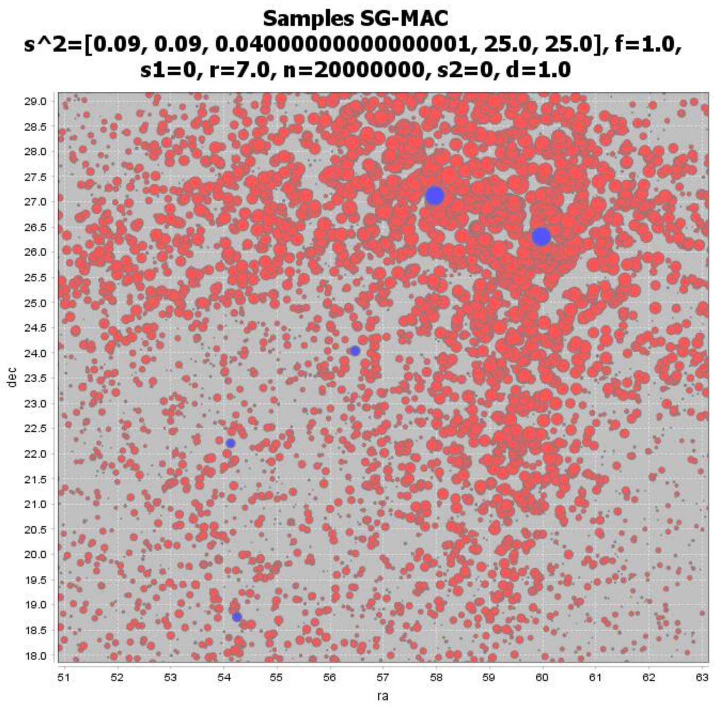


Figure 5-20: Detail of the PDF sampled values

As we will see, the Pleiades cluster is represented by the local maximum at `ra`=56.5, `dec`=24.0 in the centre of Figure 5-20. Figure 5-21 presents the assignment of each individual sample to one of the clusters. The one corresponding to the Pleiades appears at the centre of the image in purple.

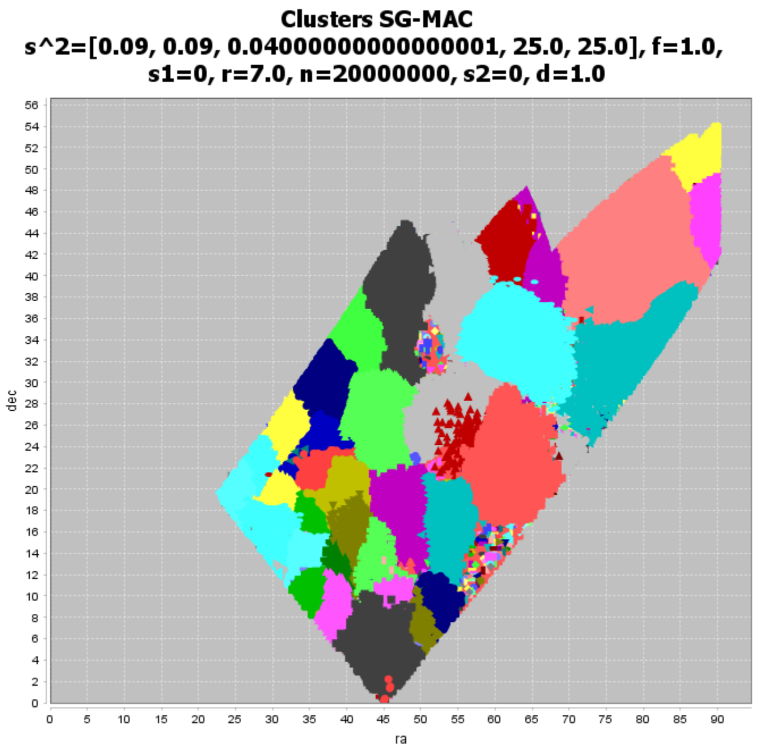


Figure 5-21: Clusters found in the TGAS_pleiades dataset

Figure 5-22 presents a detailed view of the central area of the region covered by this dataset, plus a separate diagram with the isolated samples of the cluster containing a star previously identified as belonging to the Pleiades

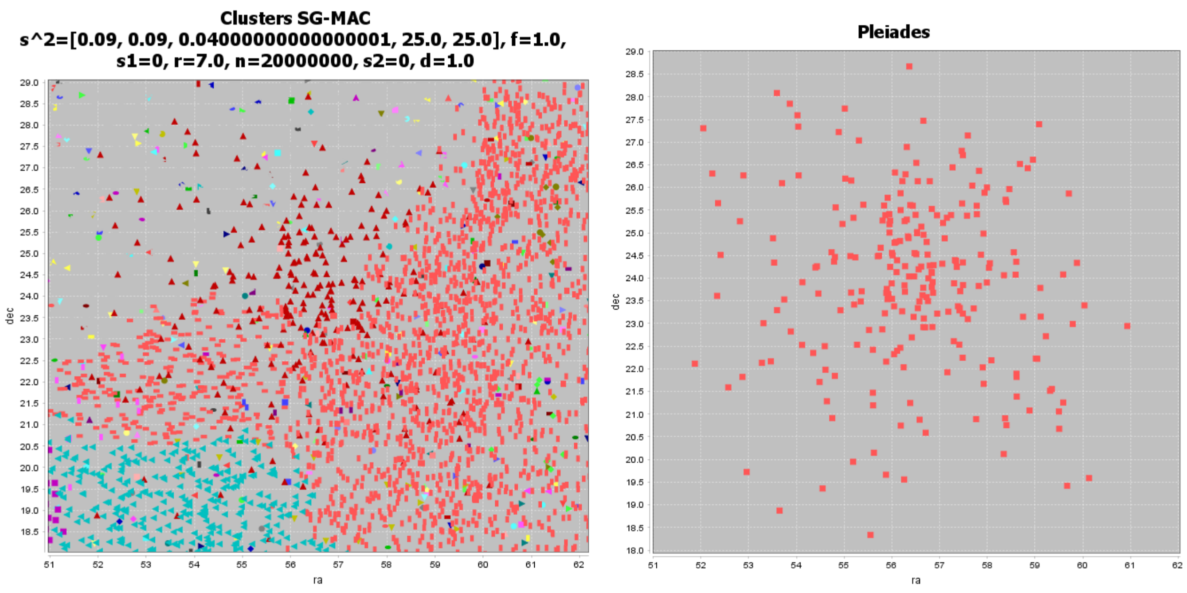


Figure 5-22: Detail of the clustering result and the candidate Pleiades cluster

As can be noticed, points of this cluster are interleaved with points of others, the difference being the distance in the non-shown dimensions.

As mentioned, [21] contains a listing of probable members of the Pleiades cluster. It is available online in the following URL:

https://www.aanda.org/articles/aa/full_html/2017/05/aa30552-17/T11.html

The summary of the comparison of the `source_id`s in that table with the members of the selected cluster the results is presented in Table 5-5.

	Reference	SG-MAC output
Total number of objects	154	245
Objects in common	150	150
% objects in common	97%	61%

Table 5-5: Comparison of objects in selected cluster and in the reference Pleiades list

The output of the SG-MAC is larger in number of elements, but has managed to capture the vast majority of the stars considered as probable members of the cluster.

The same experiment was repeated, this time using decimation to reduce the number of PDF samples used in the clustering stage to just 20% of the original dataset (around 20,000 sampled points).

In this case the convergence rate was slightly reduced to 96%, while the computation time of the clustering step grew to 135 seconds. From the generated statistics, the average path to convergence increased from 31 to 56 steps, making the total path computation costlier although the number of searched points decreased.

Further tests with different decimation values confirmed this result. The average iterations until convergence and the computation time increased when the samples decreased for these values of sigma, reversing the effects noticed when processing the same dataset with a value of sigma almost 7 times larger.

A possible explanation for that with a small value of sigma, the PDF has sharper peaks and valleys, and the decimated samples no longer reflect it properly as they would do with longer, softer slopes. In terms of signal processing, it would correspond to higher spatial frequencies, which according to the Nyquist criterion would require also a shorter sampling period if the sampling was regular.

This point requires further study to identify in which cases it is statistically safe to use decimation in the samples without obtaining the opposite effect of what was intended on the processing time.

In any case, some of the members of the cluster did change, but the concordance with the reference list was maintained high:

	Reference	SG-MAC output (with sample decimation)
Total number of objects	154	235
Objects in common	149	149
% objects in common	96%	63%

Table 5-6: Comparison of objects in selected cluster with decimation and in the reference Pleiades list

The concordance of the elements of both calculated clusters, with and without decimation is, as it could be expected, very high:

	SG-MAC output	SG-MAC output (with sample decimation)
Total number of objects	245	235
Objects in common	224	224
% objects in common	91%	95%

Table 5-7: Concordance between clusters calculated with and without decimation

5.3.2 Tests with the complete TGAS dataset

With the same parameters used in the previous tests, the clustering is repeated, but this time over the whole TGAS dataset instead of just a fraction of it. The process took 958 seconds to run the initialization (16 minutes), and 32410 seconds (9 hours) to cluster all the data points without PDF sample decimation. The solution found 109,899 clusters out of the 2,057,050 samples. Nevertheless many of them represent some kind of isolated outlier. Only the 8% of these clusters (8,779) contain more than 2 members and just the 1.9% (2,105) contain more than 10 members.

5.3.2.1 Pleiades cluster

The first step with this dataset is to return to the Pleiades cluster and check that the additional data points have not changed the previous solution. Representing the points associated to the clusters whose mode is at ±10 degrees of the mode of the cluster that was found with the TGAS_pleiades dataset, it is confirmed that the same cluster has been obtained again (in this run, represented by green ovals). Actually, by comparing the `source_id` it can be confirmed that the list of points is exactly the same.

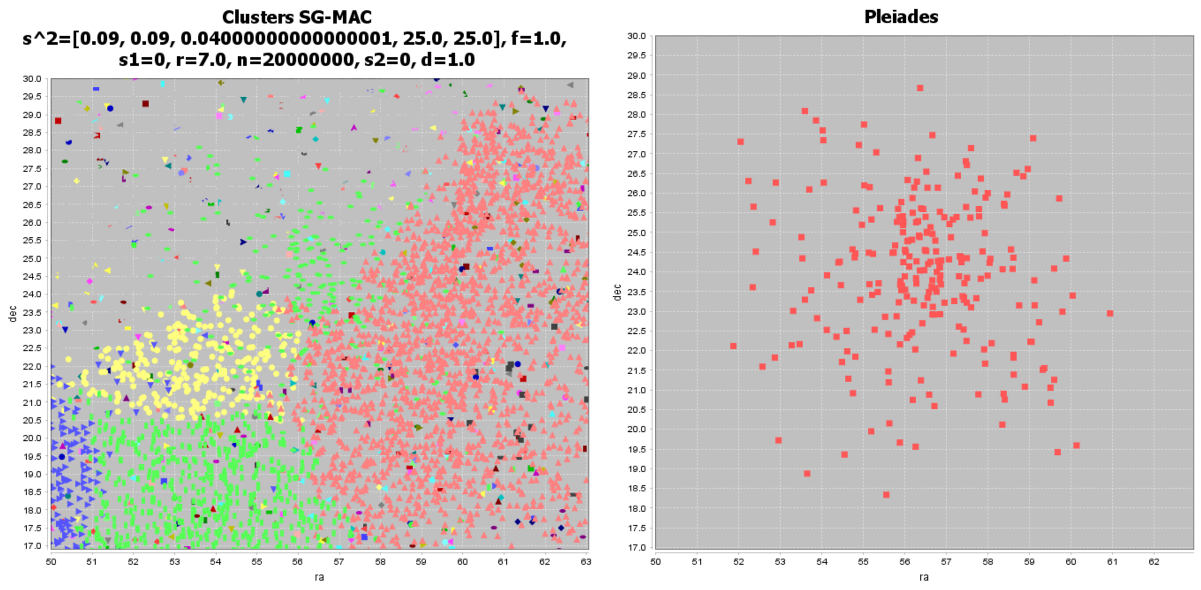
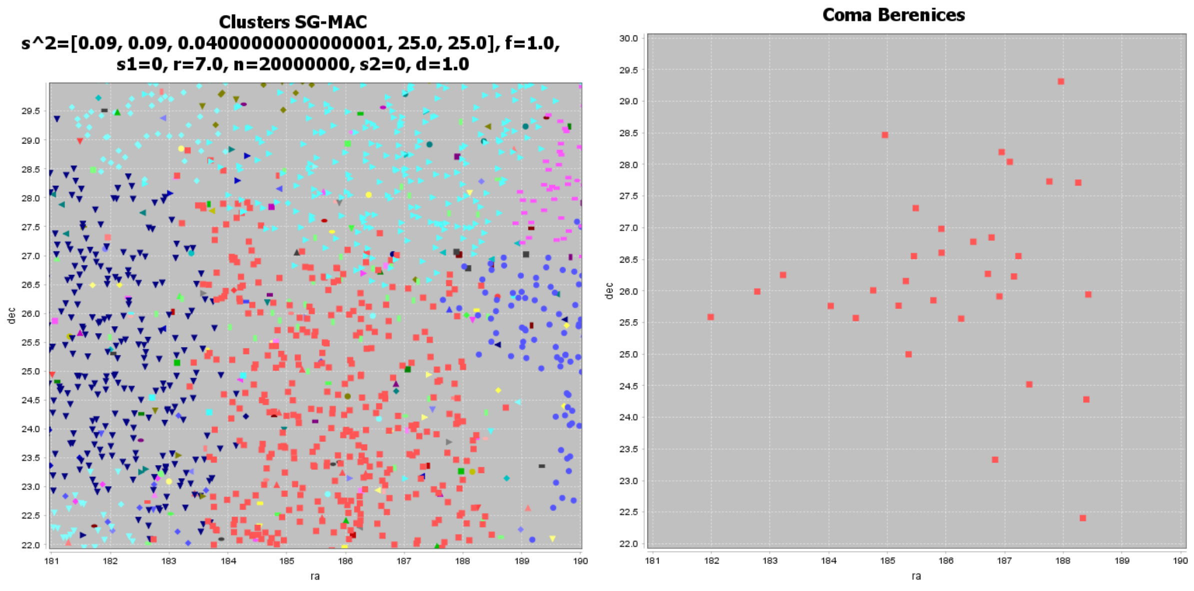


Figure 5-23: Clusters in the area of Pleiades and selected cluster points

5.3.2.2 Coma Berenices cluster

The second step is to inspect the region of the cluster Coma Berenices (ra=186.02°, dec=25.95°). As a candidate cluster we can select the cluster represented in light green squares that can be appreciated from the 2D projection of the generated clusters in Figure 5-24:


Clusters SG-MAC

s^2=[0.09, 0.09, 0.04000000000000001, 25.0, 25.0], f=1.0, s1=0, r=7.0, n=20000000, s2=0, d=1.0

Figure 5-24: Clusters in the area of Coma Berenices, and candidate cluster

By extracting the `source_id`s of these points and comparing with the reference list provided in [21] and available online in the URL:

https://www.aanda.org/articles/aa/full_html/2017/05/aa30552-17/T10.html

The result is that the algorithm produced an accurate, although incomplete, list of the Coma Berenices stars, as can be seen in Table 5-8.

	Selected SG-MAC clusters	Reference
Total number of objects	35	49
Objects in common	34	34
% objects in common	97%	69%

Table 5-8: Concordance between selected clusters and the reference Coma Berenices list

In general, there is no such thing as an optimum value of sigma for this particular task. Increasing or decreasing the value in one dimension might improve the detection of some star clusters but impact negatively on the grouping of others, be it by increasing the number of stars associated to it or by splitting the cluster into smaller units. Hierarchical grouping over the clusters generated by SG-MAC, e.g. applying the full HMAC algorithm might help, but in that case identifiable clusters would be distributed at different levels of the clustering solution.

In any case, with these tests SG-MAC has demonstrated being valuable as a tool to assist astronomers identify open clusters of stars, using large, high accuracy, multidimensional datasets.

6 Future work

This work has focused on the comparison between MAC and the modified SG-MAC in terms of performance and robustness for a single case. Additional testing would be required to confirm the robustness of SG-MAC over a wide range of applications, or characterizing in which conditions it is not advisable, e.g. whether there are a minimum number of input samples to apply pre-selection.

As was seen during the tests, decimation of the samples can both improve and impact negatively the computation time, depending on the dataset and the parameters of the algorithm. Further investigation on this relationship is required, although it seems possible to find an optimum empirically by running the clustering in a reduced but representative subset of the full dataset with the same density of data points, and checking whether decimation improves the computation time versus the case with all the samples.

Similarly, several design decisions had alternatives that have not been fully explored. Different algorithms for convergence or gradient estimation were tried, but other alternatives can be explored and compared for robustness and performance with the current solution. Further work is required to study the causes of non-convergence and formulate improved criteria to reduce them.

MAC (and SG-MAC) represents only the first level in the processing of an hierarchical clustering with HMAC. If SG-MAC is indeed an equivalent substitute for MAC, the integration into HMAC should be trivial, but this multilevel approach has not been tested.

One principal line of future work is to parallelize and distribute the algorithm implementation. Even though this has been a design goal of the modifications from the very beginning, distributed environments and big datasets always present challenges, and quite some more effort will be required, probably tailored for a specific environment and infrastructure rather than as a general algorithm. An exciting line of work would be to integrate the SG-MAC code as a library in some of the popular frameworks for processing of big datasets and machine learning, such as Apache Spark.

7 Conclusion

In this work, modifications to the Modal Association Clustering algorithm have been presented with the goal of improving significantly its performance. A new algorithm, SG-MAC, has been tested against several different datasets to check the similarity of its results to those of the original MAC version. One of them, TGAS, is an actual scientific dataset representative for one of the potential applications of a clustering algorithm.

During the tests, the increase in performance of up to 3 or 4 orders of magnitude for 100.000 data points over a bi-dimensional dataset has been confirmed, while maintaining an acceptable degree of similarity with the solutions provided by MAC. Even more, it has been estimated that the total computation times grow with the number of data points at a slower rate than they would do with a $O(N^2)$ complexity, meaning that scalability has also been improved at least in some cases.

Part of the improvement is achieved by the introduction of indexing of points close to each other, so that a local neighbourhood of points can be obtained efficiently rather than scanning the whole dataset. Another important part comes from simplifications in the algorithm (e.g. finding closest sample vs computing Gaussians) and a convergence criterion that cuts down the last steps of the path since the local maxima are known in advance.

Nevertheless, the computation times are strongly dependent on the parameters of the clustering, in particular the standard deviation of the Gaussian kernels used to estimate the PDF matching the observed distribution of data points. For relatively high values of sigma related to the distribution of data points, decimation of the PDF gradient samples improve significantly performance. For small values, decimation presents a negative effect. In any case, it helps reducing the memory requirements to make the PDF estimation fit into the available memory of each computing node.

SG-MAC has also been tested on a realistic setting and demonstrated its value identifying star clusters based on their positional and movement characteristics.

Therefore SG-MAC has been shown as affective clustering technique against large, complex datasets, improving the scalability of previous algorithms for modal clustering and making feasible not only the use of this technique over very large datasets, but also the dramatic increase of performance in other existing systems.

8 Acknowledgements

This work has made use of data from the European Space Agency (ESA) mission Gaia (https://www.cosmos.esa.int/gaia), processed by the Gaia Data Processing and Analysis Consortium (DPAC, https://www.cosmos.esa.int/web/gaia/dpac/consortium). Funding for the DPAC has been provided by national institutions, in particular the institutions participating in the Gaia Multilateral Agreement.

9 Bibliography

1. Li, Jia, Surajit Ray, and Bruce G. Lindsay. "A nonparametric statistical approach to clustering via mode identification." *Journal of Machine Learning Research* 8.Aug (2007): 1687-1723.
2. Russell, Stuart, Peter Norvig, and Artificial Intelligence. "A modern approach." *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25 (1995): 817
3. Gower, John C., and Gavin JS Ross. "Minimum spanning trees and single linkage cluster analysis." *Applied statistics* (1969): 54-64.
4. Hernández Orallo, José, Ramírez Quintana Ma. José, Ferri, Ramírez César, Introducción a la minería de datos, Pearson Educación, SA Madrid (2004): 432-436
5. Prusti, Timo, et al. "The Gaia mission." *Astronomy & Astrophysics* 595 (2016): A1.
6. Brown, Anthony GA, et al. "Gaia Data Release 1-Summary of the astrometric, photometric, and survey properties." *Astronomy & Astrophysics*595 (2016): A2.
7. Astraatmadja, Tri L., and Coryn AL Bailer-Jones. "Estimating distances from parallaxes. III. Distances of two million stars in the Gaia DR1 catalogue." *The Astrophysical Journal* 833.1 (2016): 119.
8. Rokach, Lior, and Oded Maimon. "Clustering methods." *Data mining and knowledge discovery handbook*. Springer US, 2005. 321-352.
9. Murtagh, Fionn. "A survey of recent advances in hierarchical clustering algorithms." *The Computer Journal* 26.4 (1983): 354-359.
10. Everitt, Brian S., et al. "Hierarchical clustering." *Cluster Analysis, 5th Edition*(2011): 71-110.
11. Everitt, Brian S., et al. "Finite Mixture Densities as Models for Cluster Analysis." *Cluster Analysis, 5th Edition*: 143-186.
12. Banfield, Jeffrey D., and Adrian E. Raftery. "Model-based Gaussian and non-Gaussian clustering." *Biometrics* (1993): 803-821.
13. Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society. Series B (methodological)* (1977): 1-38.
14. Kriegel, Hans-Peter, et al. "Density based subspace clustering over dynamic data." *Scientific and Statistical Database Management*. Springer Berlin/Heidelberg, 2011.
15. Utrilla, Enrique, et al. "Massive Astrometric Daily Data Processing: The Gaia SOC Processing Pipeline." *Proceedings of the 2014 conference on Big Data from Space (BiDS'14)* (2014): 271-274
16. Jenssen, Robert, et al. "Optimizing the Cauchy-Schwarz PDF distance for information theoretic, non-parametric clustering." *EMMCVPR*. 2005: 34-45
17. Parzen, Emanuel. "On estimation of a probability density function and mode." *The annals of mathematical statistics* 33.3 (1962): 1065-1076.
18. Shalchyan, Vahid, and Dario Farina. "A non-parametric Bayesian approach for clustering and tracking non-stationarities of neural spikes." *Journal of neuroscience methods* 223 (2014): 85-91.
19. Görür, Dilan, and Carl Edward Rasmussen. "Dirichlet process gaussian mixture models: Choice of the base distribution." *Journal of Computer Science and Technology* 25.4 (2010): 653-664.
20. Sudderth, Erik Blaine. *Graphical models for visual object recognition and tracking*. Diss. Massachusetts Institute of Technology, 2006: 105-112
21. Collaboration, Gaia, et al. "Gaia Data Release 1. Open cluster astrometry: performance, limitations, and future prospects." (2017).
22. Gorski, Krzysztof M., et al. "HEALPix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere." *The Astrophysical Journal* 622.2 (2005): 759.

23. Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." *Kdd*. Vol. 96. No. 34. 1996.
24. Ankerst, Mihael, et al. "OPTICS: ordering points to identify the clustering structure." *ACM Sigmod record*. Vol. 28. No. 2. ACM, 1999.
25. Jinyin, Chen, et al. "Fast Density Clustering Algorithm for Numerical Data and Categorical Data." *Mathematical Problems in Engineering* 2017 (2017).
26. Kriegel, Hans-Peter, Peer Kröger, and Arthur Zimek. "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.1 (2009): 1.
27. Guttman, Antonin. *R-trees: A dynamic index structure for spatial searching*. Vol. 14. No. 2. ACM, 1984.