# GRID distribution supports clustering validation of large mixed microarray data sets



**Angelica Tulipano[1], Carmela Marangi[4], Leonardo Angelini[3], Giacinto Donvito[2], Guido Cuscela[2], Giorgio Pietro Maggi[2,3], Andreas Gisel[1§]**

[1]CNR, Istituto Tecnologie Biomediche Sezione di Bari, Bari, Italy
[2]INFN Sezione di Bari, Bari (Italy),
[3]Dipartimento Interateneo di Fisica, Università degli Studi e Politecnico di Bari, Bari, Italy
[4]CNR, Istituto per le Applicazioni del Calcolo Sezione di Bari, Bari, Italy
[§]Corresponding author

*Depicted authors have names underlined.*

## Abstract

Microarray data are a rich source of information, containing the collected expression values of thousands of genes for well-defined states of a cell or tissue. Vast amounts of data (thousands of arrays) are publicly available and ready for analysis, for example to scrutinise correlations between genes at the level of gene expression. The large variety of arrays available makes it possible to combine different independent experiments to extract new knowledge. Starting with a large set of data, relevant information can be isolated for further analysis. To extract the required information from data-sets of such size and complexity requires an appropriate and powerful analysis method. In this study, we chose to use an unsupervised hierarchical clustering algorithm, Chaotic Map Clustering (CMC), in a coupled two-way approach to analyse such data. However, the clustering approach is intrinsically difficult, both in terms of the unknown structure of the data and interpretation of the clustering results. It is therefore critical to evaluate the quality of any unsupervised procedure for such a complex set of data and to validate the results, separating those clusters that are due simply to noise or statistical fluctuations. We used a resampling method to perform this validation. The resampling procedure applies the clustering algorithm to a large number of random subsamples of the original data-matrix and, consequently, the whole process becomes computationally intensive and time consuming. Using Grid technology, we show that we can drastically speed up this process by distributing the clustering of each matrix to a separate worker node, and thus retrieve resampling results within a few hours instead of several days. Further, we offer an online service to cluster large microarray data sets and conduct the subsequent validation described in this paper.

## Introduction

Today, biologists can measure the expression levels of thousands of genes under different experimental conditions using DNA microarray chips. A typical biological experiment produces different data-sets of expression values, monitoring the experimental conditions applied to a tissue or a cell culture. Often, after in-house analysis, researchers deposit their raw data with one of the available public repositories, from where they can be retrieved for different kinds of analysis. Public repositories, such as GEO [1] and ArrayExpress [2], have already collected vast quantities of various microarray experiment results. Such repositories allow independent data sets of expression values, obtained with the same chip design but in different conditions and in different laboratories, to be combined and compared. Studies on such enormous amounts of heterogeneous data can reveal new and significant information, and represent an interesting and important approach that can provide new insights into the behaviour of genes. Of course, this is a difficult task, because it requires an appropriate data-normalisation process and, even more important, an efficient method of analysis, such as clustering. Because of the heterogeneity of the data sets, supervised clustering methods and parametric algorithms are unsuitable, as nothing is known *a priori* about the structure and distribution of the data. The simplest way to analyse large and mixed data-sets without any loss of information is to use an unsupervised clustering method that requires neither *a priori* assumptions about the data nor a cut-off above a fixed threshold expression value. This gives us the possibility of discovering unknown correlations between genes or unexpected behaviours in different experimental situations. Chaotic Map Clustering (CMC) [3], which we have tested successfully for microarray data (unpublished data), was the unsupervised clustering algorithm chosen. This unsupervised method of analysis of a full non-restricted data set naturally produces a lot of noise, necessitating a very accurate procedure for validation of the clustering results. We have to be able to evaluate the quality of the results and to determine the optimal settings for the clustering procedure. Resampling, based on a cross-validation method [4], is an efficient way to evaluate clustering results, telling us if they are really due to a strong correlation between

genes or if they arise simply as a result of statistical fluctuations or noise. The resampling method requires the creation and clustering analysis of random subsets of the original data-matrix. To obtain statistically relevant results requires the creation and analysis of tens of resampled matrices. With matrices such as our test-set, with a size of 22,215 x 587, the resampling validation procedure with our clustering algorithm becomes a very memory-intensive and time-consuming computational process. It is crucial to speed up this procedure in order to perform this challenging microarray data analysis within a reasonable time frame. Grid technology gives us the possibility to split and distribute the resampling procedure over different processors, allowing us to evaluate the quality of the clustering research in a few hours rather than several days. In this way, we can efficiently analyse any mixed data set, and even increase the number of experiments included within the analysis process, thus allowing analysis of even larger matrices of microarray expression values.

## Chaotic Map Clustering of microarray data

As a test-set for our clustering approach, we selected and downloaded from GEO a data-collection derived from the Affymetrix microarray design 'Human Genome U133 Array Set' (HG-U133A). This heterogeneous data-set includes 587 data samples from different laboratories and covers more than 20 different biological experiments relating to 22,215 different genes. We did not set any threshold for expression values, but considered the whole distribution of the data, from the lowest to the highest value, to be informative. Microarray data are intrinsically noisy owing to the procedure of measurement itself, but we maintain that even low expression values can have high information content, especially in such a mixed and large comparison between different biological experiments. Background and noise data are, in any case, evaluated in a second round (see below) by means of the whole procedure of clustering and validation of the results.

In order to have a comparable set of data, we scaled each data-set point by means of global normalisation, carrying out a logarithmic transformation and setting the median of the distribution of expression values of each microarray experiment to zero. We organised the data into an expression matrix, $D = N \times S$, where $N$ (=22,215) is the total number of genes in the array design, and $S$ (=587) is the number of samples (experiments).

Generally speaking, the clustering process aims to investigate and discover the unknown structure of a set of data by grouping objects that are more similar to each other according to some similarity measure. In this specific context, clustering microarray data can reveal groups of genes with similar gene-expression profiles that are co-regulated in different samples or groups of experiments.

To analyse data with no *a priori* knowledge of their structure (*i.e.*, the number of classes, or the geometric distribution), we chose an unsupervised hierarchical clustering algorithm: Chaotic Map Clustering (CMC) [3], which we have tested extensively on the clustering and analysis of heterogeneous microarray data sets (personal communication A. Tulipano).

Furthermore, to discover unknown relationships between genes within subsets of experiments that could be hidden by the signals of other genes, a coupled two-way approach [5] was applied using CMC. The first approach considers the samples as the objects to be clustered; the other considers the genes as the objects to be clustered. Using the groups of genes and samples obtained with two-way clustering, this method identifies submatrices of the total expression matrix on which further analysis can be performed locally with the user's preferred analytical tools, revealing new partitions of samples and genes and leading to new information. In this manner, by focusing on small subsets, we lowered the noise induced by the other samples and genes, and were able to discover partitions and correlations that were masked or hidden when the full data set was used in the analysis.

Details of the results, in terms of their biological relevance, are beyond the scope of the present paper and are thus not discussed further here. The focus of this report is the validation of the clustering results: we had to search for new solutions because the validation of such large data sets became computationally very intensive.

## Cluster validation

In order to evaluate the whole procedure of clustering, selecting the optimal settings for the

algorithm, and to validate the vast number of clusters found by the process, we used a cluster-validation method based on resampling [4]. This is a cross-validation procedure, where subsets of the data-matrix under investigation, whose sizes are $fN \times S$, where $0 < f < 1$ is the reduction factor, are constructed randomly, and the clustering algorithm is applied to each subset. From these results, we created an $N \times N$ connectivity matrix $T$, for each resampled matrix, whose elements are:

$$\mathbf{T}_{ij} = \begin{cases} 1 & \text{points } i \text{ and } j \text{ belonging to the same cluster} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{T}_{ii} = \begin{cases} 2 & \text{points } i \text{ is present in the resample} \\ 0 & \text{otherwise} \end{cases}$$

and compared it to the connectivity matrix of the original data-matrix.

Starting from the overlap of the original and resampled connectivity matrices, we can define three quantities [6], namely "sensitivity" (*sens*), "specificity" (*spec*) and "positive predictive value" (*ppv*), which can be regarded as useful "quality measures" of a clustering result.

To define these quantities, we considered the results obtained on the full size data-set as the "truth". According to the "truth", we have two classes: either gene i and gene j are in the same cluster (positive) or not (negative). We then compare the results obtained through resampling.

Table 1 lists all possible combinations: true positive (TP)—ij are in the same cluster, both in the original and in the resampled data-set; false negative (FN)—ij are in the same cluster in the original data-set, but not in the resampled set; true negative (TN)—ij are not in the same cluster, either in the original data-set or in the resampled one; false positive (FP)—ij are in the same cluster in the resampled matrix, but not in the original matrix.

It is now possible to define *ppv* as the average, with respect to the resamples, of the number of TP pairs divided by the number of the pairs belonging to the positive class

$$ppv = \left\langle \frac{N_{TP}}{N_{TP}+N_{FP}} \right\rangle (1)$$

and, with similar notation, *sens* is defined as

$$sens = \left\langle \frac{N_{TP}}{N_{TP}+N_{FN}} \right\rangle (2)$$

and *spec* is defined as

$$spec = \left\langle \frac{N_{TN}}{N_{TN}+N_{FP}} \right\rangle (3)$$

Evaluating these quality measures, we are able to identify stable clustering solutions, which are less likely to result from noise or statistical fluctuations, and can also evaluate the efficiency of the set of resolution parameters used for clustering.

To validate the clusters obtained by applying the CMC clustering algorithm to the original matrix of 22,215 x 587, 50 randomly resampled matrices of 16,661 x 587 (*i.e.*, a reduction factor of 25%) were generated. Each matrix was then clustered using CMC with the same set of resolution parameters. Clustering a single matrix of such a size with CMC is an intensive computational process that requires more than 1.5 GigaBytes of RAM, and takes about 2 hours of computing time (one CPU Xeon 3.0GHz). Moreover, the creation of the connectivity matrix and its comparison with the original matrix takes several hours. Clustering the whole set of resampled matrices of at least 50 random matrices and calculating the overlap of the connectivity matrices would occupy one single CPU for more than two weeks. With this method, cluster validation would be a slow and inefficient procedure. Splitting the whole cluster-evaluation process of the resampled matrices into several jobs, one for each resampled matrix, and distributing them on several CPUs would speed up the whole validation procedure, allowing quicker evaluation of the quality of our clustering.

## Grid distribution

Because of the enormous quantity of computer resources available, the Italian Grid Infrastructure (Virtual Organisation bio) [7] provides the possibility of splitting a large, complex application into many smaller jobs that can run in parallel, greatly reducing the time needed to reach the final results. Our resampling process is easily divisible into many smaller processes, namely every randomised matrix of the resampling process can be launched as an independent job.

The Job Submission Tool (JST, [8]), developed to submit and monitor a large number of jobs (in the range of hundreds of thousands) in an almost automatic way, is the engine that runs and controls the workflow to cluster large microarray

data-sets and run the necessary validation. As a test of the efficiency and functionality of our workflow (see 'Workflow Implementation and Results' below), we used the above mentioned HG-U133A test-set of 587 microarray data from Affymetrix (22,215 genes), and analysed 50 randomised matrices, 16,661 x 587 in size, for the validation. The main concept of the JST is the "task". A certain number of tasks have to be executed to complete the challenge. The entire problem is first divided into steps that depend on each other, and then each step is, if possible, subdivided into the smallest possible independent elements that can be sent to the Grid. Depending on the time required for each element, a task can consist of one or several elements, in order to optimise the performance by balancing the time needed for installation and processing. Our implementation consists of two main steps and a final step, summarising the results, the first step being clustering of the main microarray data-matrix and the second, consecutive and dependent, step concerning cluster validation. Whereas the first and last steps are single events, the second can be subdivided easily into independent elements for further processing. CMC validation for such a large matrix takes several hours, and therefore each task contains one element – a matrix. The tasks are then routed to a central database (DB) server in order to assign each task to a Grid job; the server then takes full control of challenge completion. A robot is used to submit jobs to the Grid, to Worker Nodes (WN) that are initially identical and do not know which task they have to execute. Only when the job arrives at, and starts to be executed on, a WN does it request a task to be sent from the central DB. Every job asks the central DB for a task that has not yet been assigned to any other job ("free" task). Information on the execution of each task is logged in the central DB to maintain an overview of the processing. As soon as a job submitted by the User Interface (UI) to an available WN receives the task to be executed, it starts to download from a storage element all the files (input data and libraries) required to perform the task – in our case the clustering of a single matrix and running the validation.

Only when all steps are executed correctly is the status of a particular task on the central DB updated to "Done", and the results made available on a Web server for download. In this way, the central DB monitors task execution. No manual intervention is required to manage the re-submission of failed tasks. In fact, tasks that are found in a "running" state after a fixed time interval are considered to have failed and are automatically reassigned to new jobs. As soon as no task can be assigned to the submitted jobs, meaning all tasks are labelled "Done", the robot stops submitting jobs and the processing is terminated. In this way, the architecture of the JST allows highly effective and reliable exploitation of all the computational resources available on the Grid.

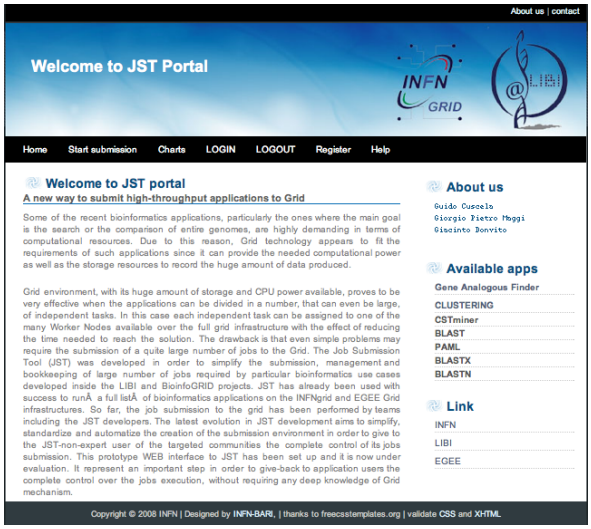


*Figure 1.* The JST portal.

Recent improvements in the JST include the implementation of a GUI (Graphic User Interface) in order to make this tool available to the bioinformatics community. The GUI (Figure 1) is available on a website [9], where users can register and, after authorisation, submit their applications to the Grid. Through the interface, the user defines parameters for the clustering application, and provides the input files that the JST will then elaborate to create the task list.

## Workflow implementation and results

To orchestrate all the steps necessary to validate the initial clustering through the JST interface, we created a workflow to cluster the original matrix, as well as to create and cluster the randomised matrices. The results were then compared with the result of the original clustering to calculate the values of *ppv*, *sens* and *spec* (Figure 2).

After login and opening the submission window (see Figure 1, 'Start submission'), JST offers a list of 'gridified' applications; choose "CLUSTERING". The input schema (see Figure 3) requires 4 types of input information: (1) one input file as a text-file containing an expression matrix, where rows refer to genes, and columns to the normalised values of the microarray data-set; (2) the number of resamplings, which establishes how many random matrices have to be produced and analysed by the CMC; (3) the distance measuring method (Euclidean or Pearson correlation); and (4) the indication whether to cluster by rows (genes) or by columns (experiments). This last parameter is important if a user wants to apply the coupled two-way approach and re-send the first clusters for the second clustering process. Before sending the submission to the Grid, JST displays all the parameters and the Grid submission commands for eventual control; only by confirming the submission at this stage does JST start to process the clustering and validation request.

The first step of the JST process consists of a single task, starting with the clustering of the original matrix and generation of the corresponding connectivity matrix. A perl script, distributed by the JST, provides the conversion of the input matrix file in a format that is accepted by the CMC clustering algorithm. Then it launches the clustering process and finally calculates the connectivity matrix of the clustering results. This step produces two outputs: a reference file of the clustering results and a text file of the connectivity matrix, both stored in a common repository. Because this connectivity matrix is needed for the tasks in the second step, JST has to make it available to every job responsible for executing these tasks. The best approach to achieve this goal is to register the file on a Grid Storage Element and to store the location in the central JST database so that every job can copy it locally in a secure and efficient way. After the first task is executed correctly, and the status of the related entry in the database is update to "Done", the Grid jobs can begin executing the second step and distributing the tasks of the second step (*i.e.*, validation). The second step consists of the random generation of each resample $D_k$, $k=1...m$ of the original data, the clustering by the CMC algorithm of this randomised matrix, generation of the related connectivity matrix $T_k$, comparison with the connectivity matrix of the original matrix,

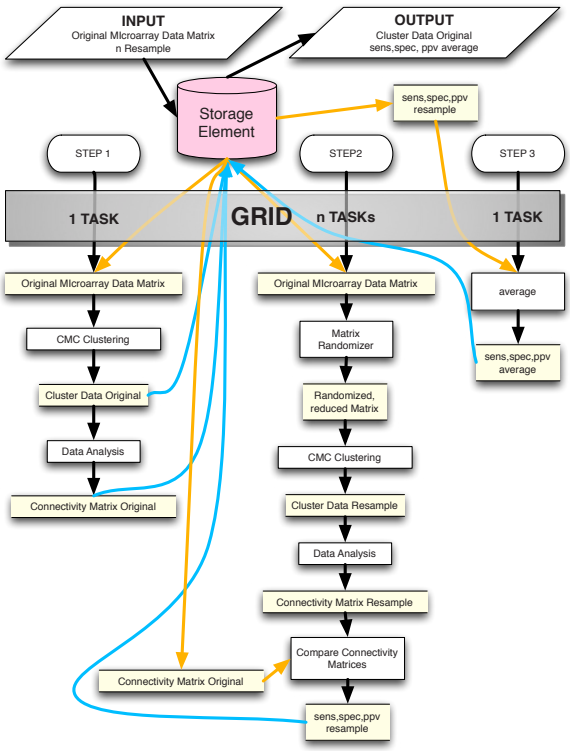


*Figure 2.* Distribution of processing over Grid nodes.
The first step of the JST process consists of a single task, starting with the clustering of the original matrix and generation of the corresponding connectivity matrix that is stored in the storage element. The second step consists of the random generation of each resample of the original data, clustering of this randomised matrix, generation of the related connectivity matrix, and computation of the *sens*, *spec* and *ppv* values. The final step retrieves all the resampling results from the storage element and calculates the average values of *ppv*, *sens* and *spec* for the whole set of resampled matrices. After all tasks are completed, the user can retrieve (from a Web link) two outputs: the clustering results of the original expression matrix, and the averaged values of the *ppv*, *sens* and *spec* file.

and computation of the quantities listed in Table 1. Again, a perl script, developed from our side and distributed by the JST, processes all these steps in order to realise the second part of the workflow. The system stores, in a common repository, one output file for each task, corresponding to a resampled matrix, with the related values of *ppv*, *sens* and *spec*. The final step retrieves all the resampling results and calculates the average values of *ppv*, *sens* and *spec* for the whole set of resampled matrices by means of a third perl script specifically implemented for this task. After all steps are completed, the user can retrieve (from a Web link sent by e-mail) two outputs: the clustering results file of the original expression matrix (step 1) and the averaged values of *ppv*, *sens* and *spec* file (step 3).

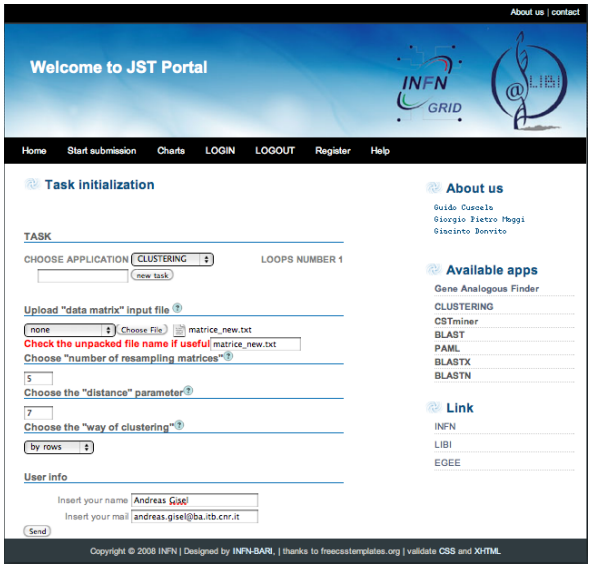


*Figure 3*. Input parameters.
The user uploads one input file containing genes (in rows) and the experiments of the microarray data-set (in columns), and sets the number of resamplings to execute and the method of distance measure (Euclidean or Pearson correlation).

For the test-case used here, a data-matrix of dimension of 22,215 x 587 and validation using 50 randomised, reduced matrices, distributed over the Grid (Figure 1) using one WN for each matrix to be analysed, we were able to reduce the processing time by 20-fold, from 16 days on a single CPU to 20 hours distributed over the Grid. The average execution time was about 8 hours per job. Considering that we have two steps, with the second being dependent on the result of the first, we have a net processing time of 16 hours, losing therefore about 4 hours of the total process time in job queuing. Nine matrices had to be resubmitted because of various different problems. One problem was the memory requirement (>1.5 GB RAM) of the clustering algorithm for a matrix of the specified size, which was a requirement that not all available WNs could satisfy.

The average values of *ppv*, *sens* and *spec* calculated for the given data set were 0.65, 0.81 and 0.95, respectively. Owing to the size of the data set, and the high level of fragmentation (more than 300 clusters), we can expect that the number of true negatives is orders of magnitude greater than the other quantities defined in Table 1. This means that, for the case at hand, the specificity values would be close to 1 even in the case of random or incorrect clustering results. For our purposes, we thus consider only *ppv* and *sens* as the relevant quality measures.

To illustrate the results in more detail, Figures 4 and 5 show the distributions obtained for the values of *ppv* and *sens* for the 50 resampled matrices. We can see from the histograms that about half the resamples exhibit sensitivity values above 0.95, and that, in almost all cases, *ppv* values are above 0.5. As it is well known that the statistical significance of the quality measures is affected strongly by the size of the data set, and by the level and nature of noise in the data, it has long been recognised that there is substantial intrinsic noise contained in microarray data. We stress that the values obtained here far exceed what is generally considered a good result in such a context, and we thus conclude that our results validate the approach proposed here.
Incidentally, we recall that re-sampling procedures can also be applied to search for an optimal set of algorithm parameters. In this case, resampling procedures need to be repeated for any choice of the clustering parameters, and records of the corresponding quality measures have to be collected. The optimal values of the parameters are then selected as those corresponding to the maximum value of *spec* and/or *ppv*. However, as we have already tested (personal communication A. Tulipano) the robustness of the algorithm against changes in the main scale parameter $\alpha$, we limited ourselves to performing the resampling in order to assess the quality of the clustering results. However, we expect that keeping the scale parameter fixed on both the original and resampled data-set will add a slight negative bias to the *ppv* measures by increasing the number of false positives.

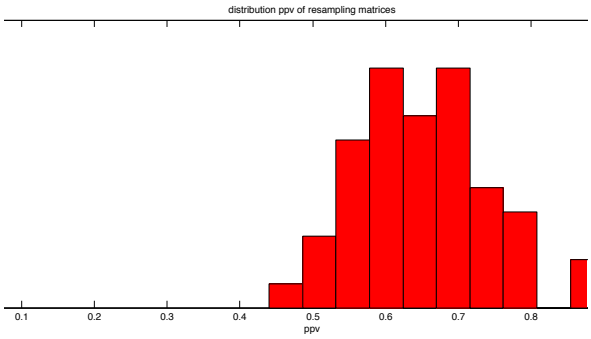


*Figure 4*. Distribution of ppv values obtained for the resampled matrices.

## Conclusion

The approach described above, that is to distribute the process of validating clustering results on the Grid, proved to be a very valid implementation for large microarray data sets. The analysis of large mixed microarray data with the CMC algorithm is a very complex procedure, requiring an efficient method of result validation. The two key functionalities of our approach are the use of the JST, modified for this application, which has already been used by other applications to distribute workload efficiently over the Grid [8], and the newly developed three-step workflow to calculate and process the resampled matrices in parallel. Because the matrices are so large, and the clustering algorithm and associated procedure of evaluation is computationally intensive, the original matrix and every resampled matrix was submitted to an independent WN, allowing the total validation to be run in parallel, completing the calculation in a fraction of the original time required when using only one CPU.

This drastic improvement in the validation process will allow researchers to analyse any combination of available data sets with almost no limit to the data size and number of resampling cycles, to guarantee reasonable accuracy of the analysis and validation. In addition, the pipeline allows users not only to run time-consuming validation processes, but also provides users with the clustering data, and can therefore be used to run time-consuming clustering on large data-sets using CMC.

The only problem we were confronted with was the large amount of RAM required by the CMC clustering algorithm operating on a large dataset. However, very few WNs were not adequately equipped to execute the clustering algorithm. In such cases, the JST resubmitted those failing jobs to new WNs without requiring any user interaction, providing the end user with a complete and accurate picture for further analysis.

## Authors' contributions

AT was responsible for the CMC algorithm, its implementation, the development of the validation process and preparation of the manuscript. CM was involved in the implementation of the CMC algorithm, implementation of the validation process and preparation of the manuscript. LA was responsible for the development and implementation of CMC algorithm. GD was responsible for the implementation of the Grid distribution and development of the graphical interface. GC was responsible for the implementation of the Grid distribution, development of the graphical interface and preparation of the manuscript. GPM was responsible for the implementation of Grid distribution. AG was responsible for the whole project, implementation of the workflow and preparation of the manuscript.

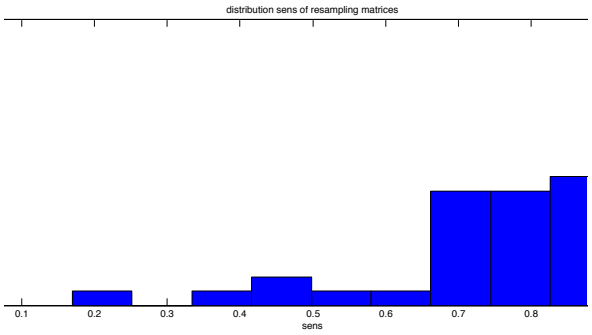


*Figure 5.* Distribution of sens values obtained for the resampled matrices.

## Acknowledgements

The authors thank Prof. Mario Pellicoro for fruitful discussions about the implementation of the CMC algorithm and Nicola Losito for technical support. We also thank Dr. Helen Rothnie for the English corrections. We would also like to thank the INFN Grid production Resource Centre's administrators and the INFN-GRID/GRID.IT management. This work was funded for INFN Bari by the MIUR (Italian Ministry for Education, University and Research) in the LIBI "International Laboratory of BioInformatics" project, F.I.R.B. 2003, under grant RBLA039M7M; and for ITB by the EU project EMBRACE, which was funded by the European Commission within its FP6 Programme, under the thematic area 'Life sciences, genomics and biotechnology for health', contract No. LUNG-CT-2004-512092.

### Competing interest statement

None declared

## References

1. http://ncbi.nlm.nih.gov/geo.
2. http://www.ebi.ac.uk/arrayexpress.
3. Angelini L, De Carlo F, Marangi C, Pellicoro M, Stramaglia S (2000) Clustering Data by Inhomogeneous Chaotic Map Lattices. Phys Rev Lett 85: 554-557.

4. Levine E, Domany E (2001), Resampling method for unsupervised estimation of cluster validity. Neural Comp 13: 2573-2593.
5. Getz G, Levine E, Domany E (2000) Coupled two-way clustering analysis of gene microarray data. Proc Natl Acad Sci U S A 97: 12079-12084.
6. Van deer Laan MJ, Bryan J (2001) Gene expression analysis with the parametric bootstrap. Biostatistics 2: 445-461.
7. http://www.italiangrid.org/
8. De Sario G, Tulipano A, Donvito G, Maggi G, Gisel A (2009) High-throughput Grid computing for Life Sciences. In: M. Cannataro editor. Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine, and Healthcare.
9. http://webcms.ba.infn.it/~jst/JST/